# Non-Convex and Multi-Objective Optimization in Data Mining

Non-Convex and Multi-Objective Optimization for
Statistical Learning and Numerical Feature Engineering

**Dissertation**

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik
von

Ingo Mierswa

Dortmund
2009

| | |
|---|---|
| Tag der mündlichen Prüfung: | 27.04.2009 |
| Dekan: | Prof. Dr. Peter Buchholz |
| Gutachter: | Prof. Dr. Katharina Morik<br>Prof. Dr. Claus Weihs |

## Acknowledgments

I would like to express my gratitude to my supervisor, Prof. Dr. Katharina Morik, whose expertise, understanding, patience, and personality added considerably to this thesis and the last years of my life. I appreciate her vast knowledge and skill and that she always gave me the freedom to work on topics of my preference and – maybe even more important – to develop my own style of work. During the last years, she became more of a mentor and friend than a professor to me.

I would like to thank Prof. Dr. Claus Weihs for the assistance he provided and his valuable hints how I could improve my thesis. It was really helpful to get those comments on all levels of detail and especially to get these comments from a statistician's point of view.

I would like to thank the members of the Artificial Intelligence Group, past and present. I find it quite interesting that the time we played soccer almost every day on a parking lot was also the time everyone of us published most of his work. But as we all know: correlation does not necessarily means causality.

I recognize that this research would not have been possible without the financial assistance of the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) who supported this work within two collaborative research centers (Sonderforschungsbereiche, SFB), namely the SFB 531 "Design and Management of Complex Technical Processes and Systems by Means of Computational Intelligence Methods" and the SFB 475 "Reduction of Complexity in Multivariate Data Structures".

Last but not least I would like to thank my family for the support they provided me through my entire life and in particular, I must acknowledge my wife and best friend, Nadja.

*According to the latest official figures,*
*43% of all statistics are totally worthless.*

# Contents

**List of Tables** **15**

**List of Figures** **19**

**List of Notations** **25**

**1. Introduction** **31**
1.1. Three Theses about Data Mining . . . . . 31
1.2. Related Work . . . . . 36
1.3. Outline . . . . . 39

**2. Basics** **41**
2.1. Machine Learning . . . . . 41
2.1.1. Supervised Learning . . . . . 42
2.1.1.1. Classification Learning . . . . . 43
2.1.1.2. Regression Learning . . . . . 43
2.1.2. Unsupervised Learning . . . . . 43
2.2. Statistical Learning . . . . . 44
2.2.1. Regularized Risk Minimization . . . . . 44
2.2.1.1. Bound on the Generalization Performance . . . . . 46
2.2.2. Large Margin Methods . . . . . 47
2.2.2.1. Support Vector Machines . . . . . 47
2.2.2.2. Non-Separable Data . . . . . 50
2.2.2.3. Non-Linear Learning with Kernels . . . . . 51
2.3. Optimization . . . . . 53
2.3.1. Linear Programming . . . . . 54
2.3.2. Quadratic and Non-Linear Programming . . . . . 55
2.3.2.1. Nelder-Mead Optimization . . . . . 55
2.3.2.2. Newton Optimization . . . . . 55
2.3.3. Non-Convex Programming . . . . . 57

2.3.3.1. Evolutionary Algorithms . . . . . 57
2.4. Multi-Objective Optimization . . . . . 59
2.4.1. Multi-Objective Evolutionary Optimization . . . . . 60
2.4.1.1. Guided Multi-Objective Optimization . . . . . 61

**I. Learning** **63**

**3. Multi-Objective Learning** **65**
3.1. Single-Objective Evolutionary Support Vector Machines . . . . . 65
3.1.1. Motivation for Evolutionary Support Vector Machines . . . . . 66
3.1.2. Evolutionary Computation for Large Margin Optimization . . . . 68
3.1.2.1. Solving the Dual Problem and Other Simplifications . . . 68
3.1.2.2. EvoSVM and PsoSVM . . . . . 69
3.1.3. Experiments and Results . . . . . 71
3.1.3.1. Data Sets . . . . . 71
3.1.3.2. Comparison for the Objective Function . . . . . 71
3.1.3.3. Comparison for Positive Kernels . . . . . 72
3.2. Multi-Objective Statistical Learning . . . . . 77
3.2.1. The Regularized Risk Consists of Multiple Objectives . . . . . 77
3.2.2. Explicit Trade-off between Error and Complexity . . . . . 79
3.2.3. First Objective: Maximizing the Margin . . . . . 80
3.2.4. Second Objective: Minimizing the Number of Training Errors . . . 82
3.2.5. Multi-Objective Evolutionary Algorithms for Large Margin Learning 83
3.2.5.1. Definition of the Objectives . . . . . 83
3.2.5.2. The Multi-Objective EvoSVM . . . . . 84
3.2.6. Selecting a Solution from the Pareto Set . . . . . 84
3.2.7. Experiments and Results . . . . . 86
3.2.7.1. Interpretation of the Pareto Fronts . . . . . 86
3.2.7.2. Multi-Objective Optimization vs. Multiple Single-Objective Runs . . . . . 91

**4. Non-Convex Optimization for Statistical Learning** **93**
4.1. Non-Positive Semidefinite Kernel Functions . . . . . 93
4.1.1. The Relevance Vector Machine: A Kernel Method for Indefinite Kernel Functions . . . . . 95
4.2. Experiments and Results . . . . . 99
4.2.1. Evolutionary Computation for Non-Convex Optimization . . . . . 99
4.2.2. Data Sets . . . . . 99
4.2.3. Comparison for Non-positive Kernels . . . . . 100

**5. Transductive Learning: Non-Convex and Multi-Objective 103**
5.1. Motivation of Transductive Learning . . . 103
5.1.1. Problem Definition . . . 105
5.2. Dual Optimization Problems for Transductive SVM . . . 106
5.2.1. First Objective: Maximizing the Margin . . . 107
5.2.2. Second Objective: Minimizing the Training Error . . . 112
5.2.3. Third Objective: Minimizing the Test Error . . . 113
5.3. Single-Objective but Non-Convex: The Evolutionary TSVM . . . 115
5.3.1. Experiments and Results . . . 116
5.4. Multi-Objective TSVM . . . 118
5.4.1. Experiments and Results . . . 122
5.4.1.1. Interpretation of the Pareto Fronts . . . 122
5.4.2. From Classification to Clustering in One Single Run . . . 123

**II. Feature Space Transformations 127**

**6. Multi-Objective Supervised Feature Construction 129**
6.1. Feature Space Transformations . . . 130
6.1.1. Feature Construction and Genetic Programming . . . 131
6.1.2. Feature Construction and Kernels . . . 132
6.1.3. Multi-Objective Feature Space Transformations . . . 132
6.2. Multi-Objective Evolutionary Feature Selection . . . 133
6.2.1. Interpretation of the Pareto Fronts . . . 134
6.3. Regularized Feature Selection and Construction . . . 135
6.3.1. Regularized Risk for Feature Space Transformations . . . 136
6.3.2. Definition of $\Omega(X)$ for Feature Selection and Feature Construction 136
6.4. Multi-Objective Evolutionary Feature Construction . . . 138
6.4.1. Problem-Specific Search Operations . . . 138
6.4.1.1. Mutations . . . 138
6.4.1.2. Crossover . . . 139
6.4.1.3. Selection . . . 139
6.4.2. Code Bloat and Intron Prevention . . . 140
6.4.2.1. Intron Prevention by Sampling Equivalence Checks . . . 141
6.4.3. A New Generating Mutation for Intron Prevention . . . 141
6.5. Experiments and Results . . . 142
6.5.1. Interpretation of the Pareto Front . . . 145

**7. Multi-Objective Supervised Feature Extraction 149**
7.1. Feature Extraction from Audio Data . . . 150
7.2. Methods for Feature Extraction . . . 152
7.2.1. Basis Transformations . . . 153

7.2.1.1. Frequency Space . . . 154
7.2.1.2. Correlation Space . . . 154
7.2.1.3. Reconstruction of the State Space . . . 155
7.2.1.4. Reversibility . . . 156
7.2.2. Filters . . . 157
7.2.3. Mark-up of Intervals . . . 157
7.2.4. Generalized Windowing . . . 158
7.2.5. Functions . . . 159
7.2.6. Some Properties of the Methods . . . 161
7.3. Adaptive Construction of Method Trees . . . 164
7.3.1. Representation . . . 166
7.3.2. Mutation, Crossover, and Selection . . . 166
7.3.2.1. Fitness Evaluation . . . 167
7.3.3. Some Properties of the Search Space . . . 168
7.3.3.1. Size of the Search Space . . . 168
7.3.3.2. Processing a Method Tree . . . 169
7.4. Classification Using Learned Method Trees . . . 171
7.4.1. Classifying Genres . . . 172
7.4.2. User Preferences . . . 174
7.5. Multi-Objective Feature Extraction . . . 175
7.5.1. Experiments and Results . . . 175
7.5.2. Interpretation of the Pareto Front . . . 176

**8. Multi-Objective Unsupervised Feature Selection 181**
8.1. Unsupervised Feature Selection . . . 182
8.2. Data Clustering . . . 183
8.2.1. Combinatorial Clustering Algorithms . . . 184
8.2.2. Gaussian Mixtures . . . 185
8.2.3. Graph-Based Clustering . . . 186
8.3. Multi-Objective Feature Selection for Clustering . . . 188
8.4. Information Preserving Feature Selection . . . 189
8.4.1. Finding Interesting Points in the Pareto Front . . . 190
8.5. Experiments and Results . . . 191
8.5.1. The Data Sets . . . 191
8.5.2. Interpretation of the Pareto Fronts . . . 192
8.5.3. Pareto Front Segmentation . . . 197

**9. Multi-Objective Feature Space Transformation for Clustering 201**
9.1. Motivation for Feature Space Transformations . . . 202
9.2. Information Preserving Feature Aggregation . . . 203
9.2.1. Definition of Feature Aggregation . . . 203
9.2.2. Information Preserving Feature Aggregation . . . 204

9.2.3. Domain Preserving Feature Aggregation . . . . . 205
9.3. Criteria for Multi-Objective Unsupervised Feature Space Transformations 206
9.4. Experiments and Results . . . . . 207
9.4.1. Interpretation of the Pareto Fronts . . . . . 209

**10. Feature Set Transfers** **211**
10.1. Motivation for Feature Set Transfers . . . . . 211
10.2. Basic Concepts . . . . . 213
10.3. Comparing Learning Tasks Efficiently . . . . . 214
10.4. Negative Results . . . . . 216
10.5. Positive Results . . . . . 217
10.6. Experiments and Results . . . . . 220
10.6.1. Synthetical Data . . . . . 220
10.6.2. Real World Data . . . . . 223
10.7. Exploiting the Similarity of Constructed Features . . . . . 223
10.7.1. Similarity of Constructed Features . . . . . 225
10.7.2. Decreasing Runtime using a 2-Phase Distance Calculation . . . . . 227
10.8. Experiments and Results . . . . . 227

**11. Conclusion** **233**
11.1. Multi-Objective and Non-Convex Learning . . . . . 233
11.2. Multi-Objective Feature Space Transformations . . . . . 235

**Bibliography** **243**

# List of Tables

3.1. The evaluation data sets. $n$ is the number of data points, $m$ is the dimension of the input space. The kernel parameter $\sigma$ was optimized for the comparison SVM learner *mySVM*. The last column contains the default error, i. e. the error for always predicting the major class in percent. . . . 72
3.2. Comparison of the different implementations with regard to the objective function (the higher the better). The results are obtained by a 20-fold cross validation, the time is the cumulated time for all runs. It can easily be seen that the evolutionary version of the SVM always provides better results for the objective function than the quadratic programming solutions. Bold fonts mark significantly better results on a 1% confidence level. . . . . 73
3.3. Comparison of the different implementations with regard to the classification error (the lower the better). The results are obtained by a 20-fold cross validation, the time is the cumulated time for all runs. There is no significant difference between the results on a 1% significance level according to an ANOVA test. . . . . 74
3.4. Classification error (the lower the better), standard deviation, and runtime of all SVMs on the evaluation datasets for parameters $C = 1$, $C = 0.1$, and $C = 0.01$. The runtime $T$ is given in seconds. The last line for each table depicts the F test value and the probability that the results are not statistical significant. . . . . 75
3.5. The evaluation data sets. $n$ is the number of data points, $m$ is the dimension of the input space. The kernel parameter $\sigma$ was optimized with a grid parameter search. The last column contains the default error, i. e. the error for always predicting the major class. . . . . 85

4.1. The evaluation data sets. $n$ is the number of data points, $m$ is the dimension of the input space. The kernel parameters $\sigma$ and $d$ were optimized for the comparison SVM learner *mySVM*. The last column contains the default error, i. e. the error for always predicting the major class. . . . . 100

4.2. Comparison of the different implementations with regard to the classification error (the lower the better) for a non-positive semidefinite kernel function (Epanechnikov). The results are obtained by a 20-fold cross validation, the time is the cumulated time for all runs. Bold fonts mark significantly better results on a 1% confidence level. . . . . 102

5.1. The data sets used for transductive learning. Please note that most of the examples are not labeled. . . . . 116

6.1. The different tasks for multi-objective evolutionary feature space transformations. X denotes tasks solved by the author of this work, O denotes tasks solved by other authors, ? denotes tasks which are currently not yet solved. . . . . 133
6.2. The feature sets of the Pareto front presented in Figure 6.8. The inherent structure can again be seen, adding more features than three hardly improves the prediction performance. . . . . 148

7.1. Classification of genres with a linear SVM using the task specific feature sets. . . . . 172
7.2. Classification performance using the same non-tailored standard feature set for all classification tasks (linear SVM). . . . . 173
7.3. Classification errors with respect to different learning schemes. . . . . 173
7.4. Classification according to user preferences. . . . . 174
7.5. Comparison of the genetic programming based method tree learning approach and a simple windowed linear regression approach for time series predictions. The results are obtained by a 10-fold back-testing validation. The bold font marks a significantly better result on a 1% confidence level for the feature extraction approach. . . . . 179

8.1. The used data sets. The first column summarizes the abbreviations used in the text, the second summarizes some properties of the data set. $n$ is the total number of examples, $m$ the total number of features. The column *noise* defines how many features of $m$ where explicitly added noise features. The next columns define the mean standard deviation of the original features ($\sigma_o$) and the noise features ($\sigma_n$). The column $k$ indicates the number of clusters if it is known. The last column indicates which Pareto sets were found for the data set with both approaches. . . . 193

9.1. The used data sets for unsupervised feature space transformation. . . . . 207

10.1. The achieved accuracy using base features only and feature recommendations (based on Random, Manhattan, and Euclidian distances of base feature weights). . . . . 223

10.2. The averaged relative errors for the different approaches. The symbol $\infty$ indicates that no result was produced in a reasonable amount of time. . . 228

# List of Figures

1.1. A set of data points. The target is to predict the value on the y-axis from the location on the x-axis. . . . . 32
1.2. A complex model which minimizes the training error: this model just connects all points. . . . . 32
1.3. The most simple model for this set of data points: a simple linear equation which produces errors even on the given data points. . . . . 33
1.4. A probably perfect trade-off between model complexity and error on the given data points: no errors are made and the model seems to be sufficiently simple. . . . . 34

2.1. The simultaneous minimization of empirical risk and model complexity gives a hint which function should be used in order to generalize the given data points. . . . . 46
2.2. A simple binary classification problem for two classes $-1$ (empty bullets) and $+1$ (filled bullets). The separating hyperplane is defined by the vector $w$ and the offset $b$. The distance between the nearest data point(s) and the hyperplane is called *margin*. . . . . 48
2.3. After the transformation of all data points into the feature space $H$ the non-linear separation problem can be solved with a linear separation algorithm. In this case, a transformation in the space of polynomials with degree 2 was chosen. . . . . 52

3.1. A simple hybrid mutation which should speed-up the search for sparser solutions. It contains elements from standard mutations from both genetic algorithms and evolution strategies. . . . . 70
3.2. The left plot for each dataset shows the Pareto front delivered by the multi-objective EvoSVM proposed in this work (x: training error, y: margin size). The right plot shows the training (+) and testing (×) errors (on a hold-out set of 20%) for all individuals of the resulting Pareto fronts (x: Pareto solution counter, y: errors). Part 1 of the results. . . . . 87

3.3. The left plot for each dataset shows the Pareto front delivered by the multi-objective EvoSVM proposed in this work (x: training error, y: margin size). The right plot shows the training (+) and testing (×) errors (on a hold-out set of 20%) for all individuals of the resulting Pareto fronts (x: Pareto solution counter, y: errors). Part 2 of the results. . . . . 88
3.4. The first plot shows a simple 2-dimensional data set consisting of a global model (the hyperplane in the middle) together with some local models (the circular regions on the wrong side of the plane) and noise. The following pictures show the predictions of the different types of models of a resulting Pareto front. Model 1 provides the largest margin, it is actually so large that it is located beside the complete training data. Model 2 is a model corresponding to a point further on the right side where the global model is already found but the local models are not yet found. Model 3 shows a model further to the right where the first local models are found and Model 4 demonstrates the best model identifying both the global model and the most important local models. Finally, Model 5 already shows the clear overfitting to the noise data. . . . . 90

4.1. 2D-plots of some known non-positive semidefinite kernel functions together with their 20 main Eigenvalues. These Eigenvalues are partly negative. The variables $s$ and $t$ represent data points $x$ and $p$, $c$ and $\sigma$ are parameters used for fine-tuning the kernel functions. Source: [140]. . 96

5.1. The data sets used for transductive learning. Grey dots mark unlabeled data points. Please note for the 4-clusters data set that the distance along the y-axis is much bigger (about factor 8) than the distance between the clusters along the x-axis. . . . . 117
5.2. Predictions of both SVM approaches for the 4-Clusters data set. The traditional induction SVM calculated a diagonal hyperplane which decided to group the upper left and the lower right point with the lower left point. The transductive SVM made a better job by taking the distance between the unlabeled points also into account: the hyperplane is parallel to the x-axis and distinguishes between the upper and the lower points. . . . . 119
5.3. Confidences of both SVM approaches for the 4-Clusters data set. The inductive SVM calculates a diagonal hyperplane (indicated by the confidence values around 0.5 for the upper left and the lower right points). The transductive SVM, on the other hand, calculates a hyperplane clearly distinguishing between the upper and the lower points. . . . . 119
5.4. Predictions of both SVM approaches for the 3-Clusters data set. Both SVMs perform equally well in terms of the prediction on the given data set. However, the transductive SVM would perform better on completely unknown data since it takes the position of the middle cluster into account. 120

5.5. Confidences of both SVM approaches for the 3-Clusters data set. It can clearly be seen that the transductive SVM would make less errors on additional points from the middle cluster since the hyperplane is located between the middle and the lower cluster. . . . . 120
5.6. Predictions of both SVM approaches for the 2-Moons data set. It can clearly be seen that the non-transductive SVM was not able to capture the structure of the underlying data set. Major parts of the lower moons were wrongly predicted by the non-transductive learner. . . . . 121
5.7. Confidences of both SVM approaches. The semi-transparent colors indicate the values of the prediction function of both SVMs. It can clearly be seen that the transductive SVM was able to take the spacial structure of the unlabeled data points into account. . . . . 121
5.8. The resulting Pareto front in 2D and in 3D plots for the data set 4-Clusters. The color and the z-axis corresponds to criterion ($I$). . . . . 124
5.9. The resulting Pareto front in 2D and in 3D plots for the data set 3-Clusters. The color and the z-axis corresponds to criterion ($I$). . . . . 124
5.10. The resulting Pareto front in 2D and in 3D plots for the data set 2-Moons. The color and the z-axis corresponds to criterion ($I$). . . . . 125

6.1. The figure shows the resulting Pareto front for a simple learning problem containing only 15 equally important features. The analyst can derive a feature ranking from such a Pareto front and gets also a hint where such a ranking does not apply (e.g. between subset size 4 and 5). . . . . 135
6.2. Modified one-point crossover for individuals with variable lengths. All crossover variants ensure than single features are not added more than once to an individual. . . . . 139
6.3. A simple non-linear function which can hardly be completely learned by state of the art learning schemes without feature construction. . . . . 143
6.4. A regression SVM (RBF kernel) model built on the discussed data set. The influence of the noise attribute can clearly be seen, also the missing support vectors at the data space edges affect the model quality. . . . . 144
6.5. A regression SVM (RBF kernel) model built on the discussed data set after the noise attribute was removed. The missing support vectors at the data space edges still affect the model quality. . . . . 144
6.6. A regression SVM (RBF kernel) model built on the discussed data set after the noise attribute was removed and the value for the parameter $C$ was drastically increased. Although the error at the edges vanished, the learning takes too long time now and overfitting to the label noise begins. 145
6.7. A linear regression model based on the newly constructed features. The overall structure is found and overfitting does not occur. In contrast to the optimized SVM learning run of Figure 6.6, the learning procedure needs less than one second and the new attributes give additional insights. 146

6.8. The feature ranking induced by the multi-objective feature construction approach discussed in this chapter. It can clearly be seen that the first three features are most important for this data set. . . . . 147

7.1. The overall process of automatic feature construction for classification. . . 152

7.2. Overlay of two curves, $\nu_1 = 2Hz, a_1 = 3$ and $\nu_2 = 8Hz, a_2 = 1$, shown left in time space, right in frequency space after a Fourier transformation. 154

7.3. Autocorrelation differences for a phase shift depending on speeds ranging from 90 to 170 beats per minute. . . . . 155

7.4. Phase space representation of a popular song (left) and a classical piece (right) created with the discussed state space reconstruction with $d = 1$ and $m = 2$. . . . . 156

7.5. Intervals found in the index dimension are summarized. . . . . 157

7.6. The process of finding intervals in a series (a), first in the value dimension (b), then projected on the index dimension (c), delivering (d). . . . . 159

7.7. Constructing the cepstral method from elementary extraction operators. . 161

7.8. A method tree for feature extraction built of elementary methods. Solid arrows show the data flow, dashed lines define the tree structure. . . . . 165

7.9. Automatic feature extraction using genetic programming. . . . . 166

7.10. XML method tree representation for RapidMiner. . . . . 167

7.11. Classifier learning step using the best method tree found by the genetic programming approach. . . . . 171

7.12. The daily S&P 500 index between January 1st, 1980 and October 8th, 1992. The task is to predict the value for the next day from the values which were encountered in the past. . . . . 176

7.13. The Pareto front after generation 11. Although not all dominated points are removed yet, the Pareto front can already be seen. . . . . 177

7.14. A medium-sized feature extraction method tree from the Pareto front shown in Figure 7.13. . . . . 178

8.1. The SPIRAL data set created for single link clustering. . . . . 187

8.2. The Pareto fronts for all data sets. The left result for each dataset is achieved by the approach discussed in section 8.3 for a normalized value $\omega_{DB,norm}$ ($DB_{norm}$). It can clearly be seen that these results are not as complete and that kinks are covered by the artificial inversely proportional structure. The results on the right are achieved by our information preserving maximization approach. Part 1 of the results. . . . . 194

8.3. The Pareto fronts for all data sets. The left result for each dataset is achieved by the approach discussed in section 8.3 for a normalized value $\omega_{DB,norm}$ ($DB_{norm}$). It can clearly be seen that these results are not as complete and that kinks are covered by the artificial inversely proportional structure. The results on the right are achieved by our information preserving maximization approach. Part 2 of the results. . . . . 195
8.4. The Pareto fronts for all data sets. The left result for each dataset is achieved by the approach discussed in section 8.3 for a normalized value $\omega_{DB,norm}$ ($DB_{norm}$). It can clearly be seen that these results are not as complete and that kinks are covered by the artificial inversely proportional structure. The results on the right are achieved by our information preserving maximization approach. Part 3 of the results. . . . . 196
8.5. We applied information preserving feature selection on the real-world data set WPBC. The number of features $nf$ ($F$ in the plot), the Davies Bouldin clustering criterion $\omega_{DB}$ ($DB$ in the plot), and the number of clusters $k$ ($K$ in the plot) should be simultaneously optimized. The result is a three dimensional Pareto set containing all necessary information allowing a decision about the best clustering. The kinks could be used to segment the Pareto set and ease the analysis of the front. . . . . 198
8.6. The five points with the highest absolute deviation of $\Delta\eta_p$ to 1 are marked with perpendicular lines. This leads to an interpretable segmentation of the Pareto front which eases the process of selecting a final solution from the Pareto set. . . . . 199

9.1. The Pareto fronts delivered by the unsupervised multi-objective feature aggregation experiments on the IRIS-M data set. The Pareto sets still cover the complete range of possible solutions from 1 until 8 features for the IRIS-M data set. Additionally, features were only aggregated if this combination was necessary. . . . . 208
9.2. The Pareto fronts delivered by the unsupervised multi-objective feature aggregation experiments on the KDDCUP-2004 data set. . . . . 208
9.3. The Pareto fronts delivered by the unsupervised multi-objective feature aggregation experiments on the NEWSGROUPS data set. . . . . 209

10.1. Overview of the case-based feature construction process. Source: [138]. . . 214
10.2. The base feature weights of the synthetical test cases and for the real world cases after a dimensionality reduction on two dimensions. . . . . 221
10.3. The results of case based feature construction. The averaged relative error of all 200 test cases is plotted against the number of cases used as case base. The combination SVM weights plus Manhattan distance clearly outperforms the combination Relief plus Euclidian distance. This applies especially for data sets containing alternative features. . . . . 222

10.4. We improve the usual evolutionary based feature construction algorithms like those described in Chapter 6 and 7 by adding the discussed case base feature retrieval as additional mutation. Source: [138]. . . . . 224
10.5. The main routine to calculate the distance between two different feature sets including constructed attributes. . . . . 225
10.6. The routine `calc_distance` which calculates the distance between two constructed features using a range sensitive sampling and the squared correlation on a small artificially generated data set with size $m$. . . . . 226
10.7. The results of case based feature construction. The averaged relative error of all 241 test cases is plotted against the number of cases used as case base. The 2-phase approach clearly outperforms the SVM plus Manhattan distance. . . . . 229
10.8. The performance of YAGGA2 and YAGGA3 with respect to the number of generations. YAGGA3 performs a case base lookup in the 5th, the 15th, and the 25th generations which results in a clear performance gain and faster convergence times. Source: [138]. . . . . 230
10.9. Average performance of 10 runs for 10 different data sets after a runtime of 100 seconds. Both algorithms are performed 10 times on each data set. The bars denote the average performance (root mean squared error) and, hence, lower bars are better. It can clearly be seen that YAGGA3 leads to much smaller errors for most of the data sets. Although the standard deviations are not shown in this plot, YAGGA3 delivers significantly better results in 7 out of the 10 cases. Source: [138]. . . . . 230
10.10Another comparison on the same 10 data sets between YAGGA2 and YAGGA3. The generation number was limited to 100. The best 2 (5) cases delivered for each retrieval process formed the base for feature construction. Although not shown, YAGGA3 based on 5 cases delivers significantly better results in 9 out of the 10 cases. Source: [138]. . . . . 231

# List of Notations

**Sets**

$\mathbb{B}$ : set of boolean numbers

$\mathbb{N}$ : set of natural numbers

$\mathbb{R}$ : set of real numbers

$\mathbb{R}^m$ : the $m$-dimensional real-valued space $\mathbb{R} \times \ldots \times \mathbb{R}$

$\mathbb{C}$ : set of complex numbers, i.e. the ordered pairs of real numbers $(a, b)$ with $a + b \cdot j$ and the imaginary unit $j$ with the property $j^2 = -1$

**Basic Operations and Definitions**

$i$ : index variable

$j$ : index variable, also the imaginary unit with $j^2 = -1$ (see Sets)

$k$ : index variable, also the number of clusters (see Clustering)

$l, p, q, r, s$ : other index variables

$e$ : Euler's number, $e \approx 2.71828$

$f$ : a generic function, often the one which should be fitted to the training data (see Risk Minimization)

$g, h$ : other generic functions

$r$ : a generic objective function which should be minimized or maximized, also used with index $r_i$ in the case of multi-objective optimization

$\gamma_1 \succ \gamma_2$ : a solution $\gamma_1$ dominates a solution $\gamma_2$ if $\gamma_1$ is at least as good as $\gamma_2$ with respect to all criteria and is actually better with respect to at least one criterion

$\times$ : Cartesian (set) product, for example $X \times Y$ refers to the set of all tuples $(x, y)$ with $x \in X$ and $y \in Y$

$\langle x, x' \rangle$ : the dot product (scalar product) of the vectors $x$ and $x'$, resembles a similarity function

$||x||$ : the norm of $x$, defined as $\sqrt{\langle x, x \rangle}$ in a vector space with dot product

$\Phi$ : a mapping into a new space with dot product (often this new space is called feature space)

$k(x, x')$ : a kernel function, corresponds to the dot product in a new space after a mapping $\Phi$, i.e. $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$, resembles a similarity function in this new space

$K$ : kernel matrix consisting of the values for the pairwise kernel function results for a given set of vectors, the entries are denoted by $K_{ij}$

$d(x, x')$ : distance function, e.g. the Euclidean distance $d(x, x') = \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2}$

**Examples (Data)**

$X$ : instance space, composed of attributes (random variables) $X = X_1 \times \ldots \times X_m$, also denotes the set of all possible unlabeled observations

$X_k$ : the $k$-th attribute or feature of the instance space, each attribute can be considered as a random variable following a certain probability distribution

$x$ : a single observation $x \in X$ in attribute-value representation, i.e. $x$ is a vector where the component $x_k$ refers to the attribute $X_k$

$x_i$ : the $i$-th observation in an instance set $X$

$Y$ : label / target space, a single random variable, also denotes the set of all possible labels for observations, often assumed to be boolean $Y = \{0, 1\}$ or $Y = \{-1, +1\}$

$y$ : specific label $y \in Y$

$X \times Y$ : the space of the training data, i.e. the set of all possible labeled observations, often $X \times Y$ is defined as $\mathbb{R}^m \times \mathbb{R}$ (regression learning) or as $\mathbb{R}^m \times \{-1, +1\}$ (binominal classification learning)

$y_i$ : the label of the $i$-th observation in a (training) data set

$(x_i, y_i)$ : the $i$-th observation and its label

$T$ : training data set $T \subset X \times Y$

$n$ : number of examples in the training data set, i.e. $n = |T|$, also the length of a series (see Series)

$m$ : number of features in the training data set, i.e. the number of random variables of an instance space $X = X_1 \times \ldots \times X_m$

**Probabilities and Distributions**

$P(X)$ : probability distribution for the random variable $X$, for example the distribution for a single feature $X_i$, i.e. $P(X_i)$

$P(x)$ : probability for the event $x$, here: the probability for the observation $x$

$P(y|x)$ : the conditional probability for the event $y$ given event $x$, here: the conditional probability for the label $y$ if the observation $x$ is given

$P(x, y)$ : the joint probability for the event $x$ together with the event $y$, here: the joint probability for the label $y$ together with the observation $x$

**Risk Minimization**

$L(y, y')$ : loss function calculating the error between the actual label $y$ and the prediction $y'$

$f$ : a function $f$, usually the one which should be fitted to the training data

$f_\gamma$ : a function class depending on a parameter (vector) $\gamma$

$\Gamma$ : the space of all possible function parameters $\gamma$

$\Omega(\gamma)$ : capacity function, measures the structural complexity of a function given the functional parameter(s) $\gamma$

$R(\gamma)$ : the expected risk, i.e. the loss which would be expected on the data following a probability distribution $P(x, y)$ given a certain loss function and a function definition $\gamma$

$R_{emp}(\gamma)$ : the empirical risk (training error) which sum up the loss functions on the training data points

$R_{reg}(\gamma)$ : the regularized risk which takes into account both the empirical risk and the structural complexity $\Omega$

$\lambda$ : used to define the general trade-off between the empirical risk and the structural complexity

### Hyperplane Models

$w$ : the normal vector of a linear separating hyperplane, corresponds to feature weights or feature weight functions

$b$ : the offset of a linear separating hyperplane

$\alpha$ : a vector of Lagrange multipliers indicating the importance of a condition in a constrained optimization problem, the components $\alpha_i$ greater than 0 indicate support vectors (examples contributing to the orientation and position of the hyperplane)

$\beta$ : a vector of Lagrange multipliers indicating the importance of a condition in a constrained optimization problem, similar to the vector $\alpha$

$\xi_i$ : the error made for the $i$-th example, often depends on a concrete loss function $L$

$C$ : used to define the trade-off between the empirical risk (training error) and the structural complexity (margin width), corresponds to $\lambda$ in the general setting

$L_p$ : the primal form of large margin optimization problems

$L_d$ : the dual form of large margin optimization problems

$L_p^{(i)}$ : the primal form of the $i$-th objective of large margin optimization problems

$L_d^{(i)}$ : the dual form of the $i$-th objective of large margin optimization problems

### Transduction

$T^*$ : training data set $T^* \subset X$, i.e. a set of unlabeled observations $x_k^* \in X$

$x_k^*$ : unlabeled observation $x_k^* \in X$ from a test set $T^*$

$y_k^*$ : the label variable corresponding to an observation $x_k^*$, part of the optimization process

$n^*$ : number of examples in the test data set, i.e. $n^* = |T^*|$

$\alpha^*$ : a vector of Lagrange multipliers (see above) for the examples from the test set

$\beta^*$ : a vector of Lagrange multipliers (see above) for the examples from the test set

$\xi_k^*$ : the error which would be made for the $k$-th test observation $x_k^*$, depends on the currently selected label $y_k^*$

$C^*$ : used to define the trade-off between the empirical risk (training error), the structural complexity (margin width), and the test error for the current label selection

**Feature Space Transformations**

$X'$ : a transformed feature space, derived from a feature space $X$ by some transformational mapping (e.g. feature selection or construction)

$nf$ : the number of features currently used in the training data, i.e. $1 \leq nf \leq m$, can be used as an optimization criterion

$\Omega(X)$ : feature space capacity function, measures the structural complexity of a feature space $X$, for supervised feature selection, construction, and extraction the function $nf$ can be used for $\Omega(X)$

$R_{reg}^{FST}(X)$ : the regularized risk for feature space transformations

**Series**

$v$ : a value series, i.e. a mapping $v : \mathbb{N} \to \mathbb{R} \times \mathbb{C}^m$

$v_i$ : short form of $v(i)$

$(v_i)_{i \in \{1,\ldots,n\}}$ : a series of length $n$

$n$ : the length of a series, also the number of examples (see Examples)

$\nu$ : frequency, used for example in $e^{i\nu a}$ with $i$ as the imaginary unit

$P_{d,m}$ : phase space representation of a series, the set of phase space vectors with delay $d$ and dimension $m$

$g$ : the overlap of a windowing, defined as $g = w/s$ for window width $w$ and step size $s$

$d$ : parameter of dynamic windowing specifying the amount the window width changes for nested windowings

**Clustering**

$k$ : the (maximum) number of clusters, also used as a general index variable

$C_q$ : the $q$-th cluster, i.e. a set of data points $x_i \in X$

$c_q$ : the centroid of the $q$-th cluster

$\omega_d$ : a cluster quality measure, the inner cluster distance sum, i.e. the sum of all pairwise distances of all data points of all clusters, based on a distance measure $d$

$\omega$ : a cluster quality measure, the optimization function of the $k$-Means clustering scheme, a shortcut for $\omega_d$ is $d$ (the Euclidean distance)

$\omega_{DB}$ : a cluster quality measure, the Davies-Bouldin index which takes into account the relative separation of the two worst separated clusters

$\omega_{GM}$ : a cluster quality measure, the log-likelihood for Gaussian mixture clustering

$\omega_{d-max}$ : a cluster quality measure, the maximum distance in graph-based clustering, based on the distance threshold $d_{max}$ denoting the strength of the weakest link in the clustered graph

$l_p$ : loss in cluster quality caused by a single feature $p$

$\eta_p$ : slope in point $p$ of the Pareto set

$\Delta\eta_p$ : change of slope in point $p$ of the Pareto set

**Feature Transfer**

$T_i$ : a learning task, specified by an instance space $X_i$ with features $X_{ik}$, also denotes the corresponding training data

$\tau$ : set of all learning tasks $T_i$

$X_B$ : set of base features, common for all learning tasks $X_i$

$\mathcal{I}_i$ : the set of irrelevant features for learning task $T_i$

$X_{ik} \sim X_{il}$ : features $X_{ik}$ and $X_{il}$ are alternative

$\mathcal{A}$ : set of features which are either irrelevant or alternative to at least one base feature

$w$ : weighting function, also the normal vector of a separating hyperplane where the coefficients correspond to feature weights

CHAPTER 1

# Introduction

The main topics of this work are optimization and data mining (DM). We can probably assume that readers are familiar with both terms at least to a certain degree. Later in this work, we will discuss some formal definitions for both ideas. Right now, we want to start with a set of alternative definitions for data mining which might lead to some new insights for some readers. These alternative definitions are rather informal and are highly justified by examples or visualizations. They should be seen as a set of theses about some aspects of the nature of data mining. Finally, these theses should also state the connection from data mining to related fields like optimization. It is this connection which will serve as a motivation for this work.

## 1.1. Three Theses about Data Mining

Before we start with the first thesis, we familiarize ourself with the used terms. The objective of *data mining* is to generalize from a given set of data points to a more generic model which either describes these points or can be used to predict properties of new and unseen points – which are similar to the known set at least to a certain degree. Let's assume we have a set of data points like those given in Figure 1.1. If we want to predict the value on the y-axis for any of the locations on the x-axis, we should try to find a functional relation between the y- and the x-axis. For now, we refer to the process of searching the best functional relation as data mining. This informal definition concentrates on models for conditional probability density functions and does of course not cover all relevant and existing aspects of data mining. It is, however, sufficient for the introduction into this work.

It is a well known fact that the search for the optimal function is usually a hard task and a large amount of methods have been proposed for guiding this search. In many

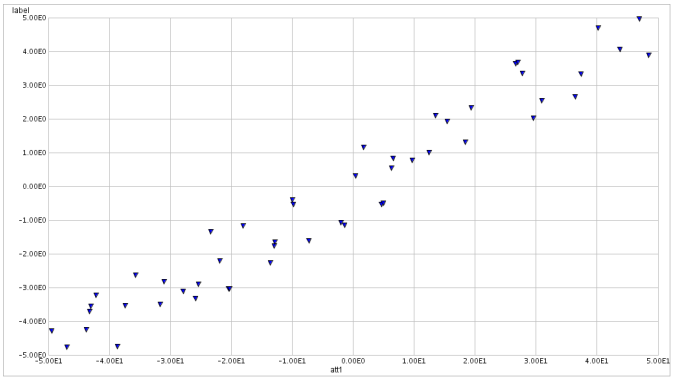



**Figure 1.1.:** A set of data points. The target is to predict the value on the y-axis from the location on the x-axis.

cases, however, a unique optimal solution cannot be found which directly brings us to our first thesis about data mining:

**DM Thesis 1: Data Mining is about Trade-offs.** If we search for a function which predicts the data points in Figure 1.1, we should take the prediction error into account. Hence, we first would like to have a function which does not make any error on the given set of data points. We probably could come up with the following function:

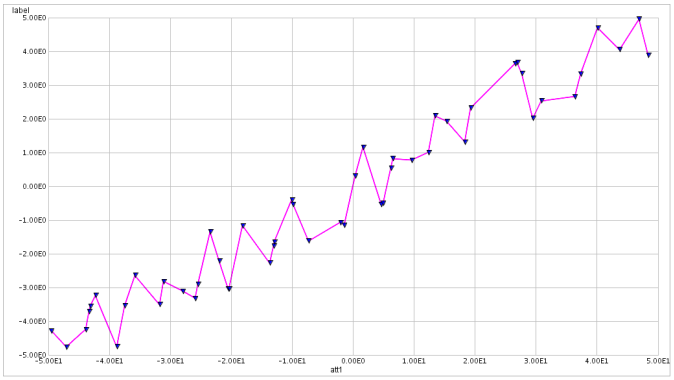



**Figure 1.2.:** A complex model which minimizes the training error: this model just connects all points.

The function (or: *prediction model*) used in Figure 1.2 can be easily found: we have just connected each data point with the next one. The result, however, is not as simple as

this construction description might sound: it is a piecewise linear function which uses one linear equation for each line section. Therefore, we regard this model as rather complicated or *complex* since we need huge efforts to describe this model with mathematical equations.

This model of course has one big advantage: it predicts the given data points perfectly. On the other hand, the complexity of the model reduces the probability that it would perform well on any data point which was not part of the given set of points. Hence, the generalization capacity of this model is probably very low.

If complex models are not desired in data mining, why should we not try and come up with the simplest model we could think of? This leads to the class of linear models like the one in this figure:

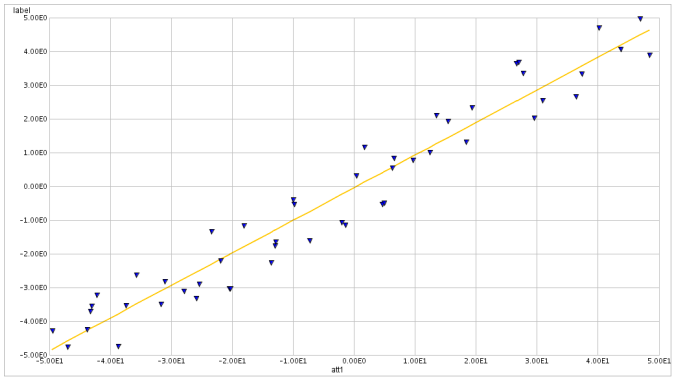



**Figure 1.3.:** The most simple model for this set of data points: a simple linear equation which produces errors even on the given data points.

As you can see, the linear function is as simple as it could be in terms of model complexity. However, we have to pay a price for this simplicity since the model is no longer error-free on the given set of data points. This does not need to be a problem at all: it is not too important that we reduce the error on the given data points but we should reduce the error on the set of *all* data points we possibly could expect. We will discuss this idea later in this work again.

One could argue that now everything is fine: we would probably decide for the simple linear function because we get a feeling that it produces less errors on unseen data points. The other alternative, the complex model stated above, probably does perform worse with respect to the prediction error. So, there *might* be a trade-off between complexity and error here but it does not seem to be too hard to decide which model is the best one, right?

Unfortunately not. We have only regarded two different models – but of course there are a lot more possible functions which might count as a solution for the given data mining problem. The piecewise linear model marks one extreme in the space of possible solutions, the simple linear model marks another one. But how can we be sure that there is not another solution which is almost as simple as the linear function but produces far less errors? We could, for example, consider the following function:

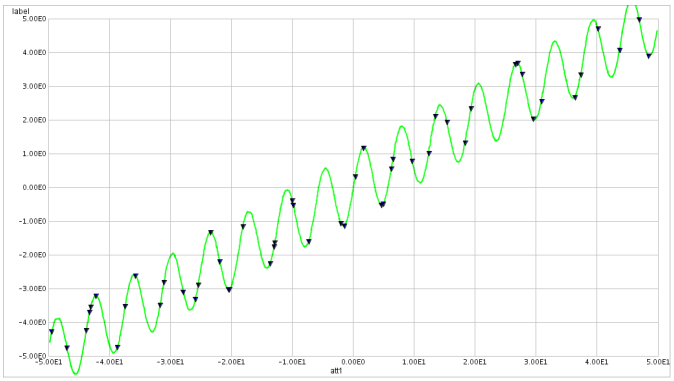



**Figure 1.4.:** A probably perfect trade-off between model complexity and error on the given data points: no errors are made and the model seems to be sufficiently simple.

The function in Figure 1.4 does not make any error for the given set of training points. And it is sufficiently simple. It only consists of a linear function and an additional harmonical component with a fixed frequency. This simplicity gives us a good feeling about the predictive power of the model considering unseen data points.

The fact that we should take two different criteria into account, namely the error and the model complexity, is a well known result from statistical learning theory. Both criteria define a *trade-off* and it has to be decided which model between the extreme solutions is most appropriate for the given set of data points.

In general, this problem is referred to as the problem of *model selection* and remains the main task of data mining which still is not solved in general. We will see in this work that today's data mining methods actually do not make the trade-off decision and it is still the decision of the analyst which model type or parameters he or she prefers. One of the major goals of this work is to make this trade-off explicit and give the analyst more information about this trade-off so that it can be better decided which model is optimal. The search for an optimal solution directly leads to the second thesis:

**DM Thesis 2: Data Mining is about Optimization.** Basically all data mining methods either use optimization or are instances of optimization algorithms. The core of model parameter estimations often is a dynamic programming or expectation maximization algorithm. Probability density functions are modeled through generative models including Bayesian Networks and Markov Random Fields which directly employs optimization techniques. And this is especially true for the model building process for conditional probability density functions which will be mainly discussed here, including methods like Artificial Neural Networks, Decision Trees, Gaussian Processes, Linear Discriminant Analysis, Perceptrons, Support Vector Machines, or Boosting and other meta learning schemes.

We will see later in this work that the most successful learning methods known today try to optimize an objective function like

$$\int L(y, f(x, \gamma))dP(x, y).$$

based on a loss function $L$ and a probability distribution $P(x, y)$ for the data points $(x, y) \in X \times Y$. The loss function directly corresponds to the errors made by the learned function $f$ with the function parameters $\gamma$. The probability distribution is usually not known and we will not be able to calculate this objective function directly. We therefore concentrate on the errors made on the data points like we did before and minimize the function

$$\frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f\left(x_i, \gamma\right)\right).$$

We have now seen that data mining is about optimization with respect to the error made on a given set of points. But our first thesis states that we should also take the structural complexity into account like it is suggested by the ideas underlying the statistical learning theory. And here is the point where classical statistical learning and this work start to differ:

**DM Thesis 3: Data Mining is about Multi-Objective Optimization.** Let's assume that we manage to measure the structural complexity of a model and let's say we measure the complexity by $\Omega(\gamma)$. We then have two different criteria which both should be optimized:

$$\begin{aligned} &\text{minimize } \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f\left(x_i, \gamma\right)\right) \text{ and} \\ &\text{minimize } \Omega(\gamma). \end{aligned}$$

Since the error (first equation) is usually a monotonically decreasing function of $\Omega$ (second equation), both criteria usually compete. This is the trade-off we discussed in the

first thesis. Classical statistical learning now introduces a weighting factor $\lambda$ to manage the trade-off between training error and capacity:

$$\frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f\left(x_i, \gamma\right)\right) + \lambda\Omega(\gamma) \qquad \text{(classical statistical learning)}.$$

The analyst now has to define a weighting factor $\lambda$ defining the trade-off for the given analysis task *before* the analysis has even started. This causes a lot of problems because usually the analyst has no idea which weighting factor $\lambda$ is optimal for a new learning task.

Hence, it would be great if the analyst would no longer has to define the trade-off factor beforehand but could get all solutions at once, i.e. with an algorithm having the same running time in Landau notation, and choose the best solution afterwards. All solutions would be ordered according to their performance with respect to both criteria and no useless solutions would be presented (those solutions which are worse with respect to one criterion without being better with respect to the other compared to all other solutions). For readers who are familiar with the concept: we would like the complete *Pareto front* of solutions with respect to both criteria, namely error and model complexity.

The result would no longer be a single solution but a set of solutions (called *Pareto set*), each one is optimal with respect to a specific trade-off between both criteria. Since the analyst should now have more information and insights into the data, he or she can easily select a final solution from this Pareto set if necessary. Although it does not seem to be likely to get so much more information about the data mining problem at hand in the same computation time, this actually *is* possible and will be one of the main topics of this work.

It is interesting to notice that a unique optimal trade-off like the one discussed for the example in the first thesis does not need to exist[1]. If only a unique optimum exists, the solutions of the resulting Pareto set would collapse into one single solution. We will see that this never happens in practice and therefore the complete Pareto set should always be the desired solution on which the analyst can base decisions.

## 1.2. Related Work

The publications discussed in this section are related to this work on a very high level of abstraction. Each chapter also contains citations which are in particular interesting for the chapter itself.

[1] Please be aware that unique here does not refer to the uniqueness introduced by statistical learning, for example by using the large margin heuristic for model selection.

This thesis discusses options to overcome the basic problem of data mining, namely the problem of model selection. Model selection is considered as a hard task [63] which probably cannot be solved in general. However, statistical learning theory has defined several criteria which could be taken into account for guiding the search for appropriate models. Introductions into statistical learning theory can be found in [184, 185] although the idea of shrinkage estimators is much older and was already discussed in [70]. The used loss functions can be found in [5, 63, 167, 199]. The consequences of using these shrinkage estimators optimizing a regularized risk taking into account the error and the model complexity are discussed in [31, 59, 83]. One of the most interesting consequences is the representer theorem stating the fact that all functions minimizing the regularized risk can be expressed in terms of a basis function applied on the given training data points. Finally, a generic discussion of data mining methods including the most important shrinkage estimators is given in [190].

The second major field of this work is that of optimization. A lot of publications exist in this field introducing the basic notation and the most important algorithms [51, 137]. However, most problems in this work lead to either non-convex optimization problems or multi-objective optimization problems (or both) [142, 202].

Today, evolutionary algorithms are the standard solution for these types of problems [6]. They work simultaneously on a set of solution candidates and can hence find the complete Pareto set in a single run [36, 66, 127]. Traditional approaches known from mathematical programming must be restarted several times with different trade-off factors to get the same result [46, 198]. Furthermore, evolutionary algorithms do not depend on the form or the continuity of the Pareto front [27, 99].

For the combination of data mining and optimization, we first focus on Support Vector Machines (SVM) as they are the most prominent representatives for conditional probability density estimators. In some sense, SVMs guarantee an optimal solution for the given data set, i.e. they define which hyperplane has to be chosen. But the important drawback here is that the "model selection" is performed *after* the user has defined the trade-off factor between error and model complexity (width of the hyperplane margin). An advantage of SVMs is, however, that many other optimization problems can also be formulated as large margin problems [177]. We will discuss cases in this work where the quadratic programming approaches usually employed by large margin methods is not able to find satisfying solutions at all. In addition to the multi-objective setting described above, this is always the case if the underlying optimization problem is no longer convex. In these cases, most SVM implementations do not even terminate [61]. There exist, for example, several useful kernel function for SVMs [96], among them the sigmoid kernel which simulates a neural network [24, 168], which will lead to such non-convex optimization problems. We will discuss more appropriate optimization techniques which will be able to solve both problems, the non-convex optimization problem and the multi-objective problem setting.

Later in this work we will discuss the transfer from multi-objective learning to multi-objective preprocessing and will mainly concentrate on feature space transformations like feature selection, feature construction, and feature extraction. We will see that this will always lead to non-convex optimization problems and we will also discuss possibilities to improve feature space transformations by multi-objective optimization.

Although many of the known data mining methods try to detect or construct relevant features, both theory and experiments have shown that data mining methods usually scale worse with an increasing number of irrelevant or redundant features [50, 94]. Decision tree learners like C4.5 [146] or CART [21] as well as instance-based learners [2] deliver bad results if the data set contains superfluous features. Other algorithms like Naive Bayes [41] or Support Vector Machines [72, 169, 185] respond with a decreasing prediction performance on correlated features, even if those are not irrelevant.

Hence, additional methods for the automatic selection of relevant features or the construction of new ones are necessary. There actually is no clear border between preprocessing and learning. An improved feature space and hence changed representation of the input data might ease the learning of hypotheses [193, 194]. Finally, the form of the input representation could not only influence the predictive power but also the understandability of models and the computation time [47].

It is quite interesting to notice that the success story of kernel methods [164] is also connected to feature space transformations. For certain transformations into a feature space, we could use an efficient kernel function instead (kernel trick) [163]. Kernel functions are the base for several analysis methods including Support Vector Machines or Kernel Principal Component Analysis (Kernel PCA) [126]. It is also interesting to note that the idea of kernel functions is quite old [3, 19, 102, 153] but achieved eminence just recently. The development of new kernel functions still is a contemporary issue [23, 52, 129, 165].

The construction of new features is in some sense similar to the calculation of kernel functions: both approaches lead to a new feature space, usually with a higher number of dimensions. Alternatives to kernel functions are the construction of new features from already known ones [139, 172, 191] and the extraction of features on a series of others [20, 162]. Furthermore, large amounts of domain specific features exist, e.g. the feature extraction from image or video data [7, 42, 57, 156] or from audio data [9, 53, 60, 151, 181, 189].

In contrast to feature construction, the mere selection of features does not produce additional features. The goal is to select a subset containing less irrelevant features. This reduced subset should support the learning method by not hiding the underlying patterns by noise introduced by irrelevant features [17, 33, 76, 89]. Alternatively, one could search for an optimal feature weighting. Feature selection actually is just a special form of feature weighting with binary weights.

Two basic approaches exist for the problem of feature selection: filtering the features according to a weighting calculation based on the data alone [4, 84, 95, 188] or taking the performance of the data mining method into account. The latter solution is called wrapper approach [76, 88] and will usually be employed in this work.

Feature selection and feature generation in some sense compete with each other: the selection tries to reduce the number of features in order to improve the predictive power of a learning scheme and the generation tries to construct new features – and hence enlarge the feature space – with the same goal of error minimization. Hybrid solutions try to weigh both effects [15, 16, 155, 183]. In this work, we will see that the definition of an appropriate feature space complexity measure together with multi-objective optimization will naturally lead to a solution for this problem. This solution turns out to work for supervised – and for a first time – also for unsupervised learning problems.

Further related work will be cited in each chapter and discussed with respect to the possibilities and problems connected to those publications.

## 1.3. Outline

After discussing some basic definitions in Chapter 2, this work is divided into two parts. The first part deals with machine learning methods and how they can be extended in a way that the results are no longer a single solution for a predefined trade-off factor but the complete set of solutions for all possible trade-offs (Chapter 3). As a by-product, we could use these new learning schemes and solve non-convex optimization problems which could hardly be solved before. This will be discussed in Chapter 4. Chapter 5 concludes the first part about learning methods and shows how non-convex optimization can be combined with multi-objective optimization in order to solve a practically very important problem, namely transductive learning.

In the second part, we transfer the idea of structural risk minimization into the preprocessing for learning methods and concentrate on multi-objective optimization for feature space transformation. In Chapter 6, we introduce the notion of feature space complexities and discuss a simple measure leading to a Pareto set as the result for supervised feature construction tasks. We extend these ideas to multi-objective feature extraction from series data in Chapter 7. One of the main results of this work is the possibility to apply feature selection and feature construction also for unsupervised learning methods. This is possible due to multi-objective optimization and a paradigm change with respect to the optimization direction of one of the criteria and will be discussed in detail in Chapters 8 (unsupervised feature selection) and 9 (unsupervised feature aggregation). Since optimizations of the feature space are usually very demanding tasks with respect to the computation time, the final Chapter 10 introduces an approach for re-using the insights from former optimization runs by means of a case-based reasoning approach.

CHAPTER 2

# Basics

This chapter discusses some basic definitions and notations which are useful in the following chapters. The first section defines the learning tasks mainly considered in this work, namely supervised and unsupervised learning. Section 2.2 introduces one of the currently most popular and successful ways of transforming supervised learning into optimization problems: statistical learning theory. We will discuss the optimization problem of statistical learning, namely the minimization of the structural risk. We will then discuss the most prominent learning method for structural risk minimization called Support Vector Machines. After a short overview over traditional optimization techniques in Section 2.3, we will introduce the idea of multi-objective optimization in Section 2.4.

## 2.1. Machine Learning

The goal of machine learning is to find models for given data sets. These models should describe the data sets, i.e. in the best case they lead to deeper insights into the data generating processes, and they should be operational in a sense that the models can be applied to new and possibly unseen data points. The latter is especially important if the models are used in predictive settings, i.e. if the models should describe the dependency of a certain property of the data points on other properties.

In this thesis, we will concentrate on data given in propositional form. This means that the data is given in rows of one table (attribute-value format). In the case of relational data sets, we assume that the data is transformed into one single table by appropriate preprocessing operations. We will now formalize the space of those data rows:

**Definition 2.1 (Instance Space)** *The* INSTANCE SPACE *is composed of a set of random variables* $X = X_1 \times \ldots \times X_m$.

**Definition 2.2 (Attributes)** *The random variables* $X_k$ *building the instance space are called* ATTRIBUTES *or* FEATURES.

The attributes correspond to the column headers of our data table. We can now define a single row of the table:

**Definition 2.3 (Instance / Observation)** *An element* $x \in X$ *is called* INSTANCE *or* OBSERVATION.

Hence, we can also denote the set of all possible observations $x \in X$ with the set variable $X$. We say that $x$ is represented by a vector of attribute-value pairs (attribute-value representation), since $x$ is a vector where the component $x_k$ refers to the attribute $X_k$. We denote the $i$-th observation in an instance set $X$ by $x_i$ and the $k$-th value of the $i$-th instance by $x_{ik}$.

In the following, we will state some of the most prominent learning tasks for data given in attribute-value representations. All learning tasks discussed here are relevant for this thesis. Several more very interesting learning tasks exist, for example subgroup discovery or association rule mining, which are not covered by this thesis and are hence not mentioned here.

### 2.1.1. Supervised Learning

In supervised learning, we define a special attribute in our instance space and use this attribute in order to describe the complete row. We denote the set of all possible values of this specific attribute with $Y$ and we use $y \in Y$ in order to specify a specific label value.

The Cartesian product $X \times Y$ then denotes the set of all possible labeled observations. We can now formalize the goal of supervised learning:

**Definition 2.4 (Supervised Learning)** SUPERVISED LEARNING *aims at finding a function* $f : X \to Y$ *deriving the label value* $y \in Y$ *from a given observation* $x \in X$.

In supervised learning settings, the data often is presented as *training data* $T \subset X \times Y$, hence as a set of pairs $(x_i, y_i)$. Here, $x_i$ denotes the $i$-th observation in the training data set $T$ and $y_i$ the corresponding label. We also refer to these training instances as *examples* or as *example set*. The number of examples in the training data set is $n = |T|$. The number of features in the corresponding instance space, i.e. the number of the underlying random variables of the space $X = X_1 \times \ldots \times X_m$, is denoted by $m$.

#### 2.1.1.1. Classification Learning

We can distinguish two basic learning tasks for supervised learning depending on the fact if $Y$ is continuous or discrete. The latter case means that $Y$ is composed of a finite set of discrete label values which are possible for each $y_i$. Since each row in the data set can only have one value out of $Y$, the data set is partitioned by the values $Y$ and is divided in distinct groups or classes. Hence, the name *classification learning* is the most widely used term for supervised learning with a discrete label space $Y$. If $Y$ is constrained to $\{-1, +1\}$, the classification task is called binary classification or binominal classification learning. A classification problem with $|Y| > 2$ can be transformed into a set of binominal classification problems by techniques like 1-vs-1, 1-vs-all, or by error correcting output codes. Hence, we will usually concentrate on binominal classification problems in this thesis instead of polynominal ones. A positive side effect of this restriction is that the definition of statistical learning problems is usually much easier for binominal classifications.

#### 2.1.1.2. Regression Learning

We now consider the case that the label space $Y$ is not discrete but continuous, for example $Y = \mathbb{R}$. In contrast to supervised classification learning, no predefined classes exist but the task now is to assign a numerical value to new and unseen rows instead of classifying unseen data points into the predefined classes. This learning task is called *regression learning* and it is actually a more general case of the classification setting. In both cases, the goal is to find a function $f$ which predicts the correct value for a given observation. In order to achieve this, we define a loss function which penalizes errors in predictions and which must be minimized during the learning process. We will discuss this idea of minimizing the prediction errors or loss later in Section 2.2.

### 2.1.2. Unsupervised Learning

In some cases, there is no specific label attribute $Y$ and hence it is not possible to learn a function $f$ which can be used to predict the value of $Y$ for unseen data points $x \in X$. But still one can try to identify patterns in the data in order to describe underlying processes. If these patterns correspond to subsets of the data sharing similar properties, for example which are located close together in the instance space, the learning task is called *clustering*. The aim of cluster analysis is to group data points into sets of similar data points. Let $X$ be a set of individual unlabeled data points $x_i$ (observations). A cluster is a subset of data points $C_q \subseteq X$. In principle, clusters may overlap. However, most clustering algorithms are designed to produce partitions of data points, i. e. a set

of clusters $C_1 \dots C_k$ such that $C_i \cap C_j \neq \emptyset \Rightarrow C_i = C_j$ (clusters do not overlap) and $\bigcup_{q=1}^{k} C_q = X$ (each data point is covered by a cluster).

In contrast to classification, the calculation of a quality measure for a function $f$ assigning a cluster membership to each (new) data point is much harder. Since there is no ground truth, it is not possible to define a loss function like in the classification setting which could simply be minimized. We will discuss some of the proposals for cluster quality measurements in later chapters of this thesis.

## 2.2. Statistical Learning

In this section, we first concentrate on the problem of supervised learning, namely binominal classifications. We will discuss the idea of regularized risk minimization. Machine learning methods following this paradigm have a solid theoretical foundation and it is possible to define bounds for prediction errors.

### 2.2.1. Regularized Risk Minimization

Let $X \in \mathbb{R}^m$ be a real-valued vector of random variables. Let $Y \in \mathbb{R}$ be another random variable. $X$ and $Y$ obey a fixed but unknown probability distribution $P(X, Y)$. As we have seen before, supervised machine learning tries to find a function $f(x, \gamma)$ which predicts the value of $Y$ for a given input $x \in X$. The function class $f$ depends on a vector of parameters $\gamma$. We define a *loss function* $L(Y, f(X, \gamma))$ in order to penalize errors during prediction [63]:

**Definition 2.5 (Loss Function)** *A convex function $L$ with arity 2, positive range, and $L(x, x) = 0$ is called* LOSS FUNCTION.

The *arity* of a function is defined as the number of arguments.

The definition of a loss function leads to a possible criterion for the selection of a function $f$, the *expected risk*:

**Definition 2.6 (Expected Risk)** *Let $X \in \mathbb{R}^m$ be a real-valued vector of random variables, $Y \in \mathbb{R}$ be another random variable, and let $X$ and $Y$ obey a fixed but unknown probability distribution $P(X, Y)$. Let $L$ be a loss function. The* EXPECTED RISK *is defined as*

$$R(\gamma) = \int L(y, f(x, \gamma)) dP(x, y).$$

Since the underlying distribution is not known we are not able to calculate the expected risk. However, instead of estimating the probability distribution in order to allow this calculation, we directly estimate the expected risk by using a set of data points $T = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq X \times Y$. This set $T$ of known data points is usually called *training data*. Using this set of data points we can calculate the *empirical risk*:

**Definition 2.7 (Empirical Risk)** *Let $T = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq X \times Y$ be a set of data points and $L$ be a loss function. The* EMPIRICAL RISK *is defined as*

$$R_{emp}(\gamma) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f\left(x_i, \gamma\right)\right).$$

If training data is sampled according to $P(X, Y)$, the empirical risk approximates the expected risk if the number of samples grows:

$$\lim_{n \to \infty} R_{emp}(\gamma) = R(\gamma).$$

It is, however, a well known problem that for a finite number of samples the minimization of $R_{emp}(\gamma)$ alone does not lead to a good prediction model [185]. For each loss function $L$, each candidate $\gamma$, and each set of tuples $T' \subseteq X \times Y$ with $T \cap T' = \emptyset$ exists another parameter vector $\gamma'$ so that $L(y, f(x, \gamma)) = L(y, f(x, \gamma'))$ for all $x \in T$ and $L(y, f(x, \gamma)) > L(y, f(x, \gamma'))$ for all $x \in T'$. Therefore, the minimization of $R_{emp}(\gamma)$ alone does not guarantee the optimal selection of a parameter vector $\gamma$ for other samples according to the distribution $P(X, Y)$. This problem is often referred to as *overfitting*.

At this point we use one of the main ideas of statistical learning theory. Think of two different functions perfectly approximating a given set of training points. The first function is a linear function, i. e. a simple hyperplane in the considered space $\mathbb{R}^m$. The second function also hits all training points but is strongly wriggling in between. Naturally, if we had to choose between these two approximation functions, we tend to select the more simple one, i. e. the linear hyperplane in this example. This derives from the observation that more simple functions behave better on unseen examples than very complicated functions. Since the mere minimization of the empirical risk according to the training data is not appropriate to find a good generalization, we incorporate the *capacity*[1] of the used function into the optimization problem (see Figure 2.1). This leads to the minimization of the *regularized risk*:

**Definition 2.8 (Regularized Risk)** *Let $R_{emp}$ be the empirical risk and let $\Omega$ be a strictly monotonic increasing function. The* REGULARIZED RISK *is defined as*

$$R_{reg}(\gamma) = R_{emp}(\gamma) + \lambda \Omega(\gamma).$$

[1] Although not the same, the capacity of a function resembles a measurement of the function complexity. In our example we measure the ability to "wriggle". More details can be found in [185].

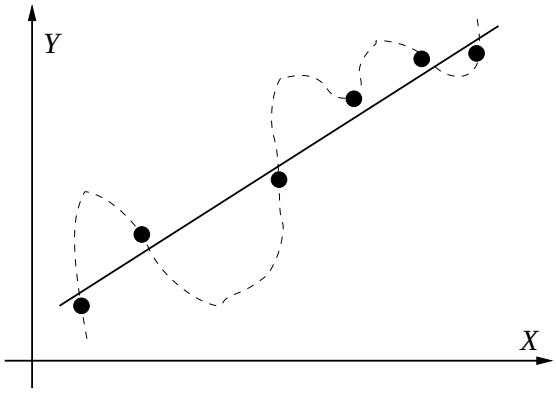



**Figure 2.1.:** The simultaneous minimization of empirical risk and model complexity gives a hint which function should be used in order to generalize the given data points.

This risk functional is also known as *structural risk* since it takes the structural complexity into account. $\Omega$ is a function which measures the capacity of the function class $f$ depending on the parameter vector $\gamma$. Since the empirical risk is usually a monotonically decreasing function of $\Omega$, we use $\lambda$ to manage the trade-off between training error and capacity. Methods minimizing this type of risk function are known as *shrinkage estimators* [70].

#### 2.2.1.1. Bound on the Generalization Performance

For certain functions $\Omega$, the regularized risk is an upper bound for the empirical risk. The capacity of the function $f$ for a given $\gamma$ can for example be measured with help of the *Vapnik-Chervonenkis dimension* (VC dimension) [185, 186]. The VC dimension is defined as the cardinality of the biggest set of tuples which can be separated with help of $f$ in all possible ways. For example, the VC dimension of linear hyperplanes in an $m$-dimensional space is $m + 1$. Using the VC dimension as a measure for capacity leads to a probabilistic bound for the regularized risk [185]. Let $f$ be a function class with finite VC dimension $h$ and $f(\gamma)$ the best solution for the empirical risk minimization for $T$ with $|T| = n$. Now choose some $\eta$ such that $0 \leq \eta \leq 1$. Then for losses smaller than some number $B$, the following bound holds with probability $1 - \eta$:

$$R(\gamma) \leq R_{emp}(\gamma) + B\sqrt{\frac{h\left(\log\frac{2l}{h} + 1\right) - \log\frac{\eta}{4}}{n}}.$$

Surprisingly, this bound is independent of $P(X, Y)$. It only assumes that both the seen and the unseen data points are independently sampled according to some $P(X, Y)$. The

existence of a guaranteed error bound is the reason for the great success of regularized risk minimization in a wide range of applications. For the function classes used in this work, the VC dimension is always known. However, in general the calculation of the VC dimension was possible for few different function classes only [141].

## 2.2.2. Large Margin Methods

As discussed in the previous section, we need to use a class of functions whose capacity can be controlled. In this section, we will discuss a special form of regularized risk minimization, namely large margin approaches. All large margin methods have one thing in common: they embed regularized risk minimization by maximizing a margin between a linear function and the nearest data points. The most prominent large margin method for classification tasks is the *Support Vector Machine* (SVM).

### 2.2.2.1. Support Vector Machines

We constrain the number of possible values of $Y$ to 2, without loss of generality these values should be $-1$ and $+1$. In this case, finding a function $f$ in order to decide which of both predictions is correct for an unseen data point is referred to as *classification learning* for the classes $-1$ and $+1$. We start with the simplest case: learning a linear function from perfectly separable data. As we shall see in the next paragraphs, the general case - non-linear functions derived from non-separable data - leads to a very similar problem.

If the data points are linearly separable, a linear hyperplane must exist in the input space $\mathbb{R}^m$ which separates both classes. This hyperplane is defined as

$$H = \{h | \langle w, h \rangle + b = 0\},$$

where $w$ is normal to the hyperplane, $|b|/||w||$ is the perpendicular distance of the hyperplane to the origin (offset or bias), and $||w||$ is the Euclidean norm of $w$. The vector $w$ and the offset $b$ define the position and orientation of the hyperplane in the input space. These parameters correspond to the function parameters $\gamma$. After the optimal parameters $w$ and $b$ were found, the prediction for new data points $x$ can be calculated as

$$f(x, w, b) = \operatorname{sgn}(\langle w, x \rangle + b),$$

which is one of the reasons why we constrained the classes to $-1$ and $+1$.

Figure 2.2 shows some data points and a separating hyperplane. If all given data points are correctly classified by the hyperplane at hand the following must hold:

$$\forall_{i=1}^{n} : y_i (\langle w, x_i \rangle + b) \geq 0. \tag{2.1}$$

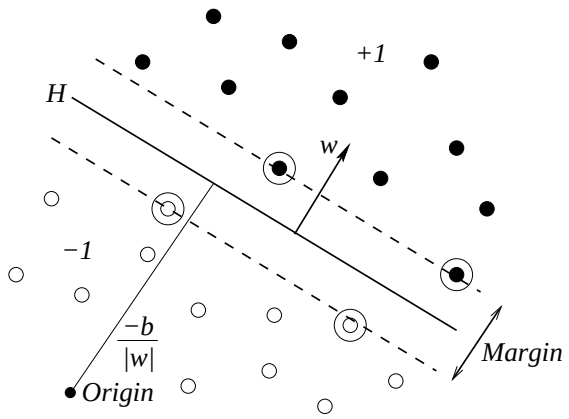



**Figure 2.2.:** A simple binary classification problem for two classes $-1$ (empty bullets) and $+1$ (filled bullets). The separating hyperplane is defined by the vector $w$ and the offset $b$. The distance between the nearest data point(s) and the hyperplane is called *margin*.

Of course, an infinite number of different hyperplanes exist which perfectly separate the given data points. However, one would intuitively choose the hyperplane which has the biggest amount of safety margin to both sides of the data points. Normalizing $w$ and $b$ in a way that the point(s) closest to the hyperplane satisfy $|\langle w, x_i\rangle + b| = 1$ we can transform equation 2.1 into

$$\forall_{i=1}^{n} : y_i \left(\langle w, x_i\rangle + b\right) \geq 1.$$

We can now define the *margin* as the perpendicular distance of the nearest point(s) to the hyperplane. Consider two points $x_1$ and $x_2$ on opposite sides of the margin. That is $\langle w, x_1\rangle + b = +1$ and $\langle w, x_2\rangle + b = -1$ and $\langle w, (x_1 - x_2)\rangle = 2$. The margin is then given by $1/||w||$.

It can be shown, that the capacity of the class of separating hyperplanes decreases with increasing margin [164]. The reason for this behavior is rather simple: with a large margin, there is only a small number of possibilities to separate the data, i.e. the VC dimension of the hyperplane is small. On the contrary, if we allow smaller margins there are more separating hyperplanes for the given data set, i.e. the VC dimension is large in comparison.

Maximizing the margin of a hyperplane therefore formalizes the regularized risk minimization discussed in the previous section. Instead of maximizing $1/||w||$ we could also minimize $\frac{1}{2}||w||^2$ which will result into more simple equations later. This leads to the following optimization problem:

**Problem 2.1 (Primal SVM Problem (separable))** *The primal SVM optimization problem for separable data is defined as*

$$\textit{minimize } \frac{1}{2}||w||^2 \tag{2.2}$$

$$\textit{subject to } \forall_{i=1}^{n} : y_i \left( \langle w, x_i \rangle + b \right) \geq 1. \tag{2.3}$$

Function 2.2 is the *objective function* and the constraints from equation 2.3 are called *inequality constraints*. They form a *constrained optimization problem*. We will use a Lagrangian formulation of the problem. This allows us to replace the inequality constraints by constraints on the Lagrange multipliers which are easier to handle. The second reason is that after the transformation of the optimization problem, the training data will only appear in dot products. This will allow us to generalize the optimization to the non-linear case (see Section 2.2.2.3). We will now introduce positive Lagrange multipliers $\alpha_i, i = 1, \ldots, n$, one for each of the inequality constraints. The Lagrangian has the form

$$L_P(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \left( y_i \left( \langle w, x_i \rangle + b \right) - 1 \right). \tag{2.4}$$

Finding a minimum of this function requires that the derivatives

$$\frac{\partial L_P(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i \quad \text{and}$$

$$\frac{\partial L_P(w, b, \alpha)}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i$$

are zero, i. e.

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad \text{and} \tag{2.5}$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i. \tag{2.6}$$

The Wolfe dual, which has to be maximized, results from the Lagrangian by substituting 2.5 and 2.6 into 2.4, thus

$$L_D(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle .$$

This leads to the dual optimization problem which must be solved in order to find a separating maximum margin hyperplane for given set of data points:

**Problem 2.2 (Dual SVM Problem (separable))** *The dual optimization problem for Support Vector Machines on linearly separable data is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ &\textit{subject to } \forall_{i=1}^{n} : \alpha_i \geq 0 \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

From an optimal vector $\alpha$ we can calculate the optimal normal vector $w$ using equation 2.5. The optimal offset can be calculated with help of equation 2.3. Please note, that $w$ is a linear combination of those data points $x_i$ with $\alpha_i \neq 0$. These data points are called *support vectors*, hence the name support vector machine. Only support vectors determine the position and orientation of the separating hyperplane, other data points might as well be omitted during learning. In Figure 2.2, the support vectors are marked with circles. The number of support vectors is usually much smaller than the total number of data points.

#### 2.2.2.2. Non-Separable Data

We now consider the case that the given set of data points is not linearly separable. The optimization problem discussed in the previous section would not have a solution since in this case constraint 2.3 could not be fulfilled for all $i$. We relax this constraint by introducing positive slack variables $\xi_i, i = 1, \ldots, n$. Constraint 2.3 becomes

$$\forall_{i=1}^{n} : y_i \left( \langle w, x_i \rangle + b \right) \geq 1 - \xi_i.$$

In order to minimize the number of wrong classifications we introduce a correction term $C \sum_{i=1}^{n} \xi_i$ into the objective function. The optimization problems then becomes

**Problem 2.3 (Primal SVM Problem (non-separable))** *The primal SVM optimization problem for non-separable data is defined as*

$$\textit{minimize } \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \xi_i \tag{2.7}$$

$$\textit{subject to } \forall_{i=1}^{n} : y_i \left( \langle w, x_i \rangle + b \right) \geq 1 - \xi_i. \tag{2.8}$$

Now we can for a first time see the correspondence between the regularized risk defined above and the optimization problem of SVM. The term $\frac{1}{2}||w||^2$ corresponds to the term $\Omega(\gamma)$ in the definition of the regularized risk with the hyperplane orientation as function

definition $\gamma$. The sum $\sum_{i=1}^{n} \xi_i$ reflects the empirical risk, i.e. the sum of training errors in terms of a certain loss function (the so-called Hinge loss, see Section 3.2.1).

The factor $C$ determines the weight of wrong predictions as part of the objective function. It corresponds to the weighting factor $\lambda$ in the general definition of the regularized risk. As in the previous section, we create the dual form of the Lagrangian. The slacking variables $\xi_i$ vanish and we get the optimization problem:

**Problem 2.4 (Dual SVM Problem (non-separable))** *The dual optimization problem for Support Vector Machines on linearly non-separable data is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle \\ &\textit{subject to } \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

It can easily be seen that the only difference to the separable case is the additional upper bound $C$ for all $\alpha_i$.

#### 2.2.2.3. Non-Linear Learning with Kernels

The optimization problem 2.4 will deliver a linearly separating hyperplane for arbitrary datasets. The result is optimal in a sense that no other linear function is expected to provide a better classification function on unseen data according to $P(X, Y)$. However, if the data is not linearly separable at all, the question arises how the described optimization problem can be generalized to non-linear decision functions. Please note that the data points only appear in the form of dot products $\langle x_i, x_j \rangle$. A possible interpretation of this dot product is the similarity of these data points in the input space $\mathbb{R}^m$. Now consider a mapping $\Phi : \mathbb{R}^m \rightarrow H$ into some other Euclidean space $H$ (called *feature space*) which might be performed before the dot product is calculated. The optimization would depend on dot products in this new space $H$, i. e. on functions of the form $\langle \Phi(x_i), \Phi(x_j) \rangle$. We can formalize this type of function:

**Definition 2.9 (Kernel Function)** *A function $k : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ with the characteristic*

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

*is called* KERNEL FUNCTION *or* KERNEL.

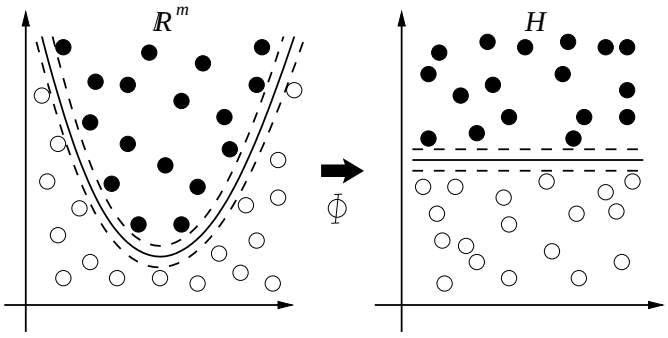



**Figure 2.3.:** After the transformation of all data points into the feature space $H$ the non-linear separation problem can be solved with a linear separation algorithm. In this case, a transformation in the space of polynomials with degree 2 was chosen.

Figure 2.3 gives a rough idea how transforming the data points can help to solve non-linear problems with the optimization in a (higher dimensional) space where the points can be linearly separated.

A fascinating property of kernels is that for some mappings $\Phi$ a kernel $k$ exists which can be calculated without actually performing $\Phi$. Since often the dimension of $H$ is greater than the dimension $m$ of the input space and $H$ sometimes is even infinite dimensional, the usage of such kernels is a very efficient way to introduce non-linear decision functions into large margin approaches. The following definitions show some prominent examples for such efficient non-linear kernels.

**Definition 2.10 (Polynomial Kernel)** *A* POLYNOMIAL KERNEL *with degree $d$ is defined as*

$$k(x_i, x_j) = (\kappa\langle x_i, x_j\rangle + \delta)^d .$$

**Definition 2.11 (RBF Kernel)** *A* RADIAL BASIS FUNCTION KERNEL (RBF KERNEL) *is defined as*

$$k(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$$

*for a $\sigma > 0$.*

**Definition 2.12 (Sigmoid Kernel)** *A* SIGMOID KERNEL *is defined as*

$$k(x_i, x_j) = \tanh(\kappa\langle x_i, x_j\rangle - \delta) .$$

The sigmoid kernel can be used to simulate a neural network. $\kappa$ and $\delta$ are scaling and shifting parameters.

Since the RBF kernel is easy interpretable and often yields good prediction performance, it is used in a wide range of applications. We will also use the RBF kernel for our experiments described in section 3.1.3 in order to demonstrate the learning ability of the proposed SVM.

We replace the dot product in the objective function by kernel functions and achieve the final optimization problem for finding a separation for non-linearly separable data points:

**Problem 2.5 (Dual SVM Problem (final))** *The dual optimization problem for non-linear Support Vector Machines on non-linearly separable data is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \\ &\textit{subject to } \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

In general, the primal optimization problem without kernel function is a convex quadratic programming problem, since the objective function is itself convex, and those points which satisfy the constraints also form a convex set. It holds that any linear constraint defines a convex set, and a set of $p$ simultaneous linear constraints defines the intersection of $p$ convex sets. This again is a convex set. The dual problem, which has to be maximized, hence can be considered to be a concave optimization problem.

It can be shown that if the kernel $k$, i. e. it's kernel matrix, is positive definite, this still holds and that the objective function is still concave [22]. The optimization problem therefore has a global unique maximum. However, in some cases a specialized kernel function must be used to measure the similarity between data points which is not positive definite, sometimes not even positive semidefinite [164]. In these cases the usual quadratic programming approaches might not be able to find a global maximum in feasible time. We will discuss possibilities to overcome this issue in Chapter 4.

## 2.3. Optimization

In the last section, we have discussed the optimization problem which must be solved in order to find the optimal hyperplane. In general, we can define optimization as the search for an optimal valid solution with respect to a given objective function. *Optimal* means that the objective function should be minimized or maximized. *Valid* means that the function arguments must not be located in forbidden ranges of the parameter space.

The core of almost every learning problem or automatic preprocessing problem for machine learning is actually an optimization problem. Therefore, we now formalize the informal definition stated above:

**Definition 2.13 (Optimization Problem)** *Let $\Gamma$ be a non-empty set and let $r$ be a function $r : \Gamma \to \mathbb{R}$ called* OBJECTIVE FUNCTION. *We search an element $\hat{\gamma} \in \Gamma$ so that either*

$$\forall \gamma \in \Gamma : r(\hat{\gamma}) \leq r(\gamma) \quad \textit{(Minimization)}$$

*or so that*

$$\forall \gamma \in \Gamma : r(\hat{\gamma}) \geq r(\gamma) \quad \textit{(Maximization).}$$

**Example 2.1 (Optimization for Statistical Learning)** *The objective function $r$ could be defined as the structural risk (see Section 2.2.1) and the parameter space $\Gamma$ could then be defined as the set of all possible prediction function parameters $\gamma$.*

Typically, $\Gamma$ is a subset of $\mathbb{R}^p$. This subset is defined by a set of *constraints*, i.e. equality and inequality constraints on the elements $\gamma$. The elements $\gamma \in \Gamma$ are called *valid* solutions. A valid solution which maximizes (minimizes) the objective function is called *optimal solution*. We differ between different types of optimization problems by distinguishing different types of objective functions and constraints[2]:

**Linear Programming (LP):** $r$ is linear and $\Gamma$ is only constrained by linear equalities and inequalities,

**(Mixed) Integer Programming:** as LP, but $\Gamma$ is constrained on integer numbers,

**Quadratic Programming (QP):** as LP, but $r$ may contain quadratic terms,

**Non-Linear Programming (NLP):** general case, both $r$ and the constraints may contain non-linear terms,

**Convex Programming (CP):** non-linear programming with a convex objective function $r$.

### 2.3.1. Linear Programming

Many practical optimization problems are linear programming (LP) problems, for example many of the problems discussed in operations research or logistics. LP can be solved with the Simplex algorithm [32] which exactly solves the problem after a finite number of steps. Although the Simplex algorithm is very efficient on a large variety of practical problems, it was possible to construct a problem which uses an exponential number of steps for every variant of the Simplex algorithm.

[2] The term *programming* was introduced for resource planning processes and is used as a synonym for the term *optimization* in literature.

Since most optimization problems in statistical learning are at least quadratic and many of the problems are non-convex, we will not discuss any details here.

### 2.3.2. Quadratic and Non-Linear Programming

LP has a property which now no longer needs to hold: any local optimum was also a global optimum in linear programming. This in general applies if the objective function is *convex*. In these cases, it suffices to define an optimization procedure which finds a local optimum also for non-linear programming. This local optimum will then also be the global one for convex objective functions.

The general problem of finding global solutions for general non-linear functions is an unsolved problem of mathematics. Solutions always use heuristics and we will discuss the probably most successful solution for this type of problems in Section 2.3.3.

#### 2.3.2.1. Nelder-Mead Optimization

The Nelder-Mead algorithm finds an approximatively correct local optimal solution if the non-linear objective function is more or less continuous [135]. The method uses a simplex, i.e. a polyeder with $p+1$ vertices in a $p$-dimensional space. With help of this simplex the behavior of the objective function is extrapolated and a new search point is determined. The most simple approach is to replace the worst point of the current simplex by the one mirrored over the centroid of the simplex. Although this algorithm is quite simple, it often does not deliver the expected optimal results. Therefore, we will discuss a better optimization scheme which is more suitable for the quadratic programming problem posed by structural risk minimization.

#### 2.3.2.2. Newton Optimization

The Newton algorithm is a standard method for numerically solving non-linear equations and systems of equations [39]. The goal is to find the *roots* of an arbitrary function $h$ (not the objective function), i.e. those $\gamma$ for which $h(\gamma) = 0$ holds. A root $\gamma$ of the function $h'$ corresponds to a local extremum of $h$ in $\gamma$.

The basis Newton algorithm is quite simple:

1. Estimate a start value $\gamma_0$ so that $|h(\gamma_0)|$ is small,
2. Calculate the tangent of $h$ at $\gamma_i$ and calculate the root of the tangent,
3. Use this root as an approximation of $\gamma_{i+1}$, i.e. of the searched root of $h$,
4. Repeat steps 2 and 3 with the new approximation until the changes converge.

**Selection of a Start Value** A widely used approach for selecting the start value is the bisection method. A large interval $[a, b]$ is chosen. If $h(a) \cdot h(b)$ is negative, then $h$ changes the sign in this interval. This means that $h$ has a root in the interval $[a, b]$. We now perform a binary search until the interval $[a, b]$ is small enough. We choose an arbitrary point of the resulting interval as start value $\gamma_0$.

**Calculation of the Next Root** The derivative approximates the tangent by a secant:

$$h'(\gamma) = \lim_{\Delta\gamma \to 0} \frac{h(\gamma + \Delta\gamma) - h(\gamma)}{\Delta\gamma}.$$

Without loss of generality, let the tangent be defined by the function $t(\gamma_n + d) = h(\gamma_n) + h'(\gamma_n) \cdot d$. For $d = \gamma - \gamma_n$ we get $t(\gamma) = h(\gamma_n) + h'(\gamma_n)(\gamma - \gamma_n)$. Let $\gamma_{n+1}$ be the only root of this linear function then we get the recursive calculation

$$0 = h(\gamma_n) + h'(\gamma_n)(\gamma_{n+1} - \gamma_n) \Rightarrow \gamma_{n+1} = \gamma_n - \frac{h(\gamma_n)}{h'(\gamma_n)}.$$

**Optimization with the Newton Method** Since we are searching for an extremum of the objective function $r$ we search a root of $r'$, hence we need the second derivative $r''$. Since most objective functions $r$ do not depend on one single parameter only, we also have to extend the Newton method for multiple dimensions.

For more than one dimension we replace the first derivative by the *gradient* $\nabla$. The second derivative is subsumed in the *Hesse Matrix*:

$$\mathrm{H}(h) = \left(\frac{\partial^2 h}{\partial\gamma_i \partial\gamma_j}\right) = \begin{pmatrix} \frac{\partial^2 h}{\partial\gamma_1\partial\gamma_1} & \frac{\partial^2 h}{\partial\gamma_1\partial\gamma_2} & \cdots & \frac{\partial^2 h}{\partial\gamma_1\partial\gamma_n} \\ \frac{\partial^2 h}{\partial\gamma_2\partial\gamma_1} & \frac{\partial^2 h}{\partial\gamma_2\partial\gamma_2} & \cdots & \frac{\partial^2 h}{\partial\gamma_2\partial\gamma_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 h}{\partial\gamma_n\partial\gamma_1} & \frac{\partial^2 h}{\partial\gamma_n\partial\gamma_2} & \cdots & \frac{\partial^2 h}{\partial\gamma_n\partial\gamma_n} \end{pmatrix}.$$

This changes our recursive calculation procedure for multiple dimensions to

$$\gamma_{n+1} = \gamma_n - [H(h(\gamma_n))]^{-1} \nabla h(\gamma_n).$$

**Quasi-Newton Method** The Newton method has several problems for practical optimization problems:

- the Hesse matrix is not always invertible,
- the optimization is not robust and stops due to small numerical instabilities, and

- the calculation of $H$ causes large runtimes for the whole optimization procedure.

The solution for these problems it to omit the exact calculation of the Hesse matrix $H$ and to approximate this matrix. Although this decreases runtime and makes the method more robust against numerical instabilities, it is no longer guaranteed to find the local optimum any longer. This approximative variant of the Newton method is called *Quasi-Newton* [137] and is the most widely used optimization procedure for the Support Vector Machine optimization problem stated above.

### 2.3.3. Non-Convex Programming

Besides the fact that the Quasi-Newton method does no longer guarantee that a local optimum is found, we will discuss in this work several problems with non-convex objective functions. In these cases the (Quasi-)Newton methods will fail since they will get stucked in the first local optimum they find and will not be able to identify the global optimum. This is in general true for all non-convex optimization problems and currently there is no exact solution known which is able to guarantee that the global optimal solution is found.

Therefore, other heuristics must be used instead of the ones discussed before which should be also usable for non-convex optimization problems. It is quite interesting that a very simple heuristic approach for solving non-convex optimization problems, namely *Evolutionary Algorithms* [6] proved to be the most successful solution for many real-world optimization tasks.

#### 2.3.3.1. Evolutionary Algorithms

We subsume several generic population-based heuristic optimization procedure under the term evolutionary algorithms (EA). The basic idea is to maintain a set of solution candidates, called *population*. Each of these search points or *individuals* can be evaluated for the value of the objective function. Hence, if we have several individuals $\gamma_i$ it must be possible to calculate $r(\gamma_i)$ for all the individuals in each population. Since nothing more than the evaluation method is necessary, EA are also called black-box optimization schemes since they can even be applied on problems where nothing is known about the problem itself.

The basis of each evolutionary algorithm is a loop in which several operations on the population are performed:

**Selection:** those individuals with higher values $r(\gamma_i)$ (for a maximization problem) should be selected with higher probability into the next population,

**Recombination:** several individuals $\gamma_i$ are selected and their parameters are partly exchanged which results in new search points (crossover),

**Mutation:** individuals $\gamma_i$ are selected from the current population and offsprings are created by slightly changing some of the parameters of those $\gamma_i$. Small changes should have higher probabilities than larger changes.

It can easily be seen that the idea of EA was inspired by biological evolution. Today, however, a lot of variants of EA exist and most of them do not rely on any biological idea any longer.

**Genetic Algorithms** Evolutionary algorithms like they were described above are the most general term for this class of optimization techniques. Depending on the structure of the individuals, i.e. the search space defining the search points or individuals, one can further distinguish different types of evolutionary algorithms.

The first class of evolutionary algorithms is called *Genetic Algorithms* (GA). Those optimization techniques work on the $d$-dimensional search space $\mathbb{B}^d$ where each individual $\gamma_i \in \mathbb{B}^n$ is described as a boolean vector of length $n$. While this does not affect the used selection and recombination schemes, the mutation is usually restricted to a random bit flip for each position of the vector with probability $1/n$. This ensures that in expectation only one bit is flipped in each generation and individual and therefore small changes have a higher probability than larger changes.

**Evolution Strategies** Another important subclass of evolutionary algorithms is called *Evolution Strategies* (ES) [14]. Here, the individuals $\gamma_i$ no longer consist of boolean but of real numbers, hence $\gamma_i \in \mathbb{R}^n$. Again, the selection and recombination do not have to be changed but we have to use a specific mutation operator. A commonly used variant adds to each of the coefficients of an individual $\gamma_i$ a small Gaussian with mean 0 and standard deviation $\sigma$. Small values for $\sigma$ infers only small steps to the optimum, but large values of $\sigma$ could lead to missing the actual optimum if you are close to it. For that reason, simple adaptation rules for $\sigma$ are used like the 1/5-rule [150]. The 1/5 rule will adapt the variance of the mutation depending on the measured success probability. The mutation strength $\sigma$ is increased after $k$ generations, if the measured success probability is larger than 1/5 and decreased if the probability is smaller 1/5. The parameters $k$ and the levels of increase and decrease are additional parameters of an evolution strategies optimization scheme.

**Genetic Programming** The last subclass of evolutionary algorithms discussed here and also used in this thesis are the *Genetic Programming* (GP) algorithms. Here, the individuals $\gamma_i$ do not have a fixed length but the length and structure of the individuals

may vary during the optimization. Most often, the individuals are not even vectors but consist of more complicated structures like trees or graphs. We will discuss the application of genetic programming together with the structure of individuals and the search point operations in detail in Chapter 7.

**Particle Swarm Optimization** Although *Particle Swarm Optimization* (PSO) [78] are not a direct subclass of evolutionary algorithms they are also inspired by phenomena which can be observed in nature and will hence also be discussed here. The basic idea of the particle swarm optimization approach is to keep track of a swarm of particles in a multidimensional space where each particle has a position $\gamma_i$ and a velocity $v_i$. The particles fly through hyperspace and have two essential reasoning capabilities: their memory of their own best position and knowledge of the global best. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions.

The particle position and velocity update equations are quite simple:

$$v_i^{new} \leftarrow w_{inertia} v_i^{old} + w_{global} r_{global} (\gamma^{globalbest} - \gamma_i^{old}) + w_{local} r_{local} (\gamma^{localbest} - \gamma_i^{old})$$
$$\gamma_i^{new} \leftarrow \gamma_i^{old} + v_i^{new}$$

The parameters $w$ are weight parameters for the different aspects of the update equation. The parameters $r$ are random numbers between 0 and 1 introducing some randomness into the search.

Particle swarm optimization is very similar to evolution strategies since it also works on the space $\mathbb{R}^n$. In contrast to evolution strategies, however, the gradient information in form of the velocity vectors is much more used instead of only randomly changing the current individuals' parameters and PSO hence is expected to reach the optimum faster in unimodal cases.

## 2.4. Multi-Objective Optimization

We already have stated that evolutionary algorithms are very successful in solving real-world optimization problems, even if the underlying problem is non-convex and nothing is known about the problem's nature at all. In this work, we also use evolutionary algorithms for another reason: they are the only efficiently working optimization procedures which are able to simultaneously optimize not only one but several objective functions which might even compete which each other.

Until now, we only have defined a single objective function $r$. Let us now consider a set $r_1, \ldots, r_v$ of objectives which should all be optimized. Simultaneously optimizing

conflicting criteria can be achieved by transforming the problem into a single-objective optimization problem. Therefore, user defined parameters have to be used in order to weigh the criteria. However, very often the user has no idea of criteria weights and, furthermore, there exist no simple decision about correct or wrong solutions. We try to maintain as much information as possible and aim at finding all solutions which are optimal for arbitrary criteria weight vectors. These solutions are called *Pareto-optimal*.

The multi-objective search space of a maximization problem is subject to a partial order:

**Definition 2.14 (Domination)** *A solution* $\gamma_1$ dominates *a solution* $\gamma_2$ *(written as* $\gamma_1 \succ \gamma_2$*) if for the* $v$ *criteria* $r_i$ *the following is true:*

$$\forall i \in \{1, \ldots, v\} : r_i(\gamma_1) \geq r_i(\gamma_2) \quad \wedge \quad \exists i \in \{1, \ldots, v\} : r_i(\gamma_1) > r_i(\gamma_2).$$

Our selection scheme needs to decide if a solution is dominated by a set $\Gamma_D$ of solutions. We define:

**Definition 2.15 (Non-Domination)** *A solution* $\gamma$ *is* non-dominated *by a set of solutions* $\Gamma_D$ *if* $\nexists \gamma_D \in \Gamma_D : \gamma_D \succ \gamma$.

Now we are able to define what we mean with Pareto-optimal solutions:

**Definition 2.16 (Pareto-Optimal)** *A solution* $\gamma$ *is* Pareto-optimal *if* $\gamma$ *is non-dominated by the complete solution space* $\Gamma$.

### 2.4.1. Multi-Objective Evolutionary Optimization

The usual approach for multi-objective problems are evolutionary algorithms which can optimize more than one target function by introducing special selection operators [202]. Traditional approaches in the field of mathematical programming must be applied more than once for multi-objective optimization [198]. Due to the population based approach of evolutionary algorithms a broad selection of Pareto-optimal solutions can be found during one run. The user can select one of these solutions after optimization. Additionally, multi-objective evolutionary algorithms do not strongly depend on form and continuity of the Pareto-optimal set [27].

Many multi-objective selection techniques were proposed for EAs during the last years. It turned out that NSGA-II is currently one of the best solutions for this task in a wide range of practical relevant optimization problems [36]. NSGA-II employs a selection technique which first sorts all individuals into levels of non-domination. Individuals from the first levels are added to the next generation until the desired population size is reached. Before adding individuals from the last possible level, this level is sorted with

respect to the crowding distance in order to preserve diversity in the population. Since NSGA-II proved to be superior compared to other selection techniques for the problems discussed in this work, we choose NSGA-II as standard selection for all optimizations.

A basic condition to pose an multi-objective optimization problem properly is that the described criteria are actually in conflict to each other. By improving on one criterion, we cannot simultaneously improve on the other criteria. Only problems for which this condition holds are sound and can be properly solved by multi-objective optimization.

#### 2.4.1.1. Guided Multi-Objective Optimization

We will see that evolutionary multi-objective optimization techniques will be able to deliver a representative set of Pareto-optimal solutions in all discussed problems of this thesis. Since this set sometimes becomes rather large and the analyst has to choose a final solution from such a set which is applied to his practical problem, it might be a good idea to incorporate techniques for selecting a final solution from Pareto sets or guiding the search for such a solution if any information is available beforehand.

Several preference based strategies have been proposed in combination with evolutionary multi-objective optimization. The first class of guided optimization techniques include those which prefer a set of solutions around one or several *reference points* [37]. During the NSGA-II selection process of the evolutionary algorithm those points from the ranks are preferred with smaller distances to the reference point(s). The result will hence be a partial Pareto front. It is interesting to note that these reference points do not need to actually exist in order to define a preferred direction of the Pareto front. Other proposed techniques combine a light beam search with the NSGA-II procedure which will also result in partial Pareto sets and ease the final decision process for a final solution [35].

A related approach for guiding the search for a final solution from a found Pareto set is based on the concept of *desirability indices* [174, 180]. These one- or two-sided index functions can be used to weigh the objective functions in a way that certain regions of the Pareto front are preferred. Both approaches, the reference point approach and the desirability index approach lead to partial Pareto sets which might ease the selection of a final solution and even work for noisy domains [101].

On the one hand, these methods work well when the analyst has at least a slight idea beforehand which parts of the Pareto front might interest him more. Another advantage of these guidance methods is that they can easily be incorporated into the multi-objective optimization method sketched above. On the other hand, these guidance methods completely rely on the definition of a reference point or of desirability function which is a hard task in cases where no optimal criteria weighing is known at all beforehand. Unfortunately, the weighting is most often not known for the problems in this thesis (compare

to the results of the next chapter) and so we will rely on the original NSGA-II which of course can easily be extended in cases where the analyst has a preference for a region before the optimization starts.

# Part I.

# Learning

CHAPTER 3

# Multi-Objective Learning

In this chapter, we embed evolutionary computation into statistical learning theory. We have discussed the connection between large margin optimization and statistical learning in Chapter 2 and have seen why this paradigm is successful for many pattern recognition problems. We now embed evolutionary computation into the most prominent representative of this class of learning methods, namely into Support Vector Machines (SVM). In contrast to former applications of evolutionary algorithms to SVMs, we do not only optimize the method or kernel parameters. We rather use both, evolution strategies and particle swarm optimization, in order to directly solve the posed constrained optimization problem. Transforming the problem into the Wolfe dual reduces the total runtime and allows the usage of kernel functions. Exploiting the knowledge about this optimization problem leads to a hybrid mutation which further decreases convergence time while classification accuracy is preserved. We will show that evolutionary SVMs are at least as accurate as their quadratic programming counterparts on six real-world benchmark data sets. The evolutionary SVM variants frequently outperform their quadratic programming competitors in terms of the objective function. Additionally, the proposed algorithm is more generic than existing traditional solutions since it will also work for non-positive semidefinite kernel functions and for several, possibly competing, performance criteria. This chapter will concentrate on the latter aspect while the following chapters will also concentrate on the optimization for non-convex optimization problems in statistical learning.

## 3.1. Single-Objective Evolutionary Support Vector Machines

We first show how evolutionary algorithms can be used to optimize the dual form of the optimization problem of SVMs. We will see that the resulting SVM performs similar to

the traditional SVMs based on quadratic programming. After the evaluation of the new SVM, called EvoSVM, we will discuss how this learning scheme can be extended

- to solve non-convex optimization problems (next chapters), and
- to perform the complete trade-off between training error and model complexity (this chapter).

### 3.1.1. Motivation for Evolutionary Support Vector Machines

Usually, the optimization problem posed by large margin methods is solved by methods known from quadratic programming problems, e.g. the (Quasi-)Newton method. However, there are some drawbacks with these approaches. First, no unique global optimum exists for kernel functions which are not positive semidefinite. Such an *indefinite* function $f$ with arity 2 might deliver a positive or a negative result for $f(x, x)$ depending on the value of $x$. Kernels based on non-positive semidefinite functions resemble a (partial) distance instead of a similarity measure. The optimization problem of large margin methods is then no longer guaranteed to be concave since the sign of the calculations might switch depending on the location in the search space. As a result, the objective function would no longer consist of a unique maximum only but of several maxima. This is also called a multi-modal problem.

In these cases, quadratic programming is not able to find satisfying solutions at all. Moreover, most implementations do not even terminate [61]. There exist several useful non-positive kernels [96], among them the sigmoid kernel which simulates a neural network [24, 168]. Therefore, a more generic optimization scheme based on evolutionary strategies was recently proposed by the author of this thesis which allows such non-positive kernels without the need for omitting the more efficient dual optimization problem [108, 109] which was proposed by other work [140]. This will be the topic of Chapter 4.

Another drawback of traditional SVMs is that they are not able to optimize several performance measures at the same time. Traditional SVMs try to maximize the prediction accuracy alone. However, depending on the application area other specific performance criteria should be optimized instead of or additionally to prediction accuracy. Although first attempts were made to incorporate multivariate performance measures into SVMs [75], the problem is not generally solved and no solution exist for competing criteria. This problem as well as the general trade-off between training error and capacity could be easily solved by an (multi-objective) evolutionary optimization approach.

Several former applications of evolutionary algorithms to SVMs exist. The first group include the optimization of method and kernel parameters [55, 159]. Evolutionary approaches can easily be used to optimize parameters like the weighting factor $C$ or kernel

parameters like $\gamma$ from the RBF kernel function. The result of a cross validation run is used as fitness measurement. This outer optimization scheme of course aims at the model selection process discussed in the introduction. But unlike the approach which will be discussed in this chapter, the outer parameter optimization needs several complete learning runs for different parameter settings and a complete validation run, e.g. a 10-fold cross validation, for performance estimation.

Other applications of evolutionary algorithms include the selection of optimal feature subsets [56] by means of genetic algorithms and the creation of new kernel functions by means of genetic programming [68]. Here a set of rules is employed to create new kernel functions in a tree based structure optimized by genetic programming. This latter approach is particularly interesting for the SVM discussed in this and the next chapters since it cannot be guaranteed that the resulting kernel functions are again positive semi-definite. This can not be solved by traditional SVM but by the evolutionary variant proposed here.

In contrast to all of the approaches discussed above, we embed evolutionary algorithms into the learning machine itself and solve the optimization problem of large margin methods like SVMs in its dual form. By doing this, we can avoid the major drawback connected to traditional SVM learning. Although the statistical learning theory takes into account both the training error and the model complexity, the user still has to define a weighting factor for both conflicting criteria. The search for this parameter is usually a non-trivial and very time consuming task as it has been described above.

In this chapter, we first show that the models built by an SVM by means of evolutionary algorithms perform very similar to those of traditional SVM implementations. For this comparison, we will just replace the inner optimization scheme of SVM by an evolutionary algorithm and let this algorithm solve the traditional single-objective optimization problem. We then propose to embed multi-objective evolutionary algorithms into SVM. This allows, for a first time, to explicitely optimize the inherent model selection trade-off which is the basic idea of statistical learning theory without applying time-consuming outer wrapper and validation approaches for optimizing the trade-off like it was described above. The result of the proposed approach is a Pareto front in the space of training error vs. model complexity and gives interesting insights into the nature of the problem at hand. Traditional SVM would only deliver a single point of this Pareto front for each different selection of the parameter $C$ while the approach proposed here delivers the complete sensible front in one single optimization run. By using a hold-out data set as a test set for the resulting models, we derive a second front showing the generalization error. Both, the Pareto front and the generalization error plot allows for a quick selection of the final solution $f(x, \gamma)$ from the Pareto front without the time-consuming optimization of a weighting factor. We refer to this as *overfitting control* since for a first time the data mining algorithm itself derives all necessary information and can hence

control the overfitting itself instead of having the user make this decision by selecting a parameter.

### 3.1.2. Evolutionary Computation for Large Margin Optimization

Before we will discuss the multi-objective optimization of both error and complexity, we will first analyze the effect by replacing the usual quadratic programming optimization by an *evolution strategies* (ES) approach [14] or by *particle swarm optimization* (PSO) [78]. In this section, we will describe both a straightforward application of these techniques and how we can exploit some information about our optimization problem and incorporate that information into our search operators.

#### 3.1.2.1. Solving the Dual Problem and Other Simplifications

The used optimization problem is the dual problem for non-linearly separable data developed in Section 2.2.2.3 (Problem 2.5):

**Problem 3.1 (Dual SVM Problem (final))** *The dual optimization problem for non-linear Support Vector Machines on non-linearly separable data is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \\ &\textit{subject to } \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

Of course it would also be possible to directly optimize the original form of our optimization problem depicted in equations 2.7 and 2.8. That is, we could directly optimize the weight vectors and the offset. But there are two drawbacks: first, the costs of calculating the fitness function would be much higher for the original optimization problem since the fulfillment of all $n$ constraints must be recalculated for each new hyperplane. It is far easier to check if all $0 \leq \alpha_i \leq C$ apply. Second, it would not be possible to allow non-linear learning with efficient kernel functions in the original formulation of the problem. This can easily be seen in case of the radial base function kernel where the explicit transformation $\Phi$ would lead to infinite spaces where the calculation of the dot product would no longer be possible. Furthermore, the kernel matrix $K$ with $K_{ij} = k\left(x_i, x_j\right)$ can be calculated beforehand in case of the dual optimization problem and the training data is never used during optimization again. This further reduces the needed runtime for optimization since the kernel matrix calculation is done only once. In cases where the kernel matrix is too large and does not fit into the main memory, a simple caching

strategy can be applied in order to prevent unnecessary recalculations. This is of course also true for traditional SVM.

This is a nice example for a case, where transforming the objective function beforehand is both more efficient and allows enhancements which would not have been possible before. Transformations of the fitness functions became also a very important topic in evolutionary algorithms research recently [175].

Another efficiency improvement can be achieved by formulating the problem with $b = 0$. All solution hyperplanes must then contain the origin and the constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$ will vanish. This is a mild restriction for high-dimensional spaces since the number of degrees of freedom is only decreased by one [22]. This means that in the calculation of $w$ and $b$ only one element is fixed during the optimization and hence we do not have to cope with this equality constraint which would take an additional runtime of $O(n)$ for each generation. After optimization has been finished, we can use the constraint in order to calculate a value for $b$. This also is a technique frequently employed by traditional SVM implementations, too.

#### 3.1.2.2. EvoSVM and PsoSVM

We will first analyze the effect of using evolutionary algorithms on the traditional single-objective optimization problem. Later, we will omit the parameter $C$ and switch to multi-objective optimization problems instead.

The first approach (*EvoSVM-G*, *G* for *G*aussian mutation) merely utilizes a standard ES optimization. Individuals are the real-valued vectors $\alpha$ and mutation is performed by adding a Gaussian distributed random variable with standard deviation $C/10$. In addition, a variance adaptation is conducted during optimization (1/5 rule [150]). The 1/5 rule will adapt the variance of the mutation depending on the measured success probability. The mutation strength is increased by factor 2 after 100 generations, if the measure success probability is larger than 1/5 and decreased by factor 2 if the probability is smaller than 1/5.

Crossover probability is high (0.9). We use tournament selection with a tournament size of 0.25 multiplied by the population size. This means that for each place in the next generation 25% of the current population are randomly selected with replacement and the best individual will be added to the next generation.

The initial individuals are random vectors with $0 \leq \alpha_i \leq C$. The maximum number of generations is 1000 and the optimization is terminated if no improvement occurred during the last 5 generations. The population size is 10.

The second version is called *EvoSVM-S* (*S* for *s*witching mutation). Here we utilize the fact that only a small amount of input data points will become support vectors

```
for i = 1 to n do {
    if (random(0, 1) < 1/n) do {
        if (alpha_i > 0) do {
            alpha_i = 0;
        } else do {
            alpha_i = random(0, C);
        }
    }
}
```

**Figure 3.1.:** A simple hybrid mutation which should speed-up the search for sparser solutions. It contains elements from standard mutations from both genetic algorithms and evolution strategies.

(*sparsity*). On the other hand, one can often observe that non-zero alpha values are equal to the upper bound $C$ and only a very small amount of support vectors exists with $0 < \alpha_i < C$. Therefore, we just use the well known mutation of genetic algorithms and switch between 0 and $C$ with probability $1/n$ for each $\alpha_i$. The other parameters are equal to those described for the EvoSVM-G.

Using this switching mutation is inspired by genetic algorithms and only allows the optimization parameter values $\alpha_i = 0$ or $\alpha_i = C$. Instead of a complete switch between 0 and $C$ or a smooth change of all values $\alpha_i$ like the Gaussian mutation does, we developed a hybrid mutation combining both elements. That means that we check for each $\alpha_i$ with probability $1/n$ if the value should be mutated at all. If the current value $\alpha_i$ is greater than 0, $\alpha_i$ is set to 0. If $\alpha_i$ is equal to 0, $\alpha_i$ is set to a random value with $0 \leq \alpha_i \leq C$. Figure 3.1 gives an overview over this hybrid mutation. The function $random(a, b)$ returns an uniformly distributed random number between $a$ and $b$. The other parameters are the same as described for the EvoSVM-G. We call this version *EvoSVM-H* ($H$ for $h$ybrid).

As was mentioned before, the optimization problem usually is concave and the risk for local extrema is small. Hence, we also applied a PSO technique which is more directed to (local) optima than evolutionary algorithms. It should be investigated if PSO, which is similar to the usual quadratic programming approaches for SVMs in a sense that the gradient information is exploited, is able to find a global optimum in shorter times. Please see Section 2.3.3.1 for further details on particle swarm optimization.

We call this last version of an evolutionary SVM *PsoSVM* and use a standard PSO with inertia weight 0.1, local best weight 1.0, and global best weight 1.0. The inertia weight is dynamically adapted during optimization [78].

### 3.1.3. Experiments and Results

We now evaluate the proposed evolutionary optimization SVM and compare our implementation to the quadratic programming approaches usually applied to large margin problems. The experiments demonstrate the competitiveness in terms of the original optimization problem, the classification error minimization, the runtime, and the robustness.

In order to compare the evolutionary SVM described in this paper with standard SVM implementations, we also applied two other SVMs on all data sets. Both SVMs use a slightly different optimization technique based on quadratic programming. The used implementations were *mySVM* [160] and *LibSVM* [25]. The latter is an adaptation of the widely used $SVM^{light}$ [72]. All experiments were performed with the machine learning environment RapidMiner [125][1].

#### 3.1.3.1. Data Sets

We apply the discussed EvoSVM and PsoSVM as well as the other SVM implementations on two synthetical and six real-world benchmark data sets. The data set Spiral consists of two intertwingling spirals of different classes. For checkerboard, the data set consists of two classes layed out in a $8 \times 8$ checkerboard. In addition, we use six benchmark data sets from the UCI machine learning repository [136] and the StatLib data set library [173]. We choose these data sets because they already define a binary classification task, consist of real-valued numbers only and do not contain missing values. Therefore, we did not need to perform additional preprocessing steps which might introduce some bias. The properties of all data sets are summarized in Table 3.1. The default error corresponds to the error a lazy default classifier would make by always predicting the major class. Classifiers must produce lower error rates in order to learn at all instead of just guessing.

We use an RBF kernel for all SVM and determine the best parameter value for $\sigma$ with a grid search parameter optimization for mySVM. This ensures a fair comparison since the parameter is not optimized for the evolutionary SVM. Possible parameters were 0.001, 0.01, 0.1, 1 and 10. The optimal value for each data set is also given in Table 3.1.

#### 3.1.3.2. Comparison for the Objective Function

The first question is if the evolutionary optimization approach is capable of delivering comparable results with respect to the objective function, i.e. the dual optimization

[1] `http://www.rapidminer.com/`

| Data Set | $n$ | $m$ | Source | $\sigma$ | Default |
|---|---|---|---|---|---|
| Spiral | 500 | 2 | Synthetical | 1.000 | 50.00 |
| Checkerboard | 300 | 2 | Synthetical | 1.000 | 50.00 |
| Liver | 346 | 6 | UCI | 0.010 | 42.03 |
| Sonar | 208 | 60 | UCI | 1.000 | 46.62 |
| Diabetes | 768 | 8 | UCI | 0.001 | 34.89 |
| Lawsuit | 264 | 4 | StatLib | 0.010 | 7.17 |
| Lupus | 87 | 3 | StatLib | 0.001 | 40.00 |
| Crabs | 200 | 7 | StatLib | 0.100 | 50.00 |

**Table 3.1.:** The evaluation data sets. $n$ is the number of data points, $m$ is the dimension of the input space. The kernel parameter $\sigma$ was optimized for the comparison SVM learner *mySVM*. The last column contains the default error, i. e. the error for always predicting the major class in percent.

problem 2.5. We applied all SVM implementations on all data sets and calculated the value for the objective function.

We perform a $k$-fold cross validation in order to determine the objective function values of all methods. The data set $T$ is divided into $k$ disjoint subsets $T_i$. For each $i \in \{1, \ldots, k\}$ we use $T \backslash T_i$ as training set and the remaining subset $T_i$ as test set. If $F_i$ is the value for the objective function on the training set $T \backslash T_i$, we calculate the average value

$$F = \frac{1}{k} \sum_{i=1}^{k} \frac{F_i}{|T \setminus T_i|}$$

over all training sets in order to measure the performance and a standard deviation. In our experiments we choose $k = 20$, i. e. for each method the average and standard deviation of 20 different runs is reported.

Table 3.2 shows the results. It can clearly be seen that for all data sets the EvoSVM approach delivers statistically significant higher values than the other SVM approaches in comparable time.

#### 3.1.3.3. Comparison for Positive Kernels

We now examine the generalization performance of all SVM implementations for a regular positive semidefinite kernel function (a radial basis function kernel).

| Data Set | Algorithm | Objective Function | Time[s] |
|---|---|---|---|
| Spiral | EvoSVM | **99.183 ± 5.867** | 11 |
| | mySVM | $-283.699 \pm 7.208$ | 6 |
| | LibSVM | $-382.427 \pm 12.295$ | 7 |
| Checkerboard | EvoSVM | **94.036 ± 1.419** | 4 |
| | mySVM | $-114.928 \pm 1.923$ | 2 |
| | LibSVM | $-127.462 \pm 1.595$ | 3 |
| Liver | EvoSVM | **103.744 ± 7.000** | 5 |
| | mySVM | $-1301.563 \pm 84.893$ | 3 |
| | LibSVM | $-1640.546 \pm 80.228$ | 3 |
| Sonar | EvoSVM | **8.436 ± 2.937** | 3 |
| | mySVM | $-558.333 \pm 31.249$ | 2 |
| | LibSVM | $-491.039 \pm 26.196$ | 2 |
| Diabetes | EvoSVM | **209.491 ± 14.003** | 10 |
| | mySVM | $-90.856 \pm 3.566$ | 8 |
| | LibSVM | $-108.242 \pm 3.886$ | 7 |
| Lawsuit | EvoSVM | **80.024 ± 18.623** | 3 |
| | mySVM | $-8790.429 \pm 308.996$ | 1 |
| | LibSVM | $-9061.420 \pm 303.227$ | 1 |
| Lupus | EvoSVM | **29.074 ± 2.582** | 1 |
| | mySVM | $-603.404 \pm 52.356$ | 1 |
| | LibSVM | $-504.564 \pm 41.593$ | 1 |
| Crabs | EvoSVM | **32.413 ± 1.231** | 2 |
| | mySVM | $-90.856 \pm 3.566$ | 1 |
| | LibSVM | $-108.242 \pm 3.886$ | 1 |

**Table 3.2.:** Comparison of the different implementations with regard to the objective function (the higher the better). The results are obtained by a 20-fold cross validation, the time is the cumulated time for all runs. It can easily be seen that the evolutionary version of the SVM always provides better results for the objective function than the quadratic programming solutions. Bold fonts mark significantly better results on a 1% confidence level.

| Data Set | Algorithm | Error | Time[s] |
|---|---|---|---|
| Spiral | EvoSVM | $16.40 \pm 4.54$ | 11 |
| | mySVM | $17.20 \pm 4.58$ | 6 |
| | LibSVM | $17.80 \pm 3.94$ | 7 |
| Checkerboard | EvoSVM | $22.67 \pm 4.90$ | 4 |
| | mySVM | $24.00 \pm 6.29$ | 2 |
| | LibSVM | $23.00 \pm 5.04$ | 3 |
| Liver | EvoSVM | $33.92 \pm 6.10$ | 5 |
| | mySVM | $31.31 \pm 5.86$ | 3 |
| | LibSVM | $33.33 \pm 4.51$ | 3 |
| Sonar | EvoSVM | $16.40 \pm 9.61$ | 3 |
| | mySVM | $14.50 \pm 9.61$ | 2 |
| | LibSVM | $13.98 \pm 7.65$ | 2 |
| Diabetes | EvoSVM | $25.52 \pm 4.30$ | 10 |
| | mySVM | $23.83 \pm 4.46$ | 8 |
| | LibSVM | $24.48 \pm 4.81$ | 7 |
| Lawsuit | EvoSVM | $31.00 \pm 11.08$ | 3 |
| | mySVM | $29.50 \pm 5.56$ | 1 |
| | LibSVM | $36.72 \pm 2.01$ | 1 |
| Lupus | EvoSVM | $23.89 \pm 14.22$ | 1 |
| | mySVM | $24.17 \pm 12.87$ | 1 |
| | LibSVM | $24.17 \pm 12.87$ | 1 |
| Crabs | EvoSVM | $3.50 \pm 3.91$ | 2 |
| | mySVM | $3.00 \pm 2.45$ | 1 |
| | LibSVM | $3.50 \pm 3.91$ | 1 |

**Table 3.3.:** Comparison of the different implementations with regard to the classification error (the lower the better). The results are obtained by a 20-fold cross validation, the time is the cumulated time for all runs. There is no significant difference between the results on a 1% significance level according to an ANOVA test.

| $C = 1$ | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Liver** | | **Ionosphere** | | **Sonar** | | **Lawsuit** | | **Lupus** | | **Crabs** | |
| | Error | T | Error | T | Error | T | Error | T | Error | T | Error | T |
| EvoSVM-G | 34.71±8.60 | 68 | 10.81±5.71 | 80 | **14.03±4.52** | 26 | 2.05±1.87 | 52 | 25.20±11.77 | 8 | **2.25±3.72** | 25 |
| EvoSVM-S | 35.37±6.39 | 4 | 8.49±3.80 | 9 | 17.45±6.64 | 6 | 2.40±1.91 | 10 | 30.92±12.42 | <1 | 4.05±4.63 | 2 |
| EvoSVM-H | 34.97±7.32 | 7 | **6.83±3.87** | 22 | 15.41±6.39 | 10 | 2.01±1.87 | 14 | **24.03±13.68** | 1 | 3.95±4.31 | 7 |
| PsoSVM | 34.78±4.95 | 8 | 9.90±4.38 | 9 | 16.94±5.61 | 7 | 3.02±2.83 | 3 | 25.22± 7.67 | <1 | 3.40±3.70 | 2 |
| mySVM | 33.62±4.31 | 2 | 8.56±4.25 | 4 | 15.81±5.59 | 2 | **1.89±2.51** | 1 | 25.28± 8.58 | 1 | 3.00±3.32 | 1 |
| LibSVM | **32.72±5.41** | 2 | 7.70±3.63 | 3 | 14.60±4.96 | 3 | 2.41±2.64 | 1 | 24.14±12.33 | 1 | 3.00±4.58 | 1 |
| F Test | **3.20 (0.01)** | | **9.78 (0.00)** | | **6.19 (0.00)** | | 1.51 (0.19) | | **11.94 (0.00)** | | **2.25 (0.05)** | |
| $C = 0.1$ | | | | | | | | | | | | |
| | **Liver** | | **Ionosphere** | | **Sonar** | | **Lawsuit** | | **Lupus** | | **Crabs** | |
| | Error | T | Error | T | Error | T | Error | T | Error | T | Error | T |
| EvoSVM-G | 33.90±4.19 | 74 | 9.40±6.14 | 89 | **21.72±6.63** | 35 | **2.35±1.92** | 50 | **24.90±10.51** | 7 | 7.20±4.36 | 27 |
| EvoSVM-S | 35.57±3.55 | 4 | 7.12±3.54 | 18 | 24.90±6.62 | 4 | 4.47±2.31 | 13 | 25.98±12.56 | <1 | 7.95±5.68 | 2 |
| EvoSVM-H | 34.76±4.70 | 5 | **6.55±4.61** | 23 | 24.40±6.09 | 11 | 4.16±3.14 | 19 | 26.51±13.03 | 1 | 6.50±5.02 | 2 |
| PsoSVM | 36.81±5.04 | 3 | 13.96±7.56 | 10 | 24.18±6.11 | 3 | 3.03±2.83 | 3 | 29.86±12.84 | 1 | 8.15±6.02 | 1 |
| mySVM | 42.03±1.46 | 2 | 35.90±1.35 | 2 | 46.62±1.62 | 2 | 7.17±2.55 | 1 | 41.25±6.92 | 1 | **6.50±4.50** | 1 |
| LibSVM | **33.08±10.63** | 2 | 11.40±6.52 | 3 | 22.40±6.45 | 3 | 4.55±3.25 | 1 | 25.29±16.95 | 1 | 21.00±12.41 | 1 |
| F Test | **34.46 (0.00)** | | **492.88 (0.00)** | | **323.83 (0.00)** | | **20.64 (0.00)** | | **64.83 (0.00)** | | **100.92 (0.00)** | |
| $C = 0.01$ | | | | | | | | | | | | |
| | **Liver** | | **Ionosphere** | | **Sonar** | | **Lawsuit** | | **Lupus** | | **Crabs** | |
| | Error | T | Error | T | Error | T | Error | T | Error | T | Error | T |
| EvoSVM-G | 42.03±1.46 | 75 | 35.90±1.35 | 86 | 45.33±2.20 | 39 | 7.17±2.55 | 55 | 40.00±6.33 | 7 | 26.20±12.66 | 27 |
| EvoSVM-S | 42.03±1.46 | 3 | 35.90±1.35 | 9 | 46.62±1.62 | 4 | 7.17±2.55 | 3 | 40.00±6.33 | <1 | 8.58±4.35 | 1 |
| EvoSVM-H | 42.03±1.46 | 3 | 35.90±1.35 | 20 | 46.27±1.42 | 12 | 7.17±2.55 | 3 | 40.00±6.33 | 1 | 7.00±4.00 | 2 |
| PsoSVM | **41.39±8.59** | 3 | 35.90±1.35 | 4 | **27.90±6.28** | 3 | 7.17±2.55 | 2 | 31.94±12.70 | 1 | 10.05±7.26 | 1 |
| mySVM | 42.03±1.46 | 2 | 35.90±1.35 | 2 | 46.62±1.62 | 2 | 7.17±2.55 | 1 | 40.00±6.33 | 1 | **6.50±4.50** | 1 |
| LibSVM | 42.03±1.46 | 2 | 35.90±1.35 | 3 | 28.46±10.44 | 2 | 7.17±2.55 | 1 | **26.11±16.44** | 1 | 50.00±0.00 | 1 |
| F Test | 0.52 (0.77) | | 0.00 (1.00) | | **442.46 (0.00)** | | 0.00 (1.00) | | **78.27 (0.00)** | | **1095.94 (0.00)** | |

**Table 3.4.:** Classification error (the lower the better), standard deviation, and runtime of all SVMs on the evaluation datasets for parameters $C = 1$, $C = 0.1$, and $C = 0.01$. The runtime $T$ is given in seconds. The last line for each table depicts the F test value and the probability that the results are not statistical significant.

We again use a $k$-fold cross validation but this time we calculate $E_i$ as the number of wrong predictions on the test set $T_i$ which leads to the average classification error of

$$E = \frac{1}{k} \sum_{i=1}^{k} \frac{E_i}{|T_i|}.$$

We again use $k = 20$ in our experiments.

Table 3.3 summarizes the results for $C = 1$. This value corresponds to $1/\left(1 - \sum K_{ii}\right)$ which is a successful heuristic for determining $C$ proposed by [63]. It can clearly be seen that the EvoSVM leads to classification errors comparable to those of the quadratic programming counterparts (mySVM and LibSVM).

The reason for slightly higher errors in some of the predictions of the quadratic programming approaches is probably a too aggressive termination criterion. Although this termination behavior further reduces runtime for mySVM and LibSVM, the classification error is often increased. On the other hand, for the cases, where the EvoSVM yields higher errors, the reason probably is a higher degree of overfitting due to the better optimization of the dual problem (see above). This can easily be augmented by slightly reducing the values of $C$ as it can be seen in the example of the Liver data set (compare $C = 1$ with $C = 0.1$ in Table 3.4).

Please note that the standard deviations of the errors achieved with the evolutionary SVM are similar to the standard deviations achieved with mySVM or LibSVM. We can therefore conclude that the evolutionary optimization is as robust as the quadratic programming approaches and differences mainly derive from different subsets for training and testing due to the used cross validation. The influence of the used randomized heuristics seems not to be larger.

Table 3.4 summarizes the results for different values of $C$ for both the EvoSVM and the PsoSVM only on the real-world benchmark datasets for a 10-fold cross validation. It can be seen that the EvoSVM variants frequently yield smaller classification errors than the quadratic programming counterparts (mySVM and LibSVM). For $C = 1$, a statistical significant better result was achieved by using LibSVM only for the Liver data set. For all other datasets, the evolutionary optimization outperforms the quadratic programming approaches. The same applies for $C = 0.1$. For rather small values of $C$ most learning schemes were not capable of producing better predictions than the default classifier (see Table 3.1). For $C = 0.01$, however, PsoSVM at least provides a similar accuracy to LibSVM.

It turns out that the standard ES approach EvoSVM-G using a mutation which adds a Gaussian distributed random variable often outperforms the other SVMs. However, the runtime for this approach might be too large in some practical situations. The mere GA based selection mutation switching between 0 and $C$ converges much faster but is often

less accurate. A part of the runtime differences between EvoSVM-S and the quadratic programming counterparts can probably also be traced back to different levels of code optimization.

The hybrid version EvoSVM-H combines the best elements of both worlds. It converges nearly as fast as the EvoSVM-S and is often nearly as accurate as the EvoSVM-G. In some cases (Ionosphere, Lupus) it even outperforms all other SVMs.

PsoSVM on the other hand does not provide the best performance in terms of classification error. However, it converged much earlier than the other competitors based on evolutionary approaches. This was expected since PSO exploits some gradient information similar to the Newton-based optimization methods used by traditional SVMs.

Please note that the standard deviations of the errors achieved with the evolutionary SVMs are similar to the standard deviations achieved with mySVM or LibSVM. We can therefore conclude that the evolutionary optimization is as robust as the quadratic programming approaches and differences mainly derive from the cross validation variances.

Therefore, evolutionary SVMs provide an interesting alternative to more traditional SVM implementations. Beside the similar results, EvoSVM is also capable of working with multi-objective optimization as we will discuss in the next sections.

## 3.2. Multi-Objective Statistical Learning

In the last sections, we have discussed how the internal optimization scheme traditionally used by Support Vector Machines can be replaced by evolutionary algorithms. We have seen that evolutionary SVMs turned out to be at least as accurate as their quadratic programming counterparts.

Now, we will finally discuss the major advantage of evolutionary SVM compared to traditional SVM solutions: we can explicitly optimize the inherent trade-off which is the basic idea of statistical learning theory. We do no longer need to apply time-consuming outer wrapper approaches for optimizing the weighting factor between both criteria. This is a clear advantage compared to the model selection process which has to be performed otherwise.

### 3.2.1. The Regularized Risk Consists of Multiple Objectives

We shortly discuss the different parts of the regularized risk introduced in Chapter 2 again and we will see that the regularized risk as it was defined in the previous chapter actually consists of two conflicting parts.

Let us remember that the instance space is defined as the Cartesian product $X = X_1 \times \ldots \times X_m$ of attributes $X_i \subseteq \mathbb{R}$. Let $Y$ be another set of possible labels. $X$ and $Y$ obey a fixed but unknown probability distribution $P(X, Y)$. We try to find a function $f(x, \gamma)$ which predicts the value of $Y$ for a given input $x \in X$. The function class $f$ depends on a vector of parameters $\gamma$. Finally, we define a *loss function* $L(Y, f(X, \gamma))$ in order to penalize errors during prediction. Since the mere minimization of the loss according to the training data is not appropriate to find a good generalization, we incorporate the *capacity* [185] of the used function into the optimization problem leading to the *regularized risk*:

**Definition 3.1 (Regularized Risk)** *Let $\Omega$ be a strictly monotonic increasing function. The* REGULARIZED RISK *is defined as*

$$R_{reg}(\gamma) = R_{emp}(\gamma) + \lambda\Omega(\gamma).$$

$\Omega$ is a function which measures the capacity of the function class $f$ depending on the parameter vector $\gamma$ (see [164] for more details). Since the empirical risk is usually a monotonically decreasing function of $\Omega$, both criteria are conflicting and we use $\lambda$ to manage the trade-off between training error and capacity.

We have seen that we need to use a class of approximation functions whose capacity can be controlled. We discussed that large margin methods define such a class. They embed regularized risk minimization by maximizing a margin between a linear function and the nearest data points. In general, we can define the underlying optimization problem as the maximization of the margin together with the minimization of a loss function:

**Definition 3.2 (Large Margin Problems)** LARGE MARGIN PROBLEMS *are defined as*

$$\textit{minimize } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} L\left(y_i, f(x_i)\right).$$

The vector $w$ is the normal vector of the separating hyperplane. Minimizing $||w||^2$ corresponds to maximizing the margin. The second term calculates the training error and $C$ corresponds to the trade-off parameter $\lambda$ known from the regularized risk.

Without loss of generality, we constrain the classes to $Y = \{-1, +1\}$. Furthermore, we use the so called Hinge loss as a loss function:

**Definition 3.3 (Hinge Loss)** *The following loss function is called* HINGE LOSS*:*

$$L(Y, f(X)) = (1 - Yf(X))_+$$

*with* $(a)_+ = \max(a, 0)$.

The definition of the Hinge loss leads to a very simple error count since it is possible to just measure the loss for each training data point – called slack variable $\xi_i$ – and to simply summarize all values. The primal definition 3.2 of the large margin problem can be combined with the Hinge loss. The slack variables $\xi_i$ vanish and the optimization problem can be transformed into its dual form (please refer to Chapter 2 for details):

**Problem 3.2 (Dual Form of the SVM Problem)** *The dual form of the SVM problem can be stated as*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \\ &\textit{subject to } \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

The parameter $C$ is a user defined weight for both conflicting parts of the optimization criterion. In the following, we will discuss how multi-objective optimization can be exploited to omit this parameter and deliver the full Pareto front for all possible trade-offs between complexity and training error. Please note that the variables $\alpha_i$ are constrained by the upper bound $C$, i.e. by the user defined trade-off factor and that the examples $x_i$ only occur in scalar products which was replaced by a kernel function $k = \langle \Phi\left(x_i\right), \Phi\left(x_j\right) \rangle$ for a (non-linear) mapping $\Phi$ in an arbitrary dot product space. This allows the search for a linearly separating hyperplane in high-dimensional spaces after a non-linear transformation and, hence, the separating of non-linearly separable data.

### 3.2.2. Explicit Trade-off between Error and Complexity

We have seen that we can use evolutionary algorithms as an optimization scheme for large margin methods as the SVM and can achieve similar classification accuracies in similar times. However, embedding evolutionary computation into large margin methods has the big advantage of a straightforward application of multi-objective selection schemes in order to simultaneously optimize several conflicting criteria. In this section, we divide the criteria of the objective function stated above into two optimization targets while the weighting factor C can be omitted. This leads to the following two optimization problems:

$$\begin{aligned} &\text{minimize } \frac{1}{2}||w||^2 \\ &\text{subject to } \forall_{i=1}^{n} : y_i \left(\langle w, x_i \rangle + b\right) \geq 1 - \xi_i \\ &\text{and } \forall_{i=1}^{n} : \xi_i \geq 0 \end{aligned} \tag{3.1}$$

and

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} \xi_i \\ &\text{subject to } \forall_{i=1}^{n} : y_i \left(\langle w, x_i \rangle + b\right) \geq 1 - \xi_i \\ &\text{and } \forall_{i=1}^{n} : \xi_i \geq 0. \end{aligned} \tag{3.2}$$

We will transform both objectives into their dual form in order to allow the efficient optimization of the problems including the usage of kernel functions.

### 3.2.3. First Objective: Maximizing the Margin

It is quite interesting that the first primal objective, i.e. maximizing the margin, can be transformed into the dual form in exactly the same way for all different large margin methods. The discriminating loss function is then omitted and the transformation is equivalent for all large margin learning methods including, for example SVMs as well as kernel logistic regression [112].

**Theorem 3.1 (Dual Form of Margin Maximization)** *The dual form of the first objective (maximize margin) for multi-objective SVMs is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \\ &\textit{subject to } \forall_{i=1}^{n} : \alpha_i \geq 0 \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

*Proof.* We introduce positive Lagrange multipliers into equation 3.1. We need multipliers $\alpha$ for the first set of inequality constraints and multipliers $\beta$ for the second set of inequality constraints:

$$L_p^{(1)}(w, b, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\langle w, x_i \rangle + b\right) + \xi_i - 1\right) - \sum_{i=1}^{n} \beta_i \xi_i$$

In order to find a solution we have to find the minimum by setting the derivatives to 0;

$$\begin{aligned}
\frac{\partial L_p^{(1)}}{\partial w}(w, b, \xi, \alpha, \beta) &= w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \\
\frac{\partial L_p^{(1)}}{\partial b}(w, b, \xi, \alpha, \beta) &= \sum_{i=1}^{n} \alpha_i y_i = 0, \\
\frac{\partial L_p^{(1)}}{\partial \xi_i}(w, b, \xi, \alpha, \beta) &= -\alpha_i - \beta_i = 0.
\end{aligned}$$

Plugging the derivatives into the primal objective function $L_p^{(1)}$ delivers

$$\begin{aligned}
L_p^{(1)}(w, b, \xi, \alpha, \beta) &= \frac{1}{2}||w||^2 - \sum_{i=1}^{n} -\alpha_i y_i \left\langle \sum_{j=1}^{n} \alpha_j y_j x_j, x_i \right\rangle + \sum_{i=1}^{n} \alpha_i \\
&= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle
\end{aligned}$$

The Wolfe dual must be maximized which leads to the formalization of the first objective of the multi-objective large margin method setting. The resulting problem is very similar to the usual dual SVM problem but without the upper bound $C$ for the $\alpha_i$ (again, the dot product is replaced by a kernel function $k$):

$$\begin{aligned}
&\text{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \\
&\text{subject to } \forall_{i=1}^{n} : \alpha_i \geq 0 \\
&\text{and } \sum_{i=1}^{n} \alpha_i y_i = 0.
\end{aligned}$$

□

### 3.2.4. Second Objective: Minimizing the Number of Training Errors

The second problem states that the sum of errors, i.e. the sum of the slack variables $\xi_i$, should be minimized. This optimization must be performed under the same inequality constraints as for the first objective.

**Theorem 3.2 (Dual Form of SVM Training Error Minimization)** *The dual form of the second objective (minimize training error) for multi-objective SVMs is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i \\ &\textit{subject to } \forall_{i=1}^{n} : \alpha_i \geq 0 \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

*Proof.* We start with equation 3.2 and add positive Lagrange multipliers $\alpha$ and $\beta$:

$$L_p^{(2)}(w, b, \xi, \alpha, \beta) = \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left( y_i \left( \langle w, x_i \rangle + b \right) + \xi_i - 1 \right) - \sum_{i=1}^{n} \beta_i \xi_i$$

The derivatives must again be set to 0 which leads to slightly different conditions on the derivatives of $L_p^{(2)}$:

$$\begin{aligned} \frac{\partial L_p^{(2)}}{\partial w}(w, b, \xi, \alpha, \beta) &= -\sum_{i=1}^{n} \alpha_i y_i x_i = 0, \\ \frac{\partial L_p^{(2)}}{\partial b}(w, b, \xi, \alpha, \beta) &= \sum_{i=1}^{n} \alpha_i y_i = 0, \\ \frac{\partial L_p^{(2)}}{\partial \xi_i}(w, b, \xi, \alpha, \beta) &= 1 - \alpha_i - \beta_i = 0. \end{aligned}$$

Plugging the derivatives into $L_p^{(2)}$ cancels out most terms because of the first two derivatives:

$$
\begin{aligned}
L_p^{(2)}(w,b,\xi,\alpha,\beta) &= \sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i y_i \langle w, x_i\rangle - \sum_{i=1}^{n}\alpha_i y_i b - \sum_{i=1}^{n}\alpha_i\xi_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\langle w, \alpha_i y_i x_i\rangle - \sum_{i=1}^{n}\alpha_i\xi_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\xi_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\beta_i\xi_i
\end{aligned}
$$

Together with the third derivative we can replace the $\beta_i$ by $1-\alpha_i$ leading to

$$
L_p^{(2)}(w,b,\xi,\alpha,\beta) = \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\alpha_i\xi_i + \sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\alpha_i
$$

The Wolfe dual must again be maximized which leads to the second objective of the multi-objective SVM setting. Maximizing the sum of $\alpha_i$ corresponds to transforming each example into a support vector. In the limit, this means that the training set is merely memorized instead of generalized which is an indication of overfitting or training error minimization respectively. The second objective (minimize error) for multi-objective SVMs is hence given as

$$
\begin{aligned}
&\text{maximize } \sum_{i=1}^{n}\alpha_i \\
&\text{subject to } \forall_{i=1}^{n} : \alpha_i \geq 0 \\
&\text{and } \sum_{i=1}^{n}\alpha_i y_i = 0.
\end{aligned}
$$

□

### 3.2.5. Multi-Objective Evolutionary Algorithms for Large Margin Learning

We have defined all objectives and constraints for the multi-objective setting of large margin learning and will now discuss some details of the optimization in this section.

#### 3.2.5.1. Definition of the Objectives

The problems stated in Theorem 3.1 and 3.2 can be used as objectives for a multi-objective evolutionary algorithm (MOEA). Both objectives share a common term, the sum $\sum_{i=1}^{n}\alpha_i$. Since this sum as part of the first objective is not conflicting with the

second objective as a whole, we can simply omit the calculation of the sum of $\alpha_i$ for the first objective.

Other efficiency improvements like those discussed in Section 3.1.2.1 can of course also be exploited including that of not recalculating the sum $\sum_{i=1}^{n} \alpha_i y_i = 0$ each generation anew. If the equality constraint should be fulfilled (e.g. for small numbers of dimensions where omitting the constraint would make a difference), it can simply be defined as a third objective by maximizing $-|\sum_{i=1}^{n} \alpha_i y_i|$. The whole set of objectives is then given as a maximization of the terms (in the SVM setting):

**Corollary 3.1 (Objectives for the Multi-Objective SVM)** *For the multi-objective SVM, the set of maximization objectives consists of*

$$\begin{aligned} \textit{(I)} \quad & -\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k\left(x_i, x_j\right), \\ \textit{(II)} \quad & \sum_{i=1}^{n} \alpha_i, \\ \textit{and (III)} \quad & -\left|\sum_{i=1}^{n} \alpha_i y_i\right| \quad \textit{(optional)} \end{aligned}$$

*subject to* $\forall_{i=1}^{n} : \alpha_i \geq 0$.

#### 3.2.5.2. The Multi-Objective EvoSVM

Individuals are again the real-valued vectors $\alpha = (\alpha_i, \ldots, \alpha_n)$. For mutation, we used the hybrid mutation discussed in the last section in order to get sparser solutions, i.e. solutions where many $\alpha_i$ are zero. Crossover probability is high (0.9). The individuals are initialized with 0 to further support sparsity. The maximum number of generations is 1000. The population size is 100. We use NSGA-II as the multi-objective selection scheme [36]. NSGA-II employs a selection technique which first sorts all individuals into levels of non-domination. Individuals from the first levels are added to the next generation until the desired population size is reached. Before adding individuals from the last possible level, this level is sorted with respect to the crowding distance in order to preserve diversity in the population.

### 3.2.6. Selecting a Solution from the Pareto Set

The first idea of supporting the user in selecting a final solution from the Pareto front might be to just calculate the first objective in its original form and check which indi-

| Data Set | $n$ | $m$ | Source | $\sigma$ | Default |
|---|---|---|---|---|---|
| Spiral | 1000 | 2 | Synthetical | 1.000 | 50.00 |
| Checkerboard | 1000 | 2 | Synthetical | 1.000 | 50.00 |
| Sonar | 208 | 60 | UCI | 1.000 | 46.62 |
| Diabetes | 768 | 8 | UCI | 0.001 | 34.89 |
| Lupus | 87 | 3 | StatLib | 0.001 | 40.00 |
| Crabs | 200 | 7 | StatLib | 0.100 | 50.00 |

**Table 3.5.:** The evaluation data sets. $n$ is the number of data points, $m$ is the dimension of the input space. The kernel parameter $\sigma$ was optimized with a grid parameter search. The last column contains the default error, i. e. the error for always predicting the major class.

vidual provides the highest value for

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k\left(x_i, x_j\right).$$

The corresponding model is the maximum margin model for the given data set without respecting the training errors since the values $\alpha_i$ were not bounded during the optimization. Although this solution is interesting in its own, this model is often not the desired one.

Alternatively, one could use another pointer to where in the Pareto front one should search for the final solution. We suggest to keep a small hold-out set of the data points of size $k$ [110, 111, 112]. These $k$ data points were not part of the input training set and are not used by the learner during the multi-objective optimization. After the optimization has finished and the Pareto front for all objectives has been found, the learner is applied to all $k$ data points of the hold-out set. The prediction error for each individual is calculated with the binary loss

$$l(y, f(x)) = \begin{cases} 1 \text{ if } y \neq f(x) \\ 0 \text{ otherwise} \end{cases}$$

which leads to the error $Err_p$ for the learned decision function $f_p$ of the $p$-th individual of the Pareto set:

$$Err_p = \sum_{q=1}^{k} l\left(y_q, f_p(x_q)\right).$$

Plotting all errors $Err_p$ together with the training set errors results in another front which can be compared to the original Pareto front. The user should examine places where the training error and the generalization error are close together and should avoid

areas where the generalization performance is much worse than the achieved objectives. The plots of both fronts together are a powerful tool to control overfitting: displaying the effect of overfitting in the generalization performance plot for all possible models ease the selection of an optimal model without getting in danger of (too much) overfitting.

### 3.2.7. Experiments and Results

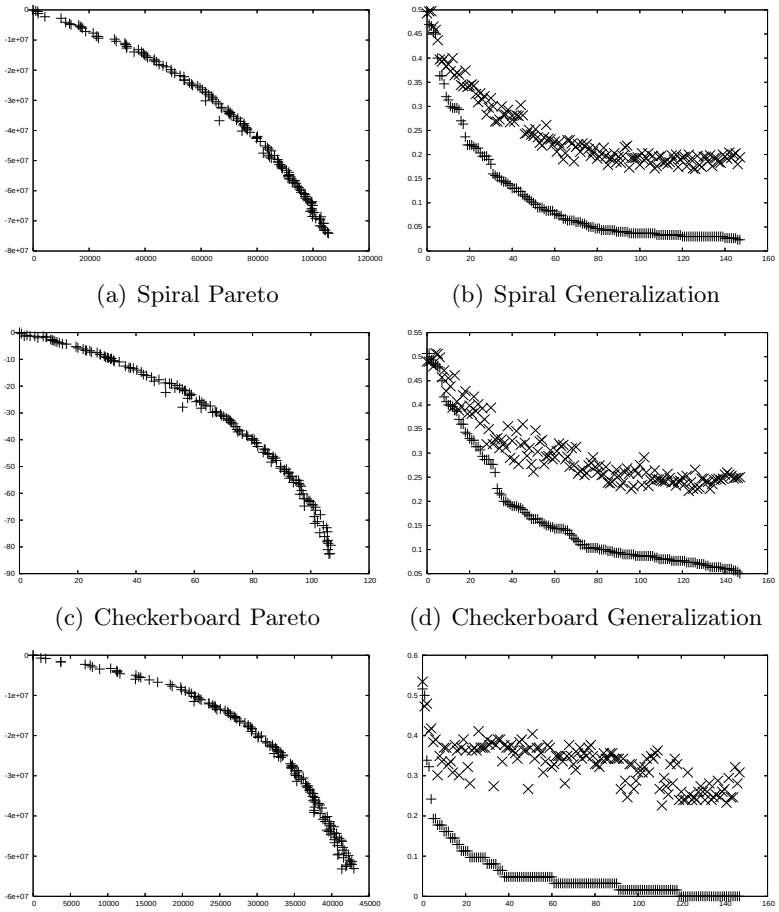
In the last experiment section, we already have seen that the original (single-objective) SVM problem can efficiently be solved by means of evolutionary algorithms. We now concentrate on the benefit of the transformation of the original SVM problem into an efficient multi-objective formalization by showing the Pareto fronts for several benchmark data sets. We use an RBF kernel for all SVMs and determine the best parameter value for $\sigma$ with a grid search parameter optimization. Possible parameters were 0.001, 0.01, 0.1, 1 and 10. A description of all data sets together with the optimal kernel parameter value $\sigma$ for each data set is given in Table 3.5. All experiments were performed with the machine learning environment RapidMiner [125][2], the new SVM implementation is called *EvoSVM* within this framework. The selection type must be set to *non-dominated sorting* in order to use the proposed solution for this operator.

Figures 3.2 and 3.3 show all results. *The left plot* for each data set shows the resulting Pareto front delivered by the multi-objective evolutionary SVM proposed in this paper. The y-axis denotes the first optimization objective from Section 3.2.5.1 (margin size) and the x-axis shows the second objective (training error). The third objective is omitted in the plots for the sake of simplicity. *The right plot* shows the prediction errors for the training set and a hold-out test set (cf. Section 3.2.6). The x-Axis simply denotes a counter over all Pareto-optimal solutions found during the optimization ordered by their training errors. The y-axis denotes the prediction error for the training (+) and testing ($\times$) data, i.e. on the hold-out set. The hold-out test set was a randomly sampled subset with a size of 20% of the given training set.

#### 3.2.7.1. Interpretation of the Pareto Fronts

First of all, it can clearly be seen that all Pareto fronts cover a wide range of solutions. Each point of a front denotes a vector $\alpha$ containing all Lagrange multipliers from which a normal vector $w$ and the bias $b$ can be calculated. Hence, each point denotes a specific SVM model. Some models emphasize a larger margin. Those models are located at higher regions of the y-axis (the margin size). Other models emphasize the minimization of the training error and are therefore located at the right parts of the x-axis. All fronts show a typical structure describing the trade-off between both criteria. It is not possible

[2] `http://www.rapidminer.com`

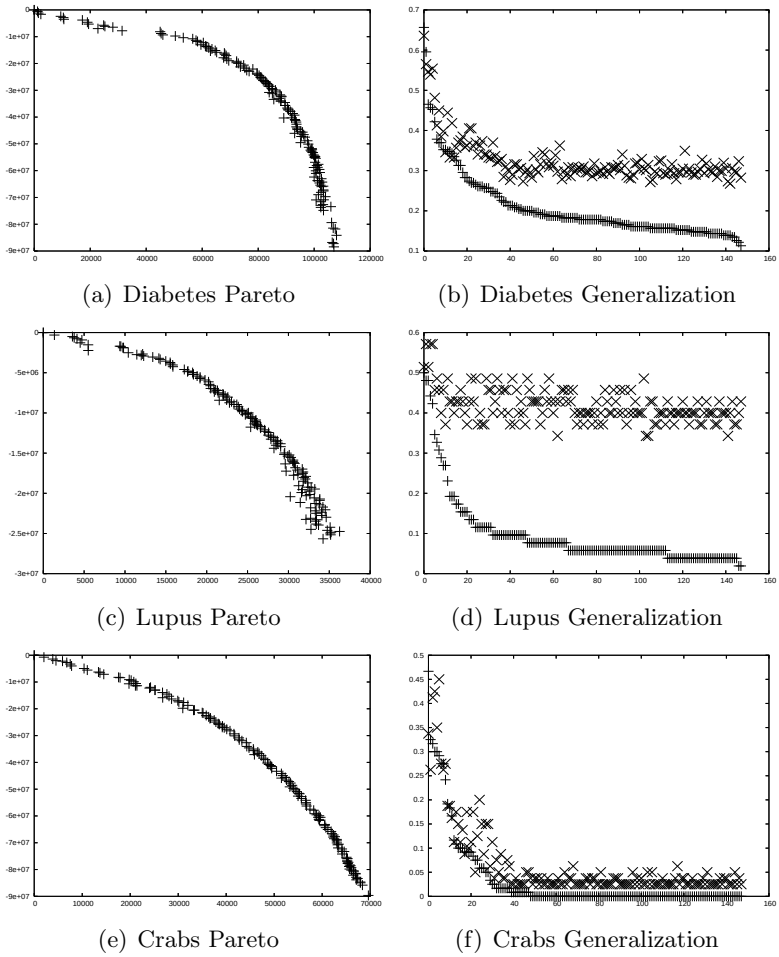


(a) Spiral Pareto (b) Spiral Generalization

(c) Checkerboard Pareto (d) Checkerboard Generalization

(e) Sonar Pareto (f) Sonar Generalization

**Figure 3.2.:** The left plot for each dataset shows the Pareto front delivered by the multi-objective EvoSVM proposed in this work (x: training error, y: margin size). The right plot shows the training (+) and testing (×) errors (on a hold-out set of 20%) for all individuals of the resulting Pareto fronts (x: Pareto solution counter, y: errors). Part 1 of the results.



(a) Diabetes Pareto (b) Diabetes Generalization

(c) Lupus Pareto (d) Lupus Generalization

(e) Crabs Pareto (f) Crabs Generalization

**Figure 3.3.:** The left plot for each dataset shows the Pareto front delivered by the multi-objective EvoSVM proposed in this work (x: training error, y: margin size). The right plot shows the training (+) and testing (×) errors (on a hold-out set of 20%) for all individuals of the resulting Pareto fronts (x: Pareto solution counter, y: errors). Part 2 of the results.

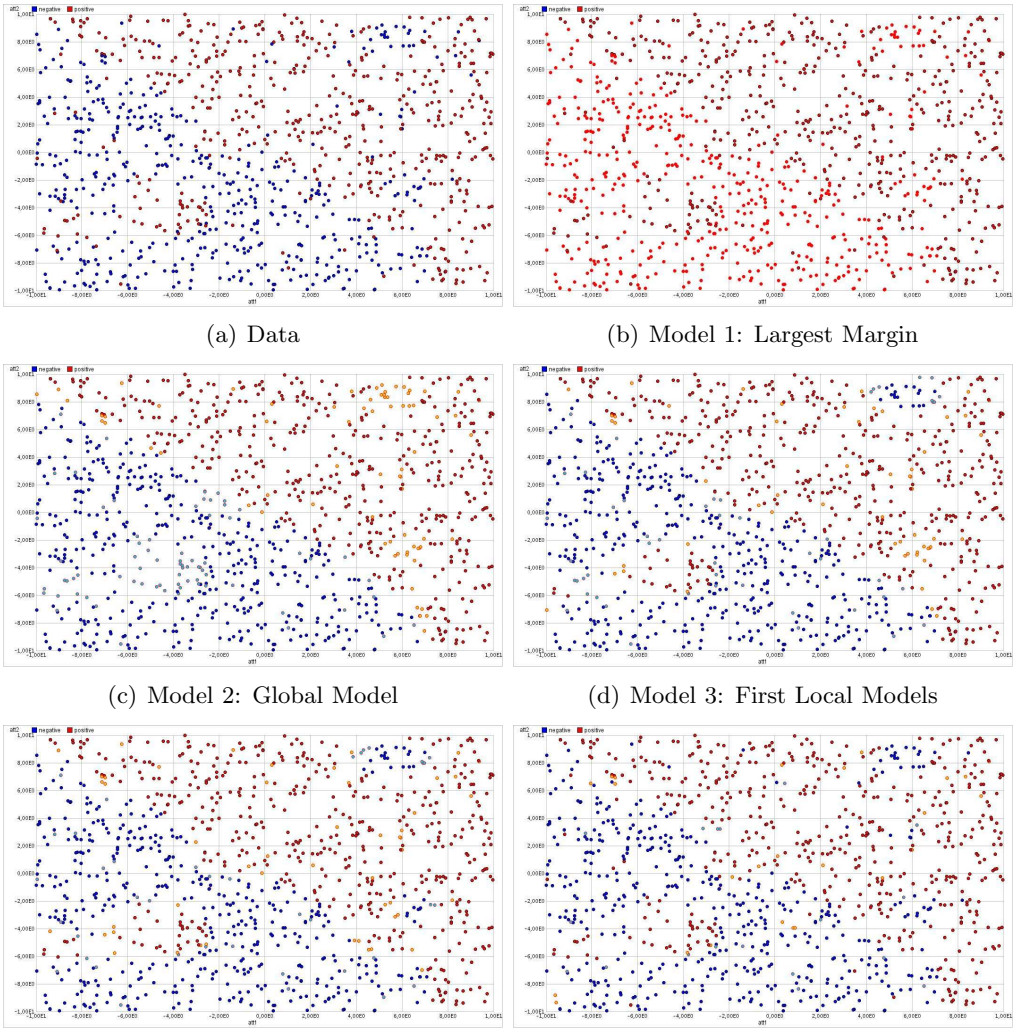
to find a model (a point) at the upper right corner of the plot since this point would dominate all others. It holds for each model / point of the final Pareto fronts that it cannot be optimized with respect to both, training error and margin size, but only for one of both criteria leading either to a model further to the right of the corresponding point or to model to the top compared to the corresponding point. Hence, such a movement along the Pareto front will never lead to a model moving to both directions at the same time. This of course follows the concept of Pareto domination.

It can also be seen that additional models located left from the most left point of the resulting plots or below the lowest point will not deliver better models. This can be easily observed for the Diabetes data set, where the front contains both a horizontal part on the left as well as a almost vertical part on the right. For the latter, it would hold that it does not make any sense to sacrifice more margin size since it is not possible to gain lower training errors. This can easily be seen in the plot but this is also something which would not be clear if an outer parameter optimization of C would have to be performed instead of using this multi-objective approach. In this manual optimization case, one could not even be sure that sensible regions for this parameter optimization would have been chosen. The multi-objective learning approach proposed in this thesis, however, does not suffer from this problem since the interesting trade-off region is automatically found by the proposed approach.

We can conclude that each Pareto front contains all sensible models in the trade-off region between the default classifier which classifies all points as the major class (the top left point) and total overfitting with zero training error (the bottom right point). All points in between correspond to a specific trade-off emphasizing either more the generalization capability by choosing larger margins or the adaption to the training data. Models with the same weight to both criteria are located in the middle region of the Pareto fronts.

We can perform a walk along such a Pareto front starting with the points in the upper left corner (large margin, high training error) to the lower left corner (overfitting). Figure 3.4 shows such a walk for a simple 2-dimensional data set consisting of a global model, several local models, and noise. Model 1 provides a margin so large that this model simply predicts the major class to the training data points. Model 2 is located further to the right of a resulting Pareto front and it corresponds to a model where the global model along the diagonal is already found. The local models, however, are still not detected by a point in this region of the Pareto front. This begins to happen for Model 3 which shows a model even further to the right in the resulting Pareto front. Here, the first local models are found. Model 4 finally demonstrates the best model identifying both the global model and the most important local models. This is a model which would be located in the region where training error and generalization error start to differ (see below). Finally, Model 5 already shows the clear overfitting to the noise data.

Until now, we have mainly discussed the left plots of Figure 3.2 and 3.3. The right plot for each Pareto front shows the prediction errors for the training set and a hold-out



(a) Data (b) Model 1: Largest Margin

(c) Model 2: Global Model (d) Model 3: First Local Models

(e) Model 4: Best (f) Model 5: Overfitting

**Figure 3.4.:** The first plot shows a simple 2-dimensional data set consisting of a global model (the hyperplane in the middle) together with some local models (the circular regions on the wrong side of the plane) and noise. The following pictures show the predictions of the different types of models of a resulting Pareto front. Model 1 provides the largest margin, it is actually so large that it is located beside the complete training data. Model 2 is a model corresponding to a point further on the right side where the global model is already found but the local models are not yet found. Model 3 shows a model further to the right where the first local models are found and Model 4 demonstrates the best model identifying both the global model and the most important local models. Finally, Model 5 already shows the clear overfitting to the noise data.

test set (cf. Section 3.2.6). The x-Axis simply denotes a counter over all Pareto-optimal solutions from the left plot ordered by their training errors. The y-axis denotes the mean prediction error for the training (+) and testing (×) data. From these plots, together with original Pareto fronts, we can conclude two things: first, there is no common region where the generalization error starts to get larger compared to the training error. In some cases (like for Spiral) this already happens in the upper left part of the Pareto front, in other cases (like for Crabs) the generalization error is hardly higher for all regions of the front. For that reason, guided multi-objective optimization schemes like desirability indices can hardly be used for this type of problem since it is not clear beforehand which part of the Pareto front could be more interesting.

The second conclusion from the combination of the Pareto front with the generalization plot is a hint in which regions of the Pareto front overfitting occurs, i.e. the training error is still minimized while the test error remains or get worse. You can detect this area in the right plots at places where the training error (+) and the testing error (×) diverge. Since the x-axis in the right plots correspond to a counter of solutions in the Pareto front, ordered by its training errors which corresponds to the x-axis in the left plot, you can find the interesting solutions in the Pareto front in the same area as on the right side. Instead of using a guided optimization scheme, we can use the generalization capability of the models in the resulting Pareto front to guide the analyst's selection of a final solution from the Pareto set.

#### 3.2.7.2. Multi-Objective Optimization vs. Multiple Single-Objective Runs

Please note that these types of plots could also be achieved for other learning schemes (e.g. usual SVM) by iteratively applying the learner for different parameter settings and produce the set of models in this way. The approach proposed in this paper has the advantage that all models are calculated in one single run which is by far less time-consuming.

Although the idea of statistical learning theory, i.e. taking the model complexity into account, is simple and appealing, current approaches did not make use of the inherent trade-off but demanded the definition of a weighting factor of the conflicting criteria from the user. The multi-objective evolutionary SVM proposed in this work is the first solution explicitely solving the basic problem of statistical learning theory.

The result of the proposed approach is a Pareto front in the space of training error vs. model complexity and gives interesting insights into the nature of the problem at hand. By using a hold-out data set as a test set for the resulting models we derive a second front showing the generalization error. Both, the Pareto front and the generalization error plot allows for a quick selection of the final solution from the Pareto front without the time-consuming optimization of a weighting factor. In fact, the selection can also be

automatically performed by selecting the solution generating the smallest generalization error.

Hence, the multi-objective evolutionary SVM approach leads to three advantages: first, it is no longer necessary to tune the SVM parameter C which weighs both conflicting criteria. This is a very time-consuming task for traditional SVM. Second, the shape and size of the Pareto front and the generalization plot on a hold-out set give interesting insights about the complexity of the learning task at hand. Finally, the user can actually see the point where overfitting occurs and can easily select a solution from the Pareto front best suiting his or her needs or let the method perform this selection.

CHAPTER 4

# Non-Convex Optimization for Statistical Learning

In the previous chapter, we have seen that evolutionary large margin methods are very competitive compared to their quadratic programming counterparts: they are at least as accurate as their traditional SVM in terms of generalization performance and they always outperform traditional approaches in terms of the original optimization problem. Furthermore, we demonstrated how the trade-off between training error and model complexity can be made explicit. We divided the optimization problem of SVM in two parts and transformed both parts into its dual form of its own. These transformations reduce the runtime for fitness evaluation and provide space for other well-known improvements like incorporating arbitrary kernel functions for non-linear classification tasks.

In this chapter, we will see that the proposed algorithms are also more generic than the existing traditional solutions since they will also work for non-positive semidefinite or indefinite kernel functions. We will state the problem of learning with indefinite kernels, discuss alternative approaches and compare the proposed algorithms with other approaches on several data sets.

## 4.1. Non-Positive Semidefinite Kernel Functions

In general, the primal optimization problem for large margin learning and SVMs without using a kernel function is a convex quadratic programming problem. This can directly be derived from the fact that the objective function itself is convex, and those points which satisfy the constraints also form a convex set. It holds that any linear constraint defines a convex set, and a set of $p$ simultaneous linear constraints defines the intersection of $p$

convex sets. This again is a convex set. The dual problem, which has to be maximized, hence can be considered to be a concave optimization problem.

It can be shown that if the kernel $k$, i. e. it's kernel matrix $K$, is positive definite, the objective function still is concave [22] after the dot product was replaced by the kernel function $k$. This means that we have a maximization problem with only one single global maximum. The optimization of the dual optimization problem is then very simple since we just need to search in the direction of higher values until we find the optimum. This type of problem is referred to as unimodal since only one single extremum exists.

We will now define what we mean by positive semidefinite kernel functions:

**Definition 4.1 (Positive Semidefinite)** *Let $X$ be a set of items. A kernel function $k$ with kernel matrix entries $K_{ij} = k(x_i, x_j)$ is called* POSITIVE SEMIDEFINITE *if the following applies*

$$c^* K c \geq 0 \text{ for all } c \in \mathbb{C}^n$$

*where $c^*$ is the conjugate transpose of $c$.*

A positive semidefinite kernel function will lead to a concave optimization problem, i.e. to an optimization problem having a global unique maximum which usually can be found by means of a gradient descent. However, in some cases a specialized kernel function must be used to measure the similarity between data points which is not positive definite, sometimes not even positive semidefinite. While positive definite kernels – just as the regular dot product – resemble a similarity measure, these non-positive semidefinite kernels (or indefinite kernels) can be considered as a (partial) distance measure.

One may ask why a solution for non-positive semidefinite kernels would be interesting at all. There are several reasons for studying the effect of non-PSD kernel functions on the optimization problem[1]. First, the test for positive semi-definiteness can be a challenging task for new kernel functions which often cannot be solved by a practitioner. Second, some kernel functions are interesting in spite that it can be shown that they are not positive semidefinite, e.g. the sigmoid kernel function $k(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle - \delta)$ of neural networks or a fractional power polynomial [24, 168]. Third, promising empirical results were reported for such non-PSD or indefinite kernels [96]. Unfortunately, the implementations in those approaches either work on the primal problem, which is hardly feasible for real-word problems, or they are numerically unstable and do not guarantee to find an optimum in all cases. Finally, several approaches of learning the kernel function were proposed where the result not necessarily must again be positive semidefinite even if only definite kernel functions were used as base functions [100].

Traditional SVMs are hardly applicable for indefinite kernel functions since the used optimization techniques based on quadratic programming can only solve problems with

[1] For a deeper discussion of the applications of non-PSD kernels see [140].

one single global optimum. If the optimization problem is multi-modal, i.e. the objective function has several extrema, they will not be able to find satisfying solutions at all. This is the often case for kernel functions which are not positive semidefinite. Most traditional implementations do not even terminate for these kernel functions [61]. A more generic optimization scheme should allow such non-positive kernels without the need for omitting the more efficient dual optimization problem as it was suggested in [140].

Before we discuss former approaches to learn SVM functions for such non-PSD kernels, we state some of the most important non-positive semidefinite kernel functions for two instances $x_i$ and $x_j$:

**Definition 4.2 (Epanechnikov Kernel)** *The* Epanechnikov Kernel *is defined as*

$$\left(1 - \frac{||x_i - x_j||^2}{\sigma}\right)^d \text{ for } \frac{||x_i - x_j||^2}{\sigma} \leq 1.$$

**Definition 4.3 (Gaussian Combination Kernel)** *The* Gaussian Combination Kernel *is defined as*

$$\exp\left(\frac{-||x_i - x_j||^2}{\sigma_1}\right) + \exp\left(\frac{-||x_i - x_j||^2}{\sigma_2}\right) - \exp\left(\frac{-||x_i - x_j||^2}{\sigma_3}\right).$$

**Definition 4.4 (Multiquadric Kernel)** *The* Multiquadric Kernel *is defined as*

$$\sqrt{\frac{||x_i - x_j||^2}{\sigma} + c^2}.$$

Figure 4.1 shows 2D-plots of these kernel function together with their 20 main Eigenvalues. It can clearly be seen that the Eigenvalues are partly negative and that the corresponding kernel functions resemble a distance function instead of a similarity function in some regions.

### 4.1.1. The Relevance Vector Machine: A Kernel Method for Indefinite Kernel Functions

Traditional SVM based on (Quasi-)Newton optimization techniques are only capable of solving concave optimization problems which can only be guaranteed for positive semidefinite kernel functions. If non-PSD or indefinite kernels should be used, those learning schemes will get stucked in the first local extrema and will not deliver the global optimum. In order to overcome numerical instabilities which often occur even for PSD kernels, most modern SVM implementations additionally contain heuristics to skip small degressions of the objective functions – but this makes things even worse since it

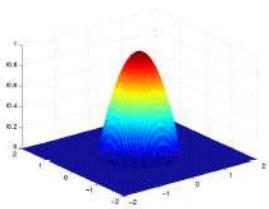
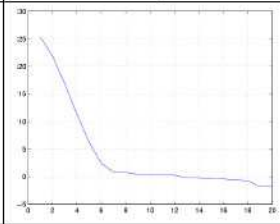
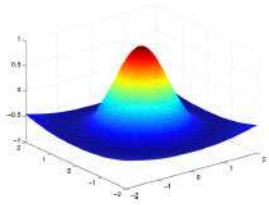
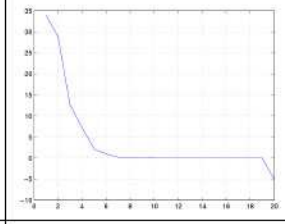
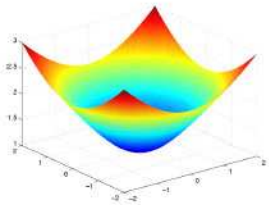
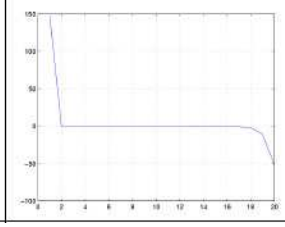
| Kernel | 2D kernel | 20 main Eigenvalues |
|---|---|---|
| Epanechnikov kernel<br>$\left(1 - \frac{\|s-t\|^2}{\sigma}\right)^p$, for $\frac{\|s-t\|^2}{\sigma} \leqslant 1$ |  |  |
| Gaussian Combination<br>$\exp\left(\frac{-\|s-t\|^2}{\sigma_1}\right) + \exp\left(\frac{-\|s-t\|^2}{\sigma_2}\right)$<br>$- \exp\left(\frac{-\|s-t\|^2}{\sigma_3}\right)$ |  |  |
| Multiquadric kernel<br>$\sqrt{\frac{\|s-t\|^2}{\sigma} + c^2}$ |  |  |

**Figure 4.1.:** 2D-plots of some known non-positive semidefinite kernel functions together with their 20 main Eigenvalues. These Eigenvalues are partly negative. The variables $s$ and $t$ represent data points $x$ and $p$, $c$ and $\sigma$ are parameters used for fine-tuning the kernel functions. Source: [140].

was reported that the algorithms no longer terminate for non-PSD kernels on several analysis problems.

Since learning with non-PSD kernels is much harder than learning with PSD kernels from an optimization point of view, we will discuss an alternative approach derived from sparse Bayesian learning in this section. The *Relevance Vector Machine* (RVM) [179] produces sparse solutions using an improper hierarchical prior and optimizing over hyper parameters. One might define these hyper parameters as the parameters of a Gaussian Process' covariance function. The main purpose of RVM is the direct prediction of probabilistic values instead of crisp classifications. For SVM, complex post processing steps like Platt's scaling [144] must be performed in order to derive probabilistic predictions. Even then, examples could be constructed where such post processing approaches will fail. In contrast, the RVM is based on a probabilistic framework which directly allows to predict the correct values. During the last years, additional research was done to further improve these probabilistic RVM predictions [148]. The main advantage to us, however, is the fact that Relevance Vector Machines depend on basis functions which do not need to be positive semidefinite.

The RVM aims at a sparse linear model, i. e. a linear combination of a set of $l$ basis functions $\Phi_i(x)$:

$$f(x) = \sum_{i=1}^{l} \alpha_i \Phi_i(x).$$

This function is used for calculating regression predictions and can easily be transformed into classification learning by thresholding. Please note that a set of $l$ different base extensions $\Phi_i$ is used, each of them weighted by the factor $\alpha_i$. Hence, the output is a linearly weighted sum of $l$, generally nonlinear and fixed, basis functions $\Phi_i$. The task then become to find a good set of weights $\alpha_i$. One can easily see that the SVM is a special case of the RVM where a basis function $\Phi_i$ is simply defined for each support vector $x_i$ and the prediction function hence becomes

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b$$

with $l$ as the number of support vectors and $b$ as the offset or bias which could also be considered as a constant base function and would then become part of a sum with $l + 1$ base functions.

We concentrate on discussing the regression model for out training data $T \subset X \times Y$. We follow the standard probabilistic formulation and assume that the examples are samples from a generating model $f$ with additive noise:

$$y_i = f(x_i) + \varepsilon_i.$$

Here, the $\varepsilon_i$ are independent samples from some noise process which is assumed to be a Gaussian with mean 0 and variance $\sigma^2$. Thus, we can write $P(y_i|x_i)$ as $N\left(y_i|f(x_i), \sigma^2\right)$ which describes a Gaussian distribution for the true label with the function value $f(x_i)$ as mean and variance $\sigma^2$. We assume the function $f$ to be defined like in the SVM case as a sum over the kernel functions of selected training points and due to the independence of $y_i$ we can derive the likelihood of the complete data set as

$$P(y|\alpha, \sigma^2) = (2\pi\sigma^2)^{-n/2} \cdot e^{\frac{1}{2\pi\sigma^2}||y-\Phi\alpha||^2}$$

where $y$ is the vector of all $n$ labels, $\alpha$ is the vector of all $n$ weights, and $\Phi$ is the so-called design matrix consisting of the vector of all $n$ basis functions $\Phi(x_i)$ where each $\Phi(x_i)$ again is based on the training data kernel functions. This design matrix is therefore the same as the well known kernel matrix but also contains a constant column consisting of the constant 1.

If we allow $n$ parameters, the likelihood maximization would probably lead to overfitting and hence we have to add some regularization or a similar complexity penalty term. We again use a Bayesian point of view and constrain the parameters by defining a *prior* probability distribution over them.

We encode the preference to smoother (less complex) functions by choosing a zero-mean Gaussian as the prior distribution for the values $\alpha_i$ which leads to

$$P(\alpha_i|a_i) = N(\alpha_i|0, a_i^{-1}).$$

with the so-called hyper parameters $a_i$. Importantly, there is an individual hyper parameter $a_i$ associated independently with every weight $\alpha_i$ moderating the strength of the prior thereon.

Relevance vector learning becomes the search for the hyper parameter posterior mode, i.e. the maximization of the marginal likelihood

$$P(y|X, a, \sigma^2) = \int P(y|X, \alpha, \sigma^2)P(\alpha|a)d\alpha$$

either by direct optimization (e.g. by conjugate gradients) or faster by means of an EM-like algorithm. Sparsity is achieved since most of the parameters $a_i$ usually become infinite during the optimization. The corresponding basis functions are then removed. The remaining basis functions on training vectors are called *relevance vectors*. Please refer to [179] for more details, especially for the necessary approximations to calculate the marginal likelihood given above.

It should be pointed out that it is a common choice to use the same basis function $\Phi(x)$ for all training examples, often a RBF kernel with fixed parameter $\sigma$. There is, however, no need for such a fixed basis function and indeed it is even possible to use basis functions which would not lead to a positive semidefinite kernel matrix. Therefore, we use an RVM in our comparison experiments for non-positive semidefinite kernel functions.

## 4.2. Experiments and Results

In this section, we evaluate the proposed evolutionary optimization SVM with respect to the classification performance when non-positive definite kernel functions are used. We compare our implementation to the quadratic programming approaches usually applied to large margin problems. The experiments demonstrate that evolutionary SVMs are the first feasible solution for indefinite kernel functions.

In order to compare the evolutionary SVM described in this paper with standard SVM implementations, we also applied two other SVM on all data sets. Both SVM use a slightly different optimization technique based on quadratic programming. The used implementations again were *mySVM* [160] and *LibSVM* [25]. We also applied an RVM learner [179] on all data sets which should provide better results for this type of kernel functions. All experiments were performed with the machine learning environment RapidMiner [125][2].

### 4.2.1. Evolutionary Computation for Non-Convex Optimization

We used the EvoSVM as it was described before. This solution employs evolution strategies instead of the usual quadratic programming approaches. The used optimization problem is the dual problem for non-linear separation of non-separable data developed in the previous chapters (Problem 2.5). Individuals are the real-valued vectors $\alpha$ and mutation is performed by adding a Gaussian distributed random variable with standard deviation $C/10$. In addition, a variance adaptation is conducted during optimization (1/5 rule [150]). Crossover probability is high (0.9). We use tournament selection with a tournament size of 0.25 multiplied by the population size. The initial individuals are random vectors with $0 \leq \alpha_i \leq C$. The population size is 10. The maximum number of generations is 10000 and the optimization is terminated if no improvement occurred during the last 10 generations. We increased the number of generations for the stopping criterion in order to respect the multi-modal type of this optimization problem.

### 4.2.2. Data Sets

We apply the discussed EvoSVM as well as the other SVM implementations on two synthetical and six real-world benchmark data sets. The data sets were also used in the previous chapter and are only shortly described here. The data set Spiral consists of two intertwingling spirals of different classes. For checkerboard, the data set consists of two classes layed out in a $8 \times 8$ checkerboard. In addition, we use six benchmark data sets from the UCI machine learning repository [136] and the StatLib data set library

[2] `http://www.rapidminer.com/`

| Data Set | $n$ | $m$ | Source | $\sigma$ | $d$ | Default |
|---|---|---|---|---|---|---|
| Spiral | 500 | 2 | Synthetical | 11.40 | 8.80 | 50.00 |
| Checkerboard | 300 | 2 | Synthetical | 9.75 | 3.00 | 50.00 |
| Liver | 346 | 6 | UCI | 218.61 | 4.74 | 42.03 |
| Sonar | 208 | 60 | UCI | 5.23 | 9.00 | 46.62 |
| Diabetes | 768 | 8 | UCI | 998.99 | 2.56 | 34.89 |
| Lawsuit | 264 | 4 | StatLib | 195.56 | 8.30 | 7.17 |
| Lupus | 87 | 3 | StatLib | 896.28 | 6.27 | 40.00 |
| Crabs | 200 | 7 | StatLib | 29.37 | 2.61 | 50.00 |

**Table 4.1.:** The evaluation data sets. $n$ is the number of data points, $m$ is the dimension of the input space. The kernel parameters $\sigma$ and $d$ were optimized for the comparison SVM learner *mySVM*. The last column contains the default error, i. e. the error for always predicting the major class.

[173]. Again, we chose those data sets because they already define a binary classification task, consist of real-valued numbers only and do not contain missing values. Therefore, we did not need to perform additional preprocessing steps which might introduce some bias. The properties of all data sets are summarized in Table 4.1. The default error corresponds to the error a lazy default classifier would make by always predicting the major class. Classifiers must produce lower error rates in order to learn at all instead of just guessing.

For the non-PSD experiments, we used an Epanechnikov kernel function with parameters $\sigma$ and $d$. The optimized values are given in Table 4.1.

### 4.2.3. Comparison for Non-positive Kernels

We compared the different implementations and a relevance vector machine on all data sets for a non-positive kernel function. The Epanechnikov kernel was used for this purpose with kernel parameters as given in Table 4.1. These parameters are optimized for the SVM implementation mySVM in order to ensure fair comparisons. Table 4.2 summarizes the results, again for $C = 1$ which is the heuristically best value for the Epanechnikov kernel according to the meta-parameter heuristic discussed in the previous Chapter.

It should be noticed that the runtime of the Relevance Vector Machine implementation is not feasible for real-world applications. Even on the comparatively small data sets the RVM needs more than 60 days in some of the cases for all of the twenty learning runs. Although it is very competitive compared to the other SVM approaches with respect to

the prediction error, the RVM was unfortunately not able to deliver the best result for any of the data sets.

It can also be noticed that the EvoSVM variant frequently outperforms the competitors on all data sets. While the LibSVM was hardly able to generate models better than the default model, the mySVM delivers surprisingly good predictions even for the non-PSD kernel function. Of course, the kernel parameters were optimized exactly for this learning scheme like it was mentioned above. Additionally, the mySVM offer much more heuristics to cope with numerical instabilities often occurring in SVM optimizations and is hence better capable of optimizing in multi-modal settings at least to a certain degree. However, especially for the data sets Spiral, Checkerboard, Liver, Sonar, and Crabs the results of the EvoSVM are significantly better than those of all competitors.

| Data Set | Algorithm | Error | T[s] |
|---|---|---|---|
| Spiral | EvoSVM | **16.00 ± 4.38** | 4 |
| | mySVM | 46.20 ± 6.56 | 6 |
| | LibSVM | 51.00 ± 1.00 | 2 |
| | RVM | 19.80 ± 3.16 | 2942468 |
| Checkerboard | EvoSVM | **23.00 ± 3.56** | 1 |
| | mySVM | 40.33 ± 4.99 | 1 |
| | LibSVM | 45.33 ± 1.63 | 1 |
| | RVM | 38.00 ± 7.48 | 336131 |
| Liver | EvoSVM | **31.29 ± 5.90** | 3 |
| | mySVM | 38.32 ± 7.84 | 1 |
| | LibSVM | 42.03 ± 1.46 | 1 |
| | RVM | 40.58 ± 1.75 | 203093 |
| Sonar | EvoSVM | **15.40 ± 3.66** | 3 |
| | mySVM | 46.62 ± 1.62 | 2 |
| | LibSVM | 46.62 ± 1.62 | 2 |
| | RVM | 29.79 ± 7.29 | 105844 |
| Diabetes | EvoSVM | 32.68 ± 4.77 | 3 |
| | mySVM | 32.54 ± 2.82 | 4 |
| | LibSVM | 34.89 ± 0.34 | 6 |
| | RVM | 32.76 ± 1.89 | 5702491 |
| Lawsuit | EvoSVM | 29.89 ± 10.71 | 2 |
| | mySVM | 30.93 ± 10.66 | 2 |
| | LibSVM | 36.72 ± 2.01 | 2 |
| | RVM | 37.89 ± 3.83 | 106697 |
| Lupus | EvoSVM | 31.81 ± 11.64 | 2 |
| | mySVM | 34.44 ± 18.51 | 2 |
| | LibSVM | 40.00 ± 6.33 | 1 |
| | RVM | 33.06 ± 10.07 | 5805 |
| Crabs | EvoSVM | **4.00 ± 4.90** | 1 |
| | mySVM | 11.00 ± 7.35 | 1 |
| | LibSVM | 50.00 ± 0.00 | 1 |
| | RVM | 13.50 ± 7.43 | 150324 |

**Table 4.2.:** Comparison of the different implementations with regard to the classification error (the lower the better) for a non-positive semidefinite kernel function (Epanechnikov). The results are obtained by a 20-fold cross validation, the time is the cumulated time for all runs. Bold fonts mark significantly better results on a 1% confidence level.

CHAPTER 5

# Transductive Learning: Non-Convex and Multi-Objective

Beside the inherent advantages of evolutionary algorithms (e. g. parallelization, multi-objective optimization of training error and capacity) it is now also possible to use non-positive semidefinite or indefinite kernel functions which would lead to unsolvable problems for other optimization techniques. With one exception, those traditional SVM were not able to deliver significantly better results compared to the default error and the RVM is not feasible for large real-world problems. As the experiments have shown, an SVM based on evolutionary computation is the first practical solution for this type of problem delivering considerable better results in much shorter times.

We will now deal with another practically relevant learning problem, namely transductive learning, which also leads to a non-convex optimization problem. We will discuss how transductive learning can be defined as a multi-objective optimization problem and how we can solve it with help of the EvoSVM.

## 5.1. Motivation of Transductive Learning

Usually in supervised learning, we try to find a general decision function for a given learning task. This type of problem is called *inductive inference* and it is performed on a limited sample of the input data. After the decision function was found, this general rule can be applied to new data or be evaluated on available test data not shown during training. We have already discussed that we assume the same probability distribution $P(X, Y)$ for the training and test set. This is a basic assumption for inductive learning in statistical learning theory.

For many learning problems, however, the unlabeled test set can be much larger than the labeled training set. This may happen for several reasons, such as cost of labeling or even as a result of some inherent limitations of the problem. For such a problem, an alternative approach is to obtain specific rules for the given set of examples that we wish to classify, i.e. the points from the unlabeled test set, instead of finding a general rule for the whole input space. Vapnik called this – as he said easier – learning problem *transductive inference*.

Although transductive learning is a general concept that can be applied to any learning scheme, it has been applied most successfully to Support Vector Machines [73]. The implicit margin maximization formulation of SVMs is used as an additional cost function that does not depend on labeling and can be applied to the unlabeled test set to optimize the general performance of the learning machine. This turns the labels of each of the instances of the test set into variables of the optimization problem. The optimal solution is the one that minimizes the error of the training set and maximizes the margin of both the training and test sets.

The well known Transductive SVM (TSVM) algorithm proposed by Joachims [73] relies on a greedy search of labels of the test set by switching the labels with largest errors from one class to the other. This algorithm

1. learns a model on the training data,
2. applies the model on the test data,
3. chooses two label assignments $y_k^*$ and $y_l^*$ which must correspond to positive and negative classes and cause maximal errors and switch their labels,
4. stops if the objective function is no longer optimized.

This procedure will probably not lead to globally optimal solutions since the underlying optimization problem is non-convex and the TSVM algorithm will converge to local extrema. Other optimization heuristics were proposed in order to overcome the risk for local extrema [28] but still the resulting algorithms are not able to find the global optimum for large training and test sets. Therefore, an evolutionary approach was proposed for solving the dual optimization problem of transductive SVM [166] but only the single-objective optimization problem was solved in this work. Finally, another problem formulation for transductive learning has been recently proposed [43] which is no longer based on the large margin principle of the Support Vector Machines but on the concept of large volumes which are only loosely related to SVMs. This paradigm change actually is very close to that proposed in this chapter since several criteria are taken into account. But again the proposed optimization problem is not solved with true multi-objective optimization and users have to define their preference before the optimization has started.

In this chapter, we will define the transductive optimization problem as a multi-objective optimization problem and extend the existing work by transforming each objective into the dual form on its own. We will also discuss the connection between multi-objective transductive SVM and supervised learning, semi-supervised learning, and clustering.

### 5.1.1. Problem Definition

Let $X \in \mathbb{R}^m$ be a real-valued vector of random variables. Let $Y \in \mathbb{R}$ be another random variable. $X$ and $Y$ obey a fixed but unknown probability distribution $P(X, Y)$. We again try to find a function $f(x, \gamma)$ which predicts the value of $Y$ for a given input $x \in X$. In contrast to the inductive supervised learning setting discussed before, we now have two different data sets. First, a set of training data points

$$T = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq X \times Y$$

and second, a set of test data points, i.e. a set of observations without label information

$$T^* = \{x_1^*, \ldots, x_{n^*}^*\} \subseteq X.$$

In the following, the $*$ will always mark variables for the unlabeled test data set.

As we have done before, we again constrain the number of possible values of $Y$ to 2, without loss of generality these values should be $-1$ and $+1$. And also again, we are searching for a separating hyperplane

$$H = \{h | \langle w, h \rangle + b = 0\},$$

where $w$ is normal to the hyperplane, $|b|/||w||$ is the perpendicular distance of the hyperplane to the origin (offset or bias), and $||w||$ is the Euclidean norm of $w$. This hyperplane might also separate the data points in some high-dimensional non-linearly transformed feature space.

Additionally to the margin maximization and training error minimization that we have already defined before, we now also have to maximize the margin for the unlabeled test data and minimize the number of errors for possible label assignments. This leads to the optimization problem:

**Problem 5.1 (Primal Transductive SVM Problem)** *The primal transductive SVM optimization problem is defined as*

$$
\begin{aligned}
\textit{minimize } & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i + C^*\sum_{k=1}^{n^*}\xi_k^* \\
\textit{subject to } & \forall_{i=1}^{n} : y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i, \\
& \forall_{k=1}^{n^*} : y_k^*\left(\langle w, x_k^*\rangle + b\right) \geq 1 - \xi_k^*, \\
& \forall_{i=1}^{n} : \xi_i \geq 0, \\
& \forall_{k=1}^{n^*} : \xi_k^* \geq 0.
\end{aligned}
$$

Please note that we optimize this function not only over the variables $w$, $b$ and $\xi_i$ as before but also over the new variables $\xi_k^*$ and, more important, over all possible test label assignments $y_k^*$. $C$ defines the influence of the training error and $C^*$ defines the influence of the possible test error using the current test label assignments $y_k^*$.

This optimization problem can be transformed into a dual form but this no longer leads to a convex optimization problem since the different possibilities for the label variables $y_k^*$ must be considered. In addition, the search space grows quickly since in the worst case all $2^{n^*}$ combinations of test label assignments must be tested. In the following, we will transform the different parts of the optimization function into their dual form in order to get a multi-objective optimization problem. This non-convex multi-objective optimization problem is then solved by means of the EvoSVM described in the previous chapters with an extended individual representation containing not only the Lagrange multipliers $\alpha_i$ for the training data but also those $\alpha_k^*$ for the test data together with the possible test label assignments $y_k^*$.

## 5.2. Dual Optimization Problems for Transductive SVM

We will now state the different parts of the primal problem defined above and transform all three objectives into their dual form in order to allow the efficient optimization of the problems including the usage of kernel functions. The weighting factors $C$ and $C^*$ can then be omitted. This leads to the following optimization problems. Problem one is the maximization of the margin:

$$
\begin{aligned}
\text{minimize } & \frac{1}{2}||w||^2 \\
\text{subject to } & \forall_{i=1}^{n} : y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i, \\
& \forall_{k=1}^{n^*} : y_k^*\left(\langle w, x_k^*\rangle + b\right) \geq 1 - \xi_k^*, \\
& \forall_{i=1}^{n} : \xi_i \geq 0, \\
& \forall_{k=1}^{n^*} : \xi_k^* \geq 0,
\end{aligned}
$$

problem two is the minimization of the error on the training data:

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{n} \xi_i \\ \text{subject to } & \forall_{i=1}^{n} : y_i \left( \langle w, x_i \rangle + b \right) \geq 1 - \xi_i, \\ & \forall_{i=1}^{n} : \xi_i \geq 0, \end{aligned}$$

and problem three is the minimization of the error on the test data:

$$\begin{aligned} \text{minimize } & \sum_{k=1}^{n^*} \xi_k^* \\ \text{subject to } & \forall_{k=1}^{n^*} : y_k^* \left( \langle w, x_k^* \rangle + b \right) \geq 1 - \xi_k^*, \\ & \forall_{k=1}^{n^*} : \xi_k^* \geq 0. \end{aligned}$$

Please note that we can omit the unnecessary constraints from problem two and three since they do not constrain the optimization function at all. This is of course not the case for the first optimization function. In the next paragraphs, we will transform all objectives on their own which will finally lead to a 3-dimensional multi-objective optimization problem.

### 5.2.1. First Objective: Maximizing the Margin

We first concentrate on the optimization of the margin for both the training data and the test data:

$$\begin{aligned} \text{minimize } & \frac{1}{2}||w||^2 \\ \text{subject to } & \forall_{i=1}^{n} : y_i \left( \langle w, x_i \rangle + b \right) \geq 1 - \xi_i, \\ & \forall_{k=1}^{n^*} : y_k^* \left( \langle w, x_k^* \rangle + b \right) \geq 1 - \xi_k^*, \\ & \forall_{i=1}^{n} : \xi_i \geq 0, \\ & \forall_{k=1}^{n^*} : \xi_k^* \geq 0. \end{aligned}$$

Before we can transform this objective function into the dual form we need to prove the following lemma. This will allow us the introduction of kernel functions into transductive SVM as we have seen them before for inductive SVM.

**Lemma 5.1 (Squared Sum of Weighted Label Products)** *The squared sum of weighted product of labels and examples can be written in terms of dot products since the following holds:*

$$\left( \sum_{i=1}^{n} \alpha_i y_i x_i \right)^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle .$$

*Proof.* We prove the lemma by dividing the sums into the individual terms and applying some basic transformations:

$$\begin{aligned}
&\left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)^2 \\
=&\left(\alpha_1 y_1 x_1 + \sum_{i=2}^{n} \alpha_i y_i x_i\right)^2 \\
=&\alpha_1^2 y_1^2 x_1^2 + 2\alpha_1 y_1 x_1 \sum_{i=2}^{n} \alpha_i y_i x_i + \left(\sum_{i=2}^{n} \alpha_i y_i x_i\right)^2 \\
=&\alpha_1^2 y_1^2 x_1^2 + 2\alpha_1 y_1 x_1 \sum_{i=2}^{n} \alpha_i y_i x_i + \alpha_2^2 y_2^2 x_2^2 + 2\alpha_2 y_2 x_2 \sum_{i=3}^{n} \alpha_i y_i x_i + \left(\sum_{i=3}^{n} \alpha_i y_i x_i\right)^2 \\
=&\ldots
\end{aligned}$$

We can now build new sums from all terms:

$$\begin{aligned}
&\sum_{i=1}^{n} \alpha_i^2 y_i^2 x_i^2 + \sum_{i=1}^{n} 2\alpha_i y_i x_i \sum_{j=i+1}^{n} \alpha_j y_j x_j \\
=&\sum_{i=1}^{n} \alpha_i^2 y_i^2 x_i^2 + 2\sum_{i=1}^{n} \sum_{j=i+1}^{n} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle \\
=&\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle .
\end{aligned}$$

The last transformation is derived from the fact that the double sum in the second line describes the upper (and the symmetrical lower) part of the sum matrix over all examples and the first term describes the diagonal of this matrix. □

We are now able to state the dual form of the optimization problem corresponding to the training and test set margin maximization. The result is one of the parts of an objective function which will be later used for a multi-objective optimization approach:

**Theorem 5.1 (Dual Form of Transductive Margin Maximization)** *The dual form of the first objective (maximize margin) for the multi-objective transductive Support Vector Machine (SVM) is given as the following constrained optimization problem:*

$$
\begin{aligned}
\textit{maximize} \quad & \sum_{i=1}^{n} \alpha_i + \\
& \sum_{k=1}^{n^*} \alpha_k^* - \\
& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \\
& \frac{1}{2} \sum_{k=1}^{n^*} \sum_{l=1}^{n^*} \alpha_k^* \alpha_l^* y_k^* y_l^* \langle x_k^*, x_l^* \rangle - \\
& \sum_{i=1}^{n} \sum_{k=1}^{n^*} \alpha_i \alpha_k^* y_i y_k^* \langle x_i, x_k^* \rangle \\
\textit{subject to} \quad & \forall_{i=1}^{n} : \alpha_i \geq 0, \\
& \forall_{k=1}^{n^*} : \alpha_k^* \geq 0, \\
& \sum_{i=1}^{n} \alpha_i y_i + \sum_{k=1}^{n^*} \alpha_k^* y_k^* = 0.
\end{aligned}
$$

*Proof.* We introduce positive Lagrange multipliers into the primal objective function stated above. We need multipliers $\alpha$ for the first set of training set inequality constraints, multipliers $\alpha^*$ for the first set of test set inequality constraints, multipliers $\beta$ for the second set of of training set inequality constraints, and multipliers $\beta^*$ for the second set of test set inequality constraints:

$$
\begin{aligned}
L_p^{(1)}(w, b, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*, y^*) = & \frac{1}{2} ||w||^2 - \\
& \sum_{i=1}^{n} \alpha_i \left( y_i \left( \langle w, x_i \rangle + b \right) + \xi_i - 1 \right) - \\
& \sum_{k=1}^{n^*} \alpha_k^* \left( y_k^* \left( \langle w, x_k^* \rangle + b \right) + \xi_k^* - 1 \right) - \\
& \sum_{i=1}^{n} \beta_i \xi_i - \sum_{k=1}^{n^*} \beta_k^* \xi_k^*
\end{aligned}
$$

The derivatives of $L_p^{(1)}$ have to be 0 for the minimal point:

$$
\frac{\partial L_p^{(1)}}{\partial w}(w, b, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*, y^*) = w - \sum_{i=1}^{n} \alpha_i y_i x_i - \sum_{k=1}^{n^*} \alpha_k^* y_k^* x_k^* = 0,
$$

$$\frac{\partial L_p^{(1)}}{\partial b}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) = \sum_{i=1}^{n}\alpha_i y_i + \sum_{k=1}^{n^*}\alpha_k^* y_k^* = 0,$$
$$\frac{\partial L_p^{(1)}}{\partial \xi_i}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) = -\alpha_i - \beta_i = 0,$$
$$\frac{\partial L_p^{(1)}}{\partial \xi_k^*}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) = -\alpha_k^* - \beta_k^* = 0.$$

Plugging the derivatives into the primal objective function $L_p^{(1)}$ delivers

$$\begin{aligned}
L_p^{(1)}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) =& \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i x_i\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{n^*}\alpha_i\alpha_k^* y_i y_k^* \langle x_i, x_k^*\rangle + \\
& \frac{1}{2}\left(\sum_{k=1}^{n^*}\alpha_k^* y_k^* x_k^*\right)^2 - \\
& \sum_{i=1}^{n}\alpha_i y_i \langle w, x_i\rangle - \sum_{i=1}^{n}\alpha_i y_i b - \sum_{i=1}^{n}\alpha_i\xi_i + \sum_{i=1}^{n}\alpha_i - \\
& \sum_{k=1}^{n^*}\alpha_k^* y_k^* \langle w, x_k^*\rangle - \sum_{k=1}^{n^*}\alpha_k^* y_k^* b - \sum_{k=1}^{n^*}\alpha_k^*\xi_k^* + \sum_{k=1}^{n^*}\alpha_k^* - \\
& \sum_{i=1}^{n}\beta_i\xi_i - \sum_{k=1}^{n^*}\beta_k^*\xi_k^* \\
\Leftrightarrow L_p^{(1)}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) =& \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i x_i\right)^2 + \frac{1}{2}\left(\sum_{k=1}^{n^*}\alpha_k^* y_k^* x_k^*\right)^2 + \\
& \sum_{i=1}^{n}\alpha_i + \sum_{k=1}^{n^*}\alpha_k^* - \\
& \sum_{i=1}^{n}\sum_{k=1}^{n^*}\alpha_i\alpha_k^* y_i y_k^* \langle x_i, x_k^*\rangle - \\
& \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \\
& \sum_{k=1}^{n^*}\sum_{l=1}^{n^*}\alpha_k^*\alpha_l^* y_k^* y_l^* \langle x_k^*, x_l^*\rangle .
\end{aligned}$$

We can now apply Lemma 5.1 and replace the squared sums by double sums including the inner product between all examples:

$$
\begin{aligned}
L_p^{(1)}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) =& \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \\
& \frac{1}{2}\sum_{k=1}^{n^*}\sum_{l=1}^{n^*}\alpha_k^*\alpha_l^* y_k^* y_l^* \langle x_k^*, x_l^*\rangle + \\
& \sum_{i=1}^{n}\alpha_i + \\
& \sum_{k=1}^{n^*}\alpha_k^* - \\
& \sum_{i=1}^{n}\sum_{k=1}^{n^*}\alpha_i\alpha_k^* y_i y_k^* \langle x_i, x_k^*\rangle - \\
& \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \\
& \sum_{k=1}^{n^*}\sum_{l=1}^{n^*}\alpha_k^*\alpha_l^* y_k^* y_l^* \langle x_k^*, x_l^*\rangle \\
\Leftrightarrow L_p^{(1)}(w,b,\xi,\xi^*,\alpha,\alpha^*,\beta,\beta^*,y^*) =& \sum_{i=1}^{n}\alpha_i + \\
& \sum_{k=1}^{n^*}\alpha_k^* - \\
& \sum_{i=1}^{n}\sum_{k=1}^{n^*}\alpha_i\alpha_k^* y_i y_k^* \langle x_i, x_k^*\rangle - \\
& \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \\
& \frac{1}{2}\sum_{k=1}^{n^*}\sum_{l=1}^{n^*}\alpha_k^*\alpha_l^* y_k^* y_l^* \langle x_k^*, x_l^*\rangle .
\end{aligned}
$$

The Wolfe dual must be maximized which leads to the formalization of the first objective of the multi-objective transductive SVM:

$$\begin{aligned}
\text{maximize } & \sum_{i=1}^{n} \alpha_i + \\
& \sum_{k=1}^{n^*} \alpha_k^* - \\
& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \\
& \frac{1}{2} \sum_{k=1}^{n^*} \sum_{l=1}^{n^*} \alpha_k^* \alpha_l^* y_k^* y_l^* \langle x_k^*, x_l^* \rangle - \\
& \sum_{i=1}^{n} \sum_{k=1}^{n^*} \alpha_i \alpha_k^* y_i y_k^* \langle x_i, x_k^* \rangle \\
\text{subject to } & \forall_{i=1}^{n} : \alpha_i \geq 0, \\
& \forall_{k=1}^{n^*} : \alpha_k^* \geq 0, \\
& \sum_{i=1}^{n} \alpha_i y_i + \sum_{k=1}^{n^*} \alpha_k^* y_k^* = 0.
\end{aligned}$$

Replacing the dot products by kernel functions $k$ concludes the proof. □

### 5.2.2. Second Objective: Minimizing the Training Error

Our next goal is to transform the second objective of the transductive SVM in its dual form. The objective is defined as:

$$\begin{aligned}
\text{minimize } & \sum_{i=1}^{n} \xi_i \\
\text{subject to } & \forall_{i=1}^{n} : y_i \left( \langle w, x_i \rangle + b \right) \geq 1 - \xi_i, \\
& \forall_{i=1}^{n} : \xi_i \geq 0.
\end{aligned}$$

Since the primal problem is equivalent to the primal problem of training errors for non-transductive SVM we can simply use the dual form we already have transformed. For the sake of completeness, the following corollary states the dual form of the corresponding optimization problem without the proof.

**Corollary 5.1 (Dual Form of Transductive Training Error Minimization)** *The dual form of the second objective (minimize training errors) for the multi-objective transductive SVM is*

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{n} \alpha_i \\ &\textit{subject to } \forall_{i=1}^{n} : \alpha_i \geq 0 \\ &\textit{and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

*Proof.* Refer to the proof of Theorem 3.2. □

### 5.2.3. Third Objective: Minimizing the Test Error

The third problem states that the sum of test errors, i.e. the sum of the slack variables $\xi_k^*$ for the test label assignment $y^*$, should be minimized. The following theorem states the dual optimization problem.

**Theorem 5.2 (Dual Form of Transductive Test Error Minimization)** *The dual form of the third objective (minimize test error under the possible test label assignments) for the multi-objective transductive SVM is*

$$\begin{aligned} &\textit{maximize } \sum_{k=1}^{n^*} \alpha_k^* \\ &\textit{subject to } \forall_{k=1}^{n^*} : \alpha_k^* \geq 0 \\ &\textit{and } \sum_{k=1}^{n^*} \alpha_k^* y_k^* = 0. \end{aligned}$$

*Proof.* We add positive Lagrange multipliers $\alpha^*$ and $\beta^*$:

$$L_p^{(3)}(w, b, \xi^*, \alpha^*, \beta^*, y^*) = \sum_{k=1}^{n^*} \xi_k^* - \sum_{k=1}^{n^*} \alpha_k^* \left( y_k^* \left( \langle w, x_k^* \rangle + b \right) + \xi_k^* - 1 \right) - \sum_{k=1}^{n^*} \beta_k^* \xi_k^*.$$

The derivatives must be set to 0 which leads to conditions on the derivatives of $L_p^{(3)}$:

$$\frac{\partial L_p^{(3)}}{\partial w}(w, b, \xi^*, \alpha^*, \beta^*, y^*) = -\sum_{k=1}^{n^*} \alpha_k^* y_k^* x_k^* = 0,$$
$$\frac{\partial L_p^{(3)}}{\partial b}(w, b, \xi^*, \alpha^*, \beta^*, y^*) = \sum_{k=1}^{n^*} \alpha_k^* y_k^* = 0,$$
$$\frac{\partial L_p^{(3)}}{\partial \xi_k^*}(w, b, \xi^*, \alpha^*, \beta^*, y^*) = 1 - \alpha_k^* - \beta_k^* = 0.$$

Plugging the derivatives into $L_p^{(3)}$ cancels out most terms because of the first two derivatives:

$$\begin{aligned} L_p^{(3)}(w, b, \xi^*, \alpha^*, \beta^*, y^*) =& \sum_{k=1}^{n^*} \xi_k^* - \sum_{k=1}^{n^*} \alpha_k^* y_k^* \langle w, x_k^* \rangle - \sum_{k=1}^{n^*} \alpha_k^* y_k^* b - \\ & \sum_{k=1}^{n^*} \alpha_k^* \xi_k^* + \sum_{k=1}^{n^*} \alpha_k^* - \sum_{k=1}^{n^*} \beta_k^* \xi_k^* \\ =& \sum_{k=1}^{n^*} \xi_k^* - \sum_{k=1}^{n^*} \langle w, \alpha_k^* y_k^* x_k^* \rangle - \sum_{k=1}^{n^*} \alpha_k^* \xi_k^* + \sum_{k=1}^{n^*} \alpha_k^* - \sum_{k=1}^{n^*} \beta_k^* \xi_k^* \\ =& \sum_{k=1}^{n^*} \xi_k^* - \sum_{k=1}^{n^*} \alpha_k^* \xi_k^* + \sum_{k=1}^{n^*} \alpha_k^* - \sum_{k=1}^{n^*} \beta_k^* \xi_k^*. \end{aligned}$$

Together with the third derivative we can replace the $\beta_k^*$ by $1 - \alpha_k^*$ leading to

$$L_p^{(3)}(w, b, \xi^*, \alpha^*, \beta^*, y^*) = \sum_{k=1}^{n^*} \alpha_k^* \xi_k^* - \sum_{k=1}^{n^*} \alpha_k^* \xi_k^* + \sum_{k=1}^{n^*} \alpha_k^* = \sum_{k=1}^{n^*} \alpha_k^*.$$

The Wolfe dual must again be maximized which leads to the third objective of the multi-objective transductive SVM setting:

$$\begin{aligned} &\text{maximize } \sum_{k=1}^{n^*} \alpha_k^* \\ &\text{subject to } \forall_{k=1}^{n^*} : \alpha_k^* \geq 0 \\ &\text{and } \sum_{k=1}^{n^*} \alpha_k^* y_k^* = 0. \end{aligned}$$

□

## 5.3. Single-Objective but Non-Convex: The Evolutionary TSVM

Before we discuss the results for the multi-objective transductive SVM (MO-T-SVM), we will optimize the usual dual form of transductive SVM by means of evolutionary algorithms. The dual optimization problem for single-objective TSVM is given by Theorem 5.1 plus the additional upper bounds $C$ and $C^*$ for the Lagrange multipliers $\alpha_i$ and $\alpha_k^*$:

**Problem 5.2 (Single-Objective Dual TSVM Problem)** *The dual optimization problem for single-objective Transductive Support Vector Machines (TSVM) is*

$$
\begin{aligned}
\textit{maximize} \quad & \sum_{i=1}^{n} \alpha_i + \\
& \sum_{k=1}^{n^*} \alpha_k^* - \\
& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) - \\
& \frac{1}{2} \sum_{k=1}^{n^*} \sum_{l=1}^{n^*} \alpha_k^* \alpha_l^* y_k^* y_l^* k\left(x_k^*, x_l^*\right) - \\
& \sum_{i=1}^{n} \sum_{k=1}^{n^*} \alpha_i \alpha_k^* y_i y_k^* k\left(x_i, x_k^*\right) \\
\textit{subject to} \quad & \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C, \\
& \forall_{k=1}^{n^*} : 0 \leq \alpha_k^* \leq C^*, \\
& \sum_{i=1}^{n} \alpha_i y_i + \sum_{k=1}^{n^*} \alpha_k^* y_k^* = 0.
\end{aligned}
$$

The parameters $y^*$, $\alpha$, $\alpha^*$, and $b$ found by optimizing this function also yield a decision function similar to that known from the inductive case:

$$
f(x,) = \operatorname{sgn}\left( \sum_{i=1}^{n} \alpha_i y_i k(x, x_i) + \sum_{k=1}^{n^*} \alpha_k^* y_k^* k(x, x_k^*) + b \right).
$$

We used the support vector machine based on evolution strategies optimization discussed in the chapter 3 (EvoSVM) which results in the algorithm Evo-T-SVM. Individuals are real-valued vectors consisting of $\alpha$, $\alpha^*$ and $y^*$. Hence, the individual length is $n +$

| Data Set | $n$ | $n^*$ | $m$ |
|---|---|---|---|
| 4-Clusters | 2 | 2 | 2 |
| 3-Clusters | 4 | 146 | 2 |
| 2-Moons | 7 | 393 | 2 |

**Table 5.1.:** The data sets used for transductive learning. Please note that most of the examples are not labeled.

$2n^*$. The mutation is performed by adding a Gaussian distributed random variable with standard deviation $C/10$. In addition, a variance adaptation is conducted during optimization (1/5 rule [150]). Crossover probability is high (0.9). We use tournament selection with a tournament size of 0.25 multiplied by the population size. The initial individuals are random vectors with $0 \leq \alpha_i \leq C$, $0 \leq \alpha_k^* \leq C^*$, and $y_k^* \in \{-1, +1\}$ respectively. The maximum number of generations is 10000 and the optimization is terminated if no improvement occurred during the last 10 generations. The population size is 10.

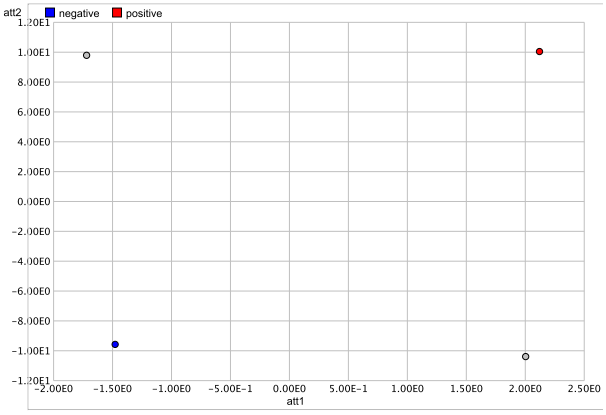
### 5.3.1. Experiments and Results

Exhaustive experiments for transductive learning are given in [28, 73, 166] including several synthetic and real-world data sets (mainly in the field of text classification). In this section, we simply show some experiments on simple low-dimensional data sets in order to give an impression of the results for readers who are less familiar with transductive learning. We will use the same data sets later in this chapter for multi-objective transductive learning since this will then allow us to interpret the Pareto fronts.

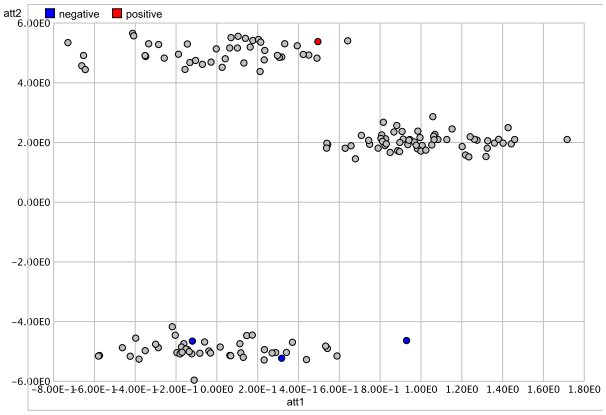
We apply the single-objective (and later also the multi-objective) transductive SVM on three different data sets. Table 5.1 summarizes some of the properties. Please note that for each data set only few labeled instances are given. Figure 5.1 shows all data sets where grey dots mark unlabeled points.

On all three data sets, we apply a traditional inductive SVM (JMySVM [160]) and the new transductive Evo-T-SVM. We start with a discussion on the results for the *4-Clusters* data set. This data set has some interesting properties:

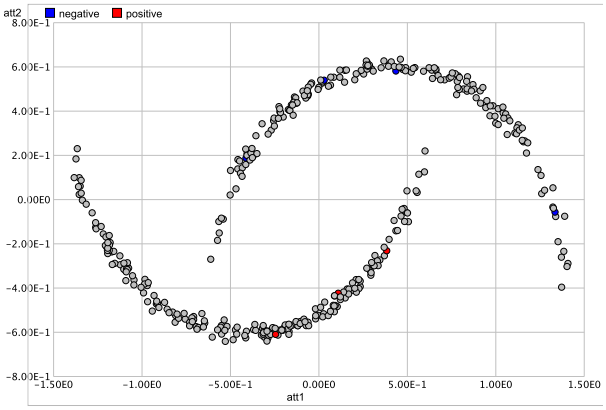
- the distance between the upper points and the lower points is larger than the distance between the left and the right group of points,
- since only the lower left and the upper right points are classified, it is likely for inductive SVMs to make an error here by ignoring these distances.




(a) 4-Clusters




(b) 3-Clusters




(c) 2-Moons

**Figure 5.1.:** The data sets used for transductive learning. Grey dots mark unlabeled data points. Please note for the 4-clusters data set that the distance along the y-axis is much bigger (about factor 8) than the distance between the clusters along the x-axis.

We expect the inductive SVM to calculate a diagonal hyperplane which tends to classify both unlabeled points to one class. A transductive SVM should be capable of taking the different distances into account and divide the upper points from the lower points by a hyperplane more or less parallel to the x-axis. As it can be seen in Figure 5.2, the results meet our expectation. The inductive SVM classifies both unlabeled data points as *negative* which is probably wrong. The transductive SVM, on the other hand, manages to take the larger distances between the upper and the lower points into account. The position of the hyperplanes is also indicated by the confidence values which are shown in Figure 5.3.

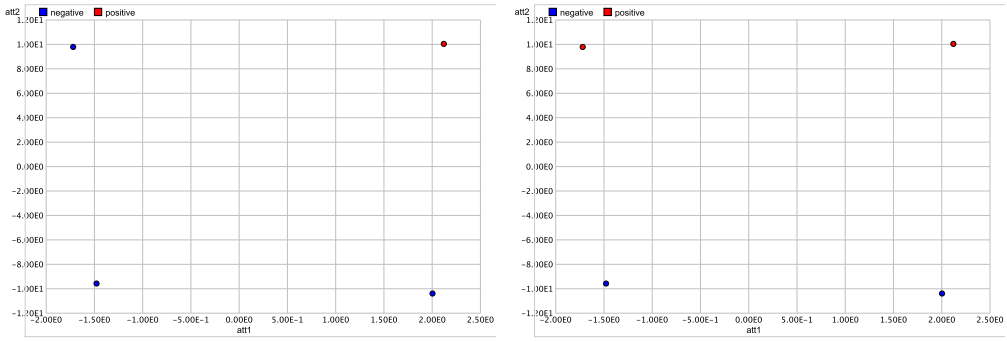
On the second data set, we can again see that the probability for generalization errors can be reduced by using transductive learning. The *3-Clusters* data set consists of three equally-sized clusters where only the upper and the lower one contain few labeled examples. The middle cluster is located a bit closer to the upper cluster and would probably also be classified as *positive* (please refer to Figure 5.1). As it can be seen in Figure 5.4, both SVM group the upper clusters together and would perform well on this given set of training data. The difference, however, can clearly be seen by looking at the confidences for both models in Figure 5.5: the inductive SVM would probably classify lower parts of the middle cluster as negative (as soon as the data points are located below the 0 on the y-axis) while the transductive SVM provides a slightly diagonal hyperplane between the lower and the middle cluster – leading to lower probabilities for errors for points sampled according to the middle cluster.

For the last data set, the *2-Moons* data set, we use an RBF kernel for both SVM. The difficulty of this data set is to take the structure into account without actually seeing it. We applied both SVMs on this data set. Figure 5.6 shows the results. It can easily be seen that the predictions of the transductive SVM are almost perfect and that the location of the unlabeled data points is taken into account. In contrast, the inductive SVM only defines a small spot for the *positive* points while classifying everything else as *negative*. This difference can also be seen in the confidence plots in Figure 5.7. In order to visualize the non-linear decision border, the background of the plot is colorized by the function value of the SVM.

## 5.4. Multi-Objective TSVM

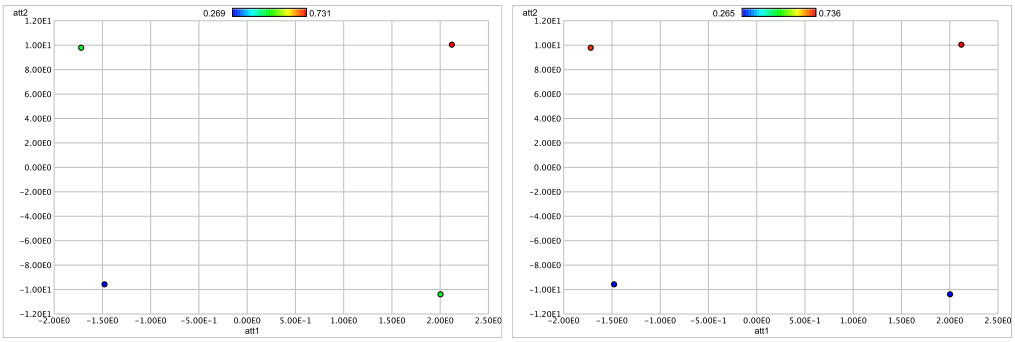
We now state the objectives for the multi-objective transductive SVM (MO-TSVM). The objectives are given as a maximization of three terms:

**Corollary 5.2 (Objectives for a Multi-Objective Transductive SVM)** *For the multi-objective transductive SVM (MO-T-SVM), the set of maximization objectives consists of*



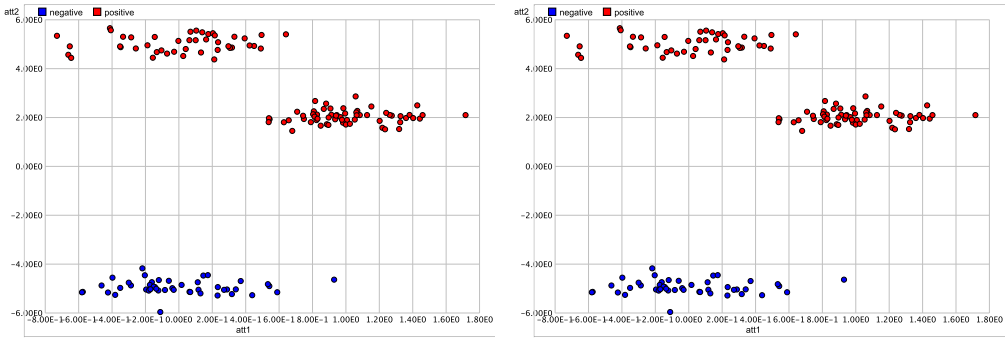

(a) JMySVM (b) Evo-T-SVM

**Figure 5.2.:** Predictions of both SVM approaches for the 4-Clusters data set. The traditional induction SVM calculated a diagonal hyperplane which decided to group the upper left and the lower right point with the lower left point. The transductive SVM made a better job by taking the distance between the unlabeled points also into account: the hyperplane is parallel to the x-axis and distinguishes between the upper and the lower points.



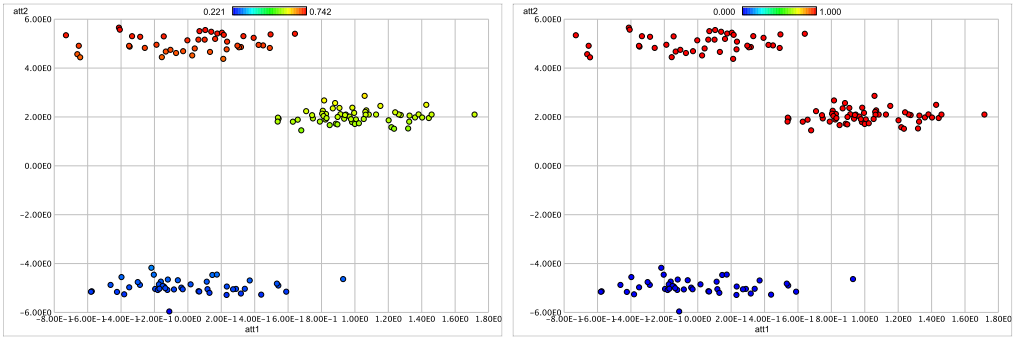

(a) JMySVM (b) Evo-T-SVM

**Figure 5.3.:** Confidences of both SVM approaches for the 4-Clusters data set. The inductive SVM calculates a diagonal hyperplane (indicated by the confidence values around 0.5 for the upper left and the lower right points). The transductive SVM, on the other hand, calculates a hyperplane clearly distinguishing between the upper and the lower points.



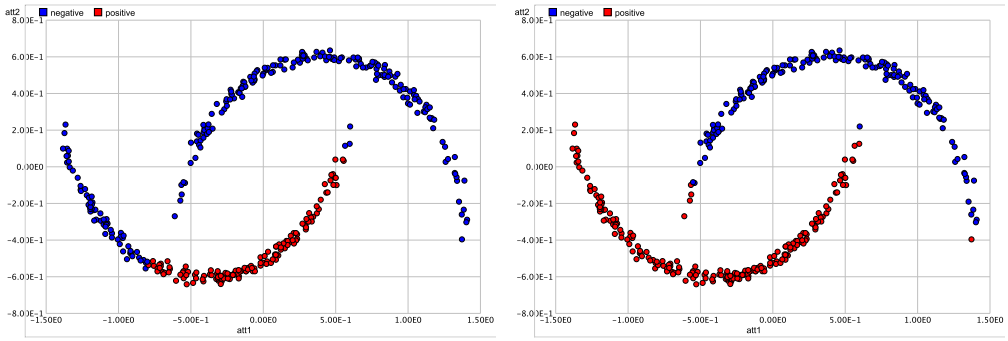

(a) JMySVM (b) Evo-T-SVM

**Figure 5.4.:** Predictions of both SVM approaches for the 3-Clusters data set. Both SVMs perform equally well in terms of the prediction on the given data set. However, the transductive SVM would perform better on completely unknown data since it takes the position of the middle cluster into account.



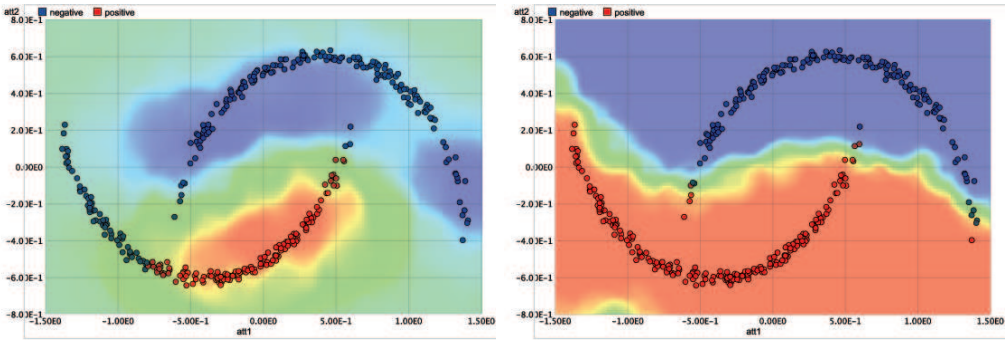

(a) JMySVM (b) Evo-T-SVM

**Figure 5.5.:** Confidences of both SVM approaches for the 3-Clusters data set. It can clearly be seen that the transductive SVM would make less errors on additional points from the middle cluster since the hyperplane is located between the middle and the lower cluster.




(a) JMySVM (b) Evo-T-SVM

**Figure 5.6.:** Predictions of both SVM approaches for the 2-Moons data set. It can clearly be seen that the non-transductive SVM was not able to capture the structure of the underlying data set. Major parts of the lower moons were wrongly predicted by the non-transductive learner.




(a) JMySVM (b) Evo-T-SVM

**Figure 5.7.:** Confidences of both SVM approaches. The semi-transparent colors indicate the values of the prediction function of both SVMs. It can clearly be seen that the transductive SVM was able to take the spacial structure of the unlabeled data points into account.

$$
\begin{aligned}
\textit{(I)}\ & \sum_{i=1}^{n} \alpha_i + \\
& \sum_{k=1}^{n^*} \alpha_k^* - \\
& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) - \\
& \frac{1}{2} \sum_{k=1}^{n^*} \sum_{l=1}^{n^*} \alpha_k^* \alpha_l^* y_k^* y_l^* k\left(x_k^*, x_l^*\right) - \\
& \sum_{i=1}^{n} \sum_{k=1}^{n^*} \alpha_i \alpha_k^* y_i y_k^* k\left(x_i, x_k^*\right) \\
\textit{(II)}\ & \sum_{i=1}^{n} \alpha_i \\
\textit{and (III)}\ & \sum_{k=1}^{n^*} \alpha_k^*
\end{aligned}
$$

*subject to* $\forall_{i=1}^{n} : \alpha_i \geq 0$ *and* $\forall_{k=1}^{n^*} : \alpha_k^* \geq 0$.

### 5.4.1. Experiments and Results

We use the same implementation of the Evo-T-SVM as before together with the non-dominated sorting selection from NSGA-II. The number of individuals was increased to 30 for 4-Clusters, to 150 for 3-Clusters, and to 200 for the 2-Moons data set. The results are shown in Figures 5.8, 5.9, and 5.10.

#### 5.4.1.1. Interpretation of the Pareto Fronts

Each plot shows the criteria (I), (II), and (III) as axes or color respectively. The points span a curved plane and it can be seen that the resulting Pareto fronts cover a wide range of the possible solution space. Each point is representing a model defined by the vectors $\alpha$ and $\alpha^*$ together with corresponding offsets. Hence, each point in the Pareto front also corresponds to a segmentation of (the given training) data points. Depending on the position of a point (model) in the 3-dimensional criteria space, the model emphasizes either a good clustering, a good classification, or a good generalization. All points in between correspond to all possible trade-offs between those goals.

The basic interpretation of the Pareto fronts is the same as for the non-transductive case (see Section 3.2.7.1). As we have seen there, some models emphasize a larger margin. Those models are located at higher regions of the z-axis (the margin size) and correspond to the red points in the Pareto set. Other models emphasize the minimization of the training error (criterion II) and are therefore located at the right parts of the x-axis. And again other models emphasize the minimization of the "training" error of the unseen data points. Those models are located at the higher parts of the y-axis. Again, all fronts show a typical structure describing the trade-off between all three criteria but in contrast to the 2-dimensional case this structure can not be seen as easily. Anyway, it is not possible to find a model (a point) above the plane described by the points of the final Pareto set since this point would dominate all others.

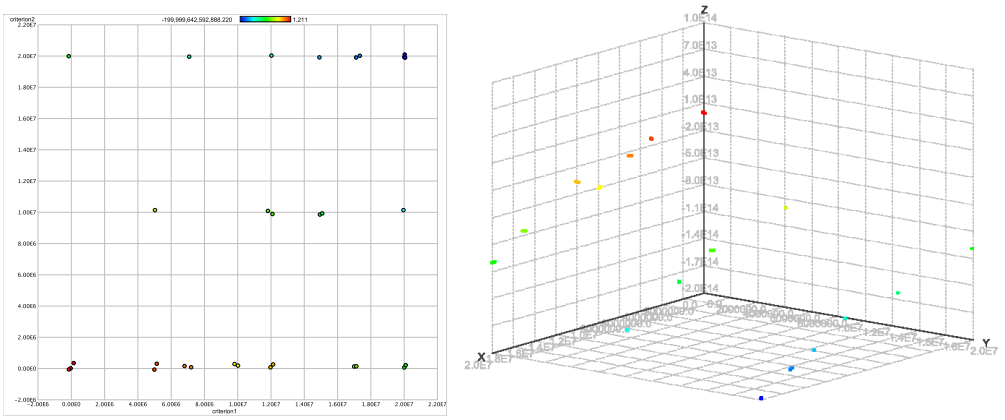
It is also quite interesting that for the simpler data sets the Pareto front shows only the finite number of discrete solutions for the possible labeling for the test data. For the more complex 2-Moons data set, on the other hand, the Pareto front is well scattered across the solution space.

We will now discuss the trade-off between the criteria depicted in the Pareto plots. Although for three criteria it can no longer be seen in the plots, we can again conclude that each Pareto front contains all sensible models in the complete trade-off region. All models between the default classifier classifying all points as the major class (a red point with smallest values with respect to the x- and y-axis) and total overfitting with zero training error (the bottom corner points) either for the labeled or the non-labeled points are possible. Users can inspect different regions of the resulting Pareto sets and choose a solution which leads to an appropriate classification and / or clustering result on the training data or an additional test data set. From a selected solution, users can define a trace in the solution space depending on the fact if solutions should be preferred which better classify or which better cluster the given data. And just as for the non-transductive SVM, users do not have to define such preferences in advance but can traverse the solution space after all feasible solutions were found.

This directly corresponds to the walk along the Pareto front we have discussed in detail in Section 3.2.7.1. We will perform a similar walk along those 3-dimensional Pareto fronts in the next section leading to further interpretations of the resulting Pareto sets.

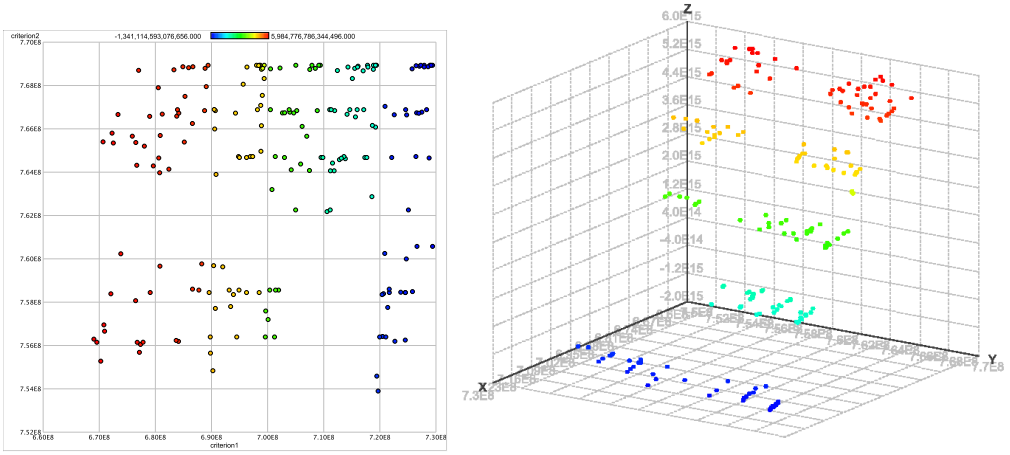
### 5.4.2. From Classification to Clustering in One Single Run

We started our discussion about transductive SVM with a motivation for using unlabeled data points in order to reduce the generalization error. A positive side effect is the reduced effort for labeling data points which often must be done manually. The general idea of making use of both the labeled and the (huge amounts of) unlabeled data is usually called *semi-supervised learning*. This class of learning methods falls between



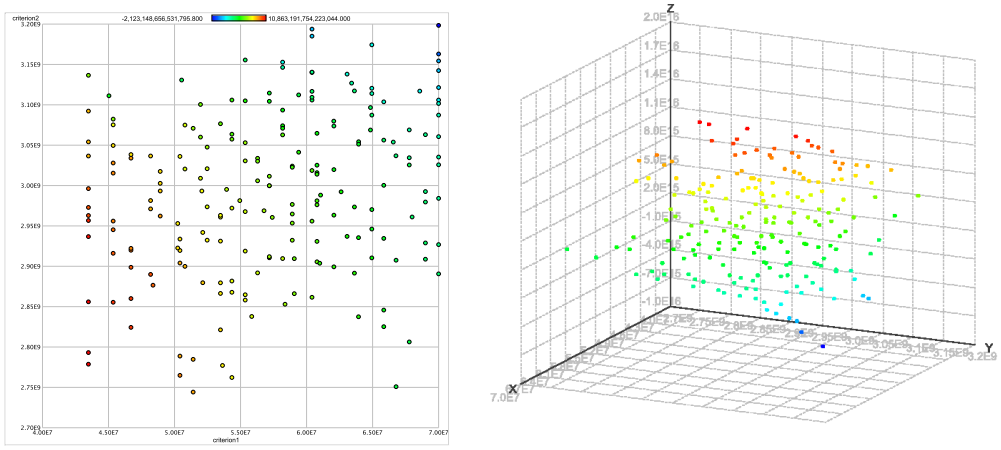
(a) 4-Clusters Pareto 2D (b) 4-Clusters Pareto 3D

**Figure 5.8.:** The resulting Pareto front in 2D and in 3D plots for the data set 4-Clusters. The color and the z-axis corresponds to criterion ($I$).



(a) 3-Clusters Pareto 2D (b) 3-Clusters Pareto 3D

**Figure 5.9.:** The resulting Pareto front in 2D and in 3D plots for the data set 3-Clusters. The color and the z-axis corresponds to criterion ($I$).



(a) 2-Moons Pareto 2D (b) 2-Moons Pareto 3D

**Figure 5.10.:** The resulting Pareto front in 2D and in 3D plots for the data set 2-Moons. The color and the z-axis corresponds to criterion ($I$).

supervised learning on completely labeled training data and unsupervised learning without any labeled training data. Transductive support vector machines are a particular instance of this class of learning methods although the original motivation of Vapnik was to define an easier class of learning tasks since it is not necessary to cover the whole input space for transductive learning.

We will now discuss the extreme spots of the resulting Pareto fronts and see how these spots are related to the concepts of supervised learning, semi-supervised learning, and unsupervised learning.

If we concentrate on only one criterion of the three criteria introduced above we could choose to

**maximize the margin:** the resulting hyperplane will be located beside all data points (labeled and unlabeled) and there will not be any meaningful interpretation of this result.

**minimize the training error:** the model is clearly overfitted to the training data.

**minimize the test error:** the model is clearly overfitted to an arbitrary test data label assignment which usually does again not allow any other interpretation.

It does not make any sense to concentrate on one of the criteria alone. We have already discussed that concentrating on the training error will lead of overfitting while concentrating on the margin maximization alone will lead to trivial models not lying in the data range at all.

Hence, we have to take pairs of criteria into account which allows for three different combinations. In the three-dimensional Pareto fronts which are the result of the multi-objective transductive SVM, this corresponds to one of the three axes between two of the criteria. All three connection lines between the extrema stated above usually form a triangular shape in the solution space. We are now making a walk along one of these three connection lines and can again choose along which line we want to move:

**maximize the margin, minimize the training error:** this will result in the traditional SVM, in case of the multi-objective SVM this will result in all possible trade-offs between model complexity and training error.

**maximize the margin, minimize the test error:** this will result in an optimal clustering with respect to the test data, in case of the multi-objective SVM this will result in all possible trade-offs between model complexity and test error (cluster performance). Please note that the results are different from those of the already known Support Vector Clustering [13].

**minimize the training error, minimize the test error:** overfitted clustering, the label assignments are correct for the training data.

The final point is that we can now combine all three of the criteria and get the whole area of the Pareto front as result:

**maximize the margin, minimize training and test error:** this will result in the transductive SVM discussed in this chapter. In case of the multi-objective transductive SVM this will result in all possible trade-offs between model complexity, training error and test error (good clusterings).

And this is the major result of the first part of this thesis. Besides the fact that multi-objective SVMs are able to deliver all these results in one single optimization run there is a quite more important conclusion from this discussion: the multi-objective transduction SVM can be seen as a formal connection between supervised and unsupervised learning. In the optimization function of this SVM there is no difference between classification and clustering. After dividing the transductive SVM optimization criterion into three single criteria and transforming each of them into its dual form of its own, this connection becomes obvious. It is as simple as this: if no training data is given, we automatically get a clustering scheme. If no test data is given, we end up with the traditional SVM for supervised learning. If both are given, we have a semi-supervised learning scheme lying in between.

# Part II.

# Feature Space Transformations

CHAPTER 6

# Multi-Objective Supervised Feature Construction

In the first part of this thesis, we have discussed the optimization procedures used directly inside of the statistical learning methods. By introducing new optimization methods we are able to solve problems like indefinite kernel learning which are not feasible for traditional optimization techniques. An additional advantage is that the very same learning algorithm can directly be applied to non-convex optimization problem of transductive learning which leads to formal connection between classification and clustering at least in the multi-objective setting.

This type of multi-objective learning is a guideline for this work since we can now also concentrate on more than one optimization criterion. These criteria can even compete with each other. The result is a Pareto set of models containing all models between those with lowest training error (overfitted models) and those with minimal model complexity (often too general models). All models are found in one single run without any user interaction or parameter settings.

We now want to transfer these positive results for statistical learning to another step in the data mining process, namely to the preprocessing phase. We will concentrate on the selection of good features in order to optimize the learning results (*feature selection*). If the given features are not sufficient to build good models, often new features must be constructed from the existing ones (*feature construction*). In the case of series data (e.g. time series), this process of feature creation is called *feature extraction* since discriminating single-valued features are extracted from the multiple-valued value series.

## 6.1. Feature Space Transformations

Although many learning schemes already try to identify relevant features and create prediction models based on those features' values, both theoretical analysis and experiments have shown that they usually do not scale well with the number of irrelevant or redundant features [50, 94]. The information gain criterion in decision trees, for example, selects features optimizing the class purity in the subsets according to the feature but it fails in cases of strong feature interactions.

The process of feature selection aims to find good subsets of features. Existing approaches include *filtering methods* like FOCUS [4] or Relief [84]. These methods calculate a weight or feature relevance measure for each feature independently - just as in the case of information gain. A (manual) selection and construction of the optimal feature set hence can be guided by such a relevance measure of the features. Several other approaches for feature relevance calculation exist, based for example on the entropy [147] or the weighting vector derived of the hyperplane of a Support Vector Machine [184]. Statistical algorithms for dimensionality reduction like Principal Component Analysis [63] also induce a weighting of the features. With exception of the SVM weighting derived from the normal vector of the calculated hyperplane, all those methods suffer from the same two drawbacks: first, they calculate the feature relevance for each feature independently from all others and can hence easily be confused by feature interactions as those known from XOR functions. Second, they calculate the feature relevance without taking a specific learning scheme into account and might therefore deliver sub-optimal results for a given learning algorithm.

In order to overcome the second problem, so-called *wrapper approaches* were proposed [87, 88]. Here, the performance of a user-defined learning algorithm is estimated by validation methods like cross validation and the search for the optimal feature subset is guided by those estimated performances. Several wrapper approaches have been proposed, including wrappers like forward selection, backward elimination [1], or genetic algorithms [196]. The first two methods are greedy search heuristics and will get stucked in the first local optimum. Genetic algorithms, on the other hand, are global search methods which will probably deliver the global optimum and can also deal well with feature interactions. This makes genetic algorithms the most appropriate solution for feature selection on large real-world feature sets and a good starting point for the feature space transformation problems discussed in the next chapters. Unfortunately, they suffer from long runtimes since the number of necessary generations might grow exponentially with the number of features. Another drawback is that they do not give any insight into the structure of the feature space, i.e. which (subsets of) features are more important than others. At least for the second problem we will discuss a solution in this chapter.

But what can we do if an inductive learning problem cannot be solved accurately by using the original feature space? This often happens in practice since standard learning algorithms cannot represent complex relationships as induced for example by trigonometric functions. For example, if only base features $X_1$ and $X_2$ are given but the target function depends highly on $X_c = \sin(X_1 \cdot X_2)$, the construction of the feature $X_c$ would ease learning – or is necessary to enable any reasonable predictions at all [17, 33, 89]. It has been shown that genetic-based feature selection can easily be extended by the construction or extraction of new features [104, 105, 115, 117, 155, 183].

In general, one is interested in a *transformation* of the original *feature space* $X$ into a feature space $X'$ in such a way that the search for a model is simplified for a given learning scheme [193, 194]. The goal is the increase of the prediction accuracy together with a higher understandability of the models. So the feature space transformation problem is defined as the search for small subsets of selected or explicitly constructed features not only helping the learning scheme but also easing the process of gaining more insight into the underlying processes.

Unfortunately, feature selection and construction are computationally very demanding tasks often requiring to search a very large space of possibilities [194]. We have discussed that heuristics like genetic algorithms have proven to be very efficient for feature selection. In the following, we will discuss how evolutionary computation, and especially multi-objective evolutionary algorithms, can help to search the very large space of feature subsets. Then, we will also extend the presented techniques to feature selection and feature construction for unsupervised learning problems in the following chapters.

### 6.1.1. Feature Construction and Genetic Programming

Before we introduce a multi-objective solution for supervised feature selection and construction, we will discuss the connection of feature construction to two closely related fields, namely genetic programming and kernel-based learning.

The (supervised) feature construction in this work resembles a wrapper approach since the performance is estimated by, for example, a cross validation of a linear regression or classification learner. The outer meta optimization selects or constructs new features which can be used by the inner linear learner and might help to improve the results.

This approach is very similar to *genetic programming* [64, 91], which also aims at finding a prediction function by applying mathematical or logical operations on a set of features. Genetic programming, however, suffers of a big problem with robustness. Small changes of values often lead to drastic changes in the value of the approximated function and genetic programming is hence very prone to overfitting. The reason for this missing robustness is a large amount of so-called introns, i.e. terms without any or only with small information for the target function. Introns are the main reason for performance

problems of genetic programming with respect to predictive power as well as to runtime.

The approach discussed in this work achieves a higher robustness against small changes of values which leads to a lower degree of overfitting by several techniques for intron prevention. We will discuss the improvements of the proposed approach in section 6.4.2.

### 6.1.2. Feature Construction and Kernels

Since the search for an appropriate kernel is equivalent to the search for an appropriate feature space transformation the success of kernel methods is also an indicator for the importance of feature construction and selection. The mapping $\Phi$ used in kernel functions actually is a specific form of feature construction. Since for many real-world data mining problems the linear dot kernel is not sufficient, the crucial aspect of learning in these cases seems to be the transformation of the input vectors into a new feature space.

One could ask why we then do not simply try to construct new kernel functions instead of constructing or extracting features from existing values. The reason is given by one of the major motivations of knowledge discovery itself: one is often not only interested in a predictive model but also in an understandable description of the underlying processes. Since most people cannot understand kernel models at all, the optimization of a kernel function would not increase the insight into the problem at hand. By explicitly selecting, constructing, and extracting features the domain expert often gets more insight into the most important terms for the learning problems. For example, the information that the model highly depends on the term $\sin(X_3 \cdot X_8)$ is often more interesting than the information stated by a collection of support vectors.

### 6.1.3. Multi-Objective Feature Space Transformations

There was some work in the field of multi-objective evolutionary feature space transformations during the last years. Table 6.1 shows the different tasks in this field and summarizes the achieved results.

Multi-objective supervised feature selection was already defined in [44]. Here, the prediction error of a classifier was estimated by a cross validation and used as a first criterion. The number of features was used as a second criterion and should be minimized.

For the supervised learning setting, the problems of feature construction and feature extraction from series data will be discussed in this and the next chapter. The idea of minimizing the number of features while the prediction accuracy should be increased can directly be transfered to feature construction and extraction.

| | supervised | unsupervised |
|---|---|---|
| selection | O [44] | X [123] |
| construction | X [155] | X [124] |
| extraction | X [105, 115, 117, 116] | ? |

**Table 6.1.:** The different tasks for multi-objective evolutionary feature space transformations. X denotes tasks solved by the author of this work, O denotes tasks solved by other authors, ? denotes tasks which are currently not yet solved.

For the first time, multi-objective optimization now also allows for feature space transformations for unsupervised learning. This problem is multi-objective by nature and we will discuss a quite surprising result in the final chapters of this work.

## 6.2. Multi-Objective Evolutionary Feature Selection

This section summarizes the results of [44] and describes the basic idea of multi-objective feature selection. Additionally, we will give an interpretation of the resulting Pareto fronts.

Feature selection is a well-suited problem setting for multi-objective optimization. In the simplest case, the task of feature selection involves two objectives: minimization of the number of features and maximization of the modeling performance. In classification tasks, performance can be assessed in terms of the misclassification rate or in terms of the prediction accuracy. For regression tasks, one could use for example the root mean squared error or other regression performance measures.

A common approach is to combine the objectives in a composite function [196]. This may yield solutions good enough on average but not in each one of the objectives separately. Alternatively, multiple runs can be performed to optimize one objective, while keeping the rest at a desired level. For example, it is possible to optimize performance for a fixed and given subset size. This can be pursued with EAs in a number of ways but, in principle, this would include the evolution of a population of solutions increasingly concentrated around the desired subset size.

In [44] it is argued that beside larger runtimes through multiple optimization runs there is a danger of eliminating useful population diversity by imposing restrictions on the subset size. Thus, the chromosomes of diverse subset size may not stand a chance to pass on well performing schemata to the next generation. Such population diversity can be maintained by aiming both at subset size minimization and performance maximization, without specifying which objective is more important. The authors employ a variation of the Niched Pareto GA (NPGA) [66]. Although this multi-objective optimization

scheme is today known to be inferior compared to modern schemes like NSGA-II, the achieved results are already quite impressive. The reason for this might be that the authors optimized the scheme parameters and algorithms in order to better fit the needs for multi-objective feature selection. One of these extensions was the introduction of a new commonality-based crossover operator. The authors evaluated their algorithm on a probabilistic neural network model which offers faster evaluation times if only subsets of the features were used.

### 6.2.1. Interpretation of the Pareto Fronts

Figure 6.1 shows the result of a multi-objective feature selection process. The used data set consists of 15 attributes which are all equally important to the learner (the label simply is the sum of all attribute values). We applied the described multi-objective feature selection approach on this data set which delivers the shown Pareto front. The x-axis shows the negative root mean squared error which should be maximized. The y-axis shows the negative number of used features, which also should be maximized. The negation transforms the original minimization problems into maximization problems. Hence, the goal is to move the Pareto front as far to the upper right part of the plot as possible.

Each point in the plot corresponds to feature subset of the original input data set. The first point, for example, corresponds to a feature subset of the data set consisting of one feature only, namely the features named *att*2. It can clearly be seen that the number of features in the Pareto front almost covers the complete range from 1 to 15. The analyst can tell from the straight line that all features are more or less equally important and even features added as 10th or 14th feature lead to a significant decrease of the error.

It can also be seen that starting with the smallest subsets, the larger subsets are often build on top of the previous ones. For example, the optimal subset with size 4 contains all three attributes *att2*, *att13*, and *att4* from the optimal subset of size 3 and one additional feature *att9*. This inherent structure in the Pareto front gives insight into the relevance ranking of the features.

However, if a perfect ranking could always be created, greedy heuristics like hill climbing would be more successful for feature selection – which they are not! The Pareto front shows, however, that in some cases the attribute subsets completely change (for example between subset size 4 and 5) and therefore greedy heuristics will fail. Such changes in subset relevance are also interesting on its own for analysts.

Although in the specific case shown in Figure 6.1 those changes are probably induced by small data changes, it should again be noted that these changes in feature set ranks in general cannot be fully explained by data changes alone. Those changes in feature set ranks are a structurally inherent part of the analysis problem at hand. Otherwise, as we

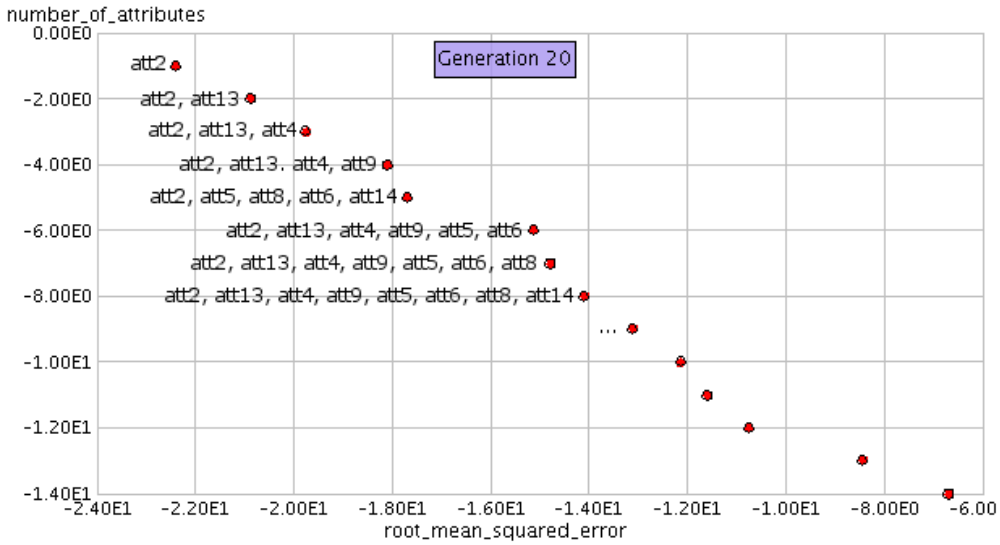



**Figure 6.1.:** The figure shows the resulting Pareto front for a simple learning problem containing only 15 equally important features. The analyst can derive a feature ranking from such a Pareto front and gets also a hint where such a ranking does not apply (e.g. between subset size 4 and 5).

have mentioned before, greedy feature selection algorithms like forward selection would not fail for most of the data sets. But this is what actually can often be observed: the greedy feature selection methods fail due to the multi-modal nature of the optimization problem and the missing structure in the rank of the feature subsets.

It can be concluded that multi-objective feature selection does not suffer from these multi-modal problems and deliver a complete ranking of all sensible feature subsets starting from the smallest to those containing all necessary features. If the data set would have also contained some noise features which should have been deselected, those features would not have been selected into many of the feature subsets and the Pareto front would have shown the typical vertical part on the right showing that adding more features would not lead to a significant increase in prediction performance and could hence be omitted.

## 6.3. Regularized Feature Selection and Construction

We will now connect the problem of feature selection and construction to the idea of statistical learning discussed so far and extend the basic multi-objective feature selection idea presented above to the case of feature construction.

### 6.3.1. Regularized Risk for Feature Space Transformations

The basic definition of machine learning used so far was the minimization of the regularized (or structural) risk

$$R_{reg}(\gamma) = R_{emp}(\gamma) + \lambda\Omega(\gamma).$$

The term $R_{emp}(\gamma)$ measured the training error and the term $\Omega(\gamma)$ measured the structural complexity of the model defined by $\gamma$. The weighting factor $\lambda$ defines the trade-off between both terms.

In the first part of this work, we have seen that this basic definition of statistical learning is actually multi-objective and that analysts can get more insight and better models by explicitly stating the corresponding optimization problems and solving them with appropriate optimization schemes. The selection or construction of new features actually can also be considered to be a part of the learning process. The search for advantageous input representations often is the key for successful learning and there seem to be a limit similar to that known from the no free lunch theorems in optimization and search [193, 194]: either the hard work is done during preprocessing or during learning [130].

Let us consider one of the most simplest learning schemes available, namely linear regression or linear separating hyperplanes respectively. The wrapper approach [87, 88] ensures that the performance estimation for this learner for a given feature space corresponds to the expected risk (and not only the empirical or regularized risk). The model description $\gamma$ therefore only describes the feature space and $\Omega(\gamma)$ should measure its complexity. We replace $\gamma$ by the definition of our input space $X$ and the complexity measure by $\Omega(X)$ and can define a new regularized risk, the regularized risk for feature space transformations:

**Definition 6.1 (Regularized Risk for Feature Space Transformations)** *Let $R_{inner}$ be the (expected) risk of the inner learning scheme and let $\Omega$ be a strictly monotonic increasing function. The* REGULARIZED RISK FOR FEATURE SPACE TRANSFORMATIONS *is defined as*

$$R_{reg}^{FST}(X) = R_{inner} + \lambda\Omega(X).$$

### 6.3.2. Definition of $\Omega(X)$ for Feature Selection and Feature Construction

The used wrapper approach already delivers the risk $R_{inner}$ at least for supervised learning problems. For unsupervised learning tasks, we will discuss several options for accessing a quality measure $R_{inner}$ in Chapter 8 and 9. We mainly concentrate on the definition and calculation of $\Omega(X)$ for supervised and unsupervised learning. This term measures the *feature space complexity*. In the case of multi-objective supervised feature selection, we have already seen above that the number of used features is a simple

and good measurement for the feature space complexity. We will now discuss a formal connection between this intuitive solution and learning theory.

In computational learning theory, the VC-dimension (for Vapnik-Chervonenkis dimension [186]) is a measure of the capacity of a statistical classification algorithm, defined as the cardinality of the largest set of points that the algorithm can shatter.

**Definition 6.2 (VC-Dimension)** *The* VC-DIMENSION *of a model $f$ is the maximum $n$ such that some data point set of cardinality $n$ can be shattered by $f$.*

**Definition 6.3 (Shattering)** *A classification model $f$ with some parameter vector $\gamma$ is said to* SHATTER *a set of $n$ data points if, for all assignments of labels to those points, there exists a $\gamma$ such that the model $f$ makes no errors when evaluating that set of data points.*

Informally, the VC-dimension or *capacity* of a classification model is related to how complicated it can be. For example, consider thresholding by means of a high-degree polynomial: if the polynomial evaluates above zero, that point is classified as positive, otherwise as negative. A high-degree polynomial can be wiggly, so it can fit a given set of training points well. But one can expect that the classifier will make errors on other points, because it is too wiggly. Such a polynomial has a high capacity. A much simpler alternative is to define a threshold by linear function. This polynomial may not fit the training set well, because it has a low capacity.

The same idea applies to feature selection. The inner learner is determined at the beginning of the feature subset optimization process and cannot be changed. At least for feature construction, which is discussed here, one would usually choose a simple model class like linear separating hyperplanes or linear regression since the hard non-linear learning work should be performed by feature construction. In this case, it can easily be shown that the complexity of the model increases with increasing number of used features.

**Lemma 6.1 (Feature Space Complexity)** *For multi-objective supervised feature selection (construction), the number of currently used features $nf$ is a feature space complexity measure, hence $\Omega(X) = nf$.*

*Proof.* Let the model class be fixed and let it be a linear hyperplane classification model. If only one feature is used, the model is only able to select one single threshold for the classification in this single dimension. The highest number $n$ of data points which can shattered is 2. Let $nf$ be the currently used number of features. In general, linear separating hyperplanes in a space $\mathbb{R}^{nf}$ have a VC-dimension of $nf + 1$ [186]. Increasing $nf$ therefore also increases the complexity $\Omega(X)$. □

Everything said above can directly be transferred to the case of feature construction. Constructing new features actually is only a special case of feature selection. One could add all features to the original input space $X$ which could be constructed by a given set of methods. This new space $X'$ could then be used as a basis for feature selection. In this case, Lemma 6.1 would still hold and the number of features which should be selected should also be minimized in order to minimize the regularized risk $R_{reg}^{FST}(X)$.

## 6.4. Multi-Objective Evolutionary Feature Construction

We will now present an extension of the multi-objective feature selection technique presented above. This new algorithm should be able to handle numerical regression and classification tasks where the label $Y$ depends in some non-linear way from the input features $X$. The algorithm should find a transformation into a new feature space $X'$. The problem of feature construction is that the search scheme must be able to handle feature interactions [54]. A feature interaction is defined as the influence of one feature on the class probability depending on another feature, i.e. the features $X_i$ and $X_j$ are interacting if $P(Y|X_i) \neq P(Y|X_i, X_j)$. We will see that evolutionary algorithms are capable of handling these interactions which often lead to multi-modal objective functions.

### 6.4.1. Problem-Specific Search Operations

The evolutionary optimization of the input representation regards any possible representation as a search point in the search space. The individuals of a population are hence given by feature sets $X$ of variable lengths. For this reason, evolutionary feature space transformations are actually an instance of the class of genetic programming. In contrast to the usual genetic programming approaches, however, the goal is not the full mathematical description of the underlying processes. The inner learning scheme still searches for the connection between the label and the input variables. The outer optimization aims at easing this inner search process by identifying single terms of this connection. This separation into inner learning and outer feature space optimization ensures a greater robustness against random noise compared to mere genetic programming.

#### 6.4.1.1. Mutations

We will now define some problem-specific search point operations which can be applied on feature sets of variable lengths. We start with two different types of mutations:

**Feature Selection:** this mutation changes the selection state of a feature with probability $1/nf$ if $nf$ is the current number of features of this individual. This makes the mutation probability dependent of the individual which should be mutated.

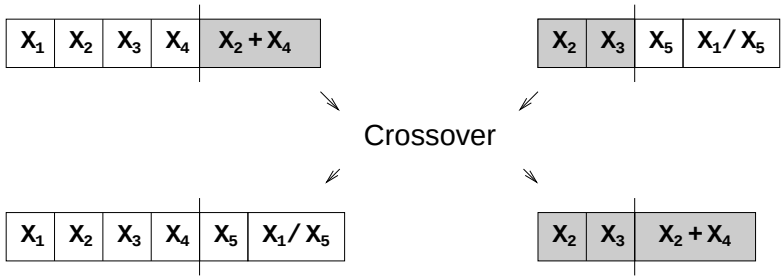



**Figure 6.2.:** Modified one-point crossover for individuals with variable lengths. All crossover variants ensure than single features are not added more than once to an individual.

**Construction:** the construction of a new feature regarding the value types of the construction function arguments. Special generators only work on integer features, other only work on real-valued features. A type-restrictive selection of generators can decrease the needed runtime [154, 155].

Selection and construction "compete" which each other. The selection aims at pruning the feature space and ease the model learning process in simpler feature spaces. The latter extends the hypothesis space for the inner learner and often this might be necessary for successful learning. By using both mutations at the same time we try to combine both optimization directions since not all constructed features help to improve the prediction accuracy [155].

#### 6.4.1.2. Crossover

Since the length of the individuals can change by adding generated features we developed crossover schemes independent of the individuals' size. Figure 6.2 shows a crossover operation based on one-point crossover. This crossover is an instance of the class of homologous search point operations [145] which are mostly used for genetic programming. We also developed a variant of uniform crossover. All crossover variants ensure that a single feature is not added more than once to an individual.

#### 6.4.1.3. Selection

The fitness of the individuals is estimated by

1. applying the feature space transformation on the data set at hand,
2. evaluating the selected learning scheme by a $k$-fold cross validation.

Since the learning step is the most time-consuming step in the feature space optimization, it is recommended to use a fast and simple learning scheme like linear regression or a perceptron.

The data mining software RapidMiner [48, 113, 125] can be used to ease the optimization process. RapidMiner offers a data management based on a central data table and different views on this table. These views can be used as individuals and the data does not need to be copied during the optimization. Only the construction of new features adds new columns once during the optimization which will be re-used if the corresponding feature should be re-constructed. This prevents multiple data storages and ensures an efficient data access.

### 6.4.2. Code Bloat and Intron Prevention

All constructive search methods like genetic programming or the approach proposed here suffer from a problem which is known as *code bloat*. This means that the size of the input representation grows faster than the corresponding increase in prediction accuracy [93]. This phenomenon can be explained by the existence of so-called *introns* [8, 157, 170]. In the case of machine learning, these introns can be defined as features without any additional information for the inner learning scheme. Examples for such features are

- features which are already part of the feature set,
- features like $X_i + X_i$ which can be seen as a weighted variant, or
- features like $X_i/X_i$ or $X_i - X_i$ which are constant.

At least for statistical learning schemes, a change of the feature weight is not relevant since learning schemes like linear regression or support vector machines calculate a feature weighting on their own. Constants can only be used by neural nets which therefore contain a default constant on their own. Another form of code bloat arises from the non-observance of the commutativity of the used mathematical operations. Adding a feature $X_i \circ X_j$ to a feature set which already contains $X_j \circ X_i$ does not provide any new information for the inner learning scheme if the composition $\circ$ is commutative. It also contradicts to the creation of simplified hypothesis.

In contrast to the approach proposed in [155], we try to reduce the number of feature space transformations which are simply useless in combination with most learning methods. In order to prevent unnecessary code bloat and the corresponding increase in runtime, the generation of such useless features should not be allowed. Therefore, we improved the generating mutation such that it will only generate features in a specific way which prevents commutative generations or the generations of useless constants. This improved mutation is the basis for an evolutionary feature construction operator which is called Yagga2 in the RapidMiner system.

#### 6.4.2.1. Intron Prevention by Sampling Equivalence Checks

Skipping commutative generations or the generation of constants significantly reduce the amount of generated introns. However, there is another desired property for the construction of new features. If the feature set already contains an equivalent feature, the one with the higher construction complexity should be removed or not created in a first place. Unfortunately, the theorem of Richardson [152] states that the detection of equivalence is NP-hard for general numerical domains and functions. Although efficient solutions exist for some special cases, we need a equivalence check which works also in the general case. Another desired property of such a check would be the identification of very similar features in their feature value ranges. It is for example, well known that $x \approx \sin(x)$ holds if $x$ is close to 0. The equivalence check should also identify these similarities and should also remove the more complicated features in these cases.

We propose a rather simple and also very efficient solution for this equivalence check:

1. create an artificial data set for the used base features regarding the actual value ranges, the number of created examples $a$ should be small,
2. calculate the constructed features (if necessary),
3. perform a correlation calculation or a t-test in order to determine the similarity of both features,
4. remove the more complicated feature if the correlation exceeds a threshold $\sigma$.

Another advantage of this algorithm is the fact that the number of necessary artificial examples $a$ can be determined depending on the desired confidence for equivalence. Lower thresholds $\sigma$ lead to simpler constructed features since more features are regarded equivalent. This sampling-based equivalence check is one of the essential ingredients for the intron prevention of the RapidMiner operator Yagga2. The final optimization of this evolutionary feature construction in discussed in the next section.

### 6.4.3. A New Generating Mutation for Intron Prevention

The previously discussed improvements for code bloat prevention decreased the number of introns but still a lot of unnecessary large features are constructed which are an indicator for overfitting. The main reason for this was that the probability for deselected features being re-selected and used for further constructions is relatively high. This leads to complex semi-successful features which are deselected and selected over and over again. The suggestions above already reduced this problem compared to the original work in [155] but still the number of too complex features is too high.

A solution for this problem could be to completely remove a feature from a feature set instead of simply deselecting it. This could, however, lead to situations where relevant original features are removed and the optimization scheme cannot move across local extrema. Therefore, we developed an alternative mutation strategy called *combined probabilistic mutation*. In the expectation, this mutation leads to the desired behavior. Let $p$ be the basic mutation probability which might also depend on the current number $nf$ of features. The combined probabilistic mutation performs the following steps:

**Removal:** the mutation deleted each of the $nf$ features of the given individual with probability $\frac{2}{4p \cdot nf}$,

**Construction:** the mutation constructs a new feature with probability $\frac{1}{4p}$,

**Adding Original:** the mutation adds with probability $\frac{1}{4po}$ one of the $o$ original features to the individual.

Since constructing and adding features increase the feature set size, the probability of feature removal is twice as high as the probability for adding features. This ensures that in the expectation the size of the feature set remains the same until longer or shorter feature sets actually prove to be better. This mutation strategy is superior especially in cases where only less complex functions of many of the original features have to be built. All three techniques for intron prevention, namely structural equivalence checks, the sampling-based equivalence checks, and the combined probabilistic mutation, are incorporated into the YAGGA2 operator which will be shortly evaluated in the next section and used as a base for the comparison in Chapter 10.

## 6.5. Experiments and Results

Exhaustive experiments for evolutionary feature construction are presented in [138, 154, 155] and also in Chapter 10. In this section, we will concentrate on a simple numerical function dependent of only one variable in order to give a visual impression of the ability of the improved feature construction. The function which has to be learned is defined as

$$f(X) = 3 \cdot X_1^3 - X_1^2 + \frac{1000}{|X_1|} + 2000 \cdot |X_1| + \varepsilon$$

with Gaussian noise $\varepsilon$ sampled from $N(0, 0.02)$ and an additional random noise feature $X_2$ which is also part of the input space $X$ together with the variable $X_1$. The data set of 500 data points is uniformly sampled in the range $[-20, 25]$ for feature $X_1$ and with mean 0 and standard deviation 1 for the noise feature $X_2$. Figure 6.3 shows the regression label plotted against the feature $X_1$. Please note the general structure of the function including the small peak around 0. The plot shows the 2% label noise but not the influence of the noise attribute $X_2$.

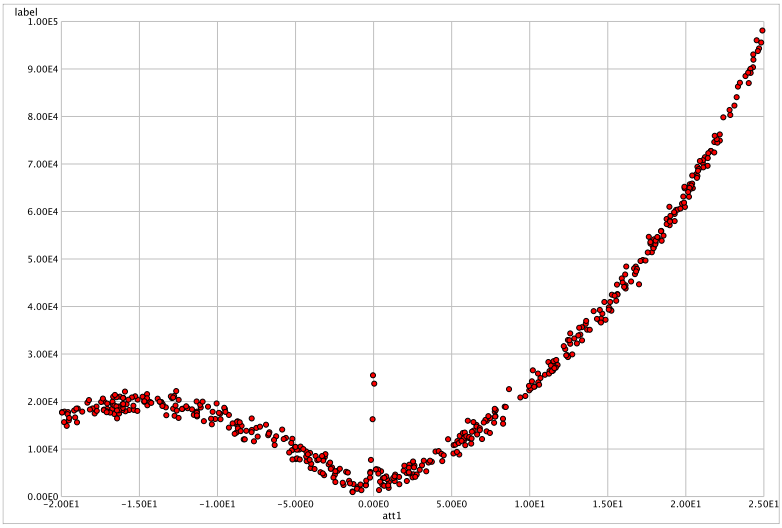



**Figure 6.3.:** A simple non-linear function which can hardly be completely learned by state of the art learning schemes without feature construction.

We first apply a regression support vector machine with radial basis kernel function but without any feature construction on this data set. Figure 6.4 shows the result. It can clearly be seen that the noise attribute affects the prediction of the learned model. Furthermore, only the most basic parts of the structures are learned (the model for example misses the peak around 0) and the error gets larger at the edges of the considered space. The latter is a well known phenomenon for support vector machine models since the number of support vectors get smaller at the edges of the data space compared to the central parts. Without support vectors, the predictions tend to become 0.

If we apply only a feature selection, the random noise attribute $X_2$ will be removed. This drastically improves the quality of the learned model, although the predictions are still wrong at the edges of the data space. The peak around 0 is also not covered by the learned model. Figure 6.5 shows the result.

As a last improvement before we apply the multi-objective feature construction discussed in this chapter, we increase the error penalty $C$ for the support vector machine. This should lower the error at the data space edges. In fact, this error is decreased as it can be seen in Figure 6.6. If $C$ is set to high values ($10^8$ in this case), the values at the edges will be correctly predicted but there are several disadvantages: first, the learning times become infeasibly long. The learning machine needed about 10 minutes for such a simple and small data set like the one discussed here. Second, overfitting to the label noise begins and a good parameter tuning for $C$ or a multi-objective learning run has to be performed. This would probably take a very long time taking the computation time

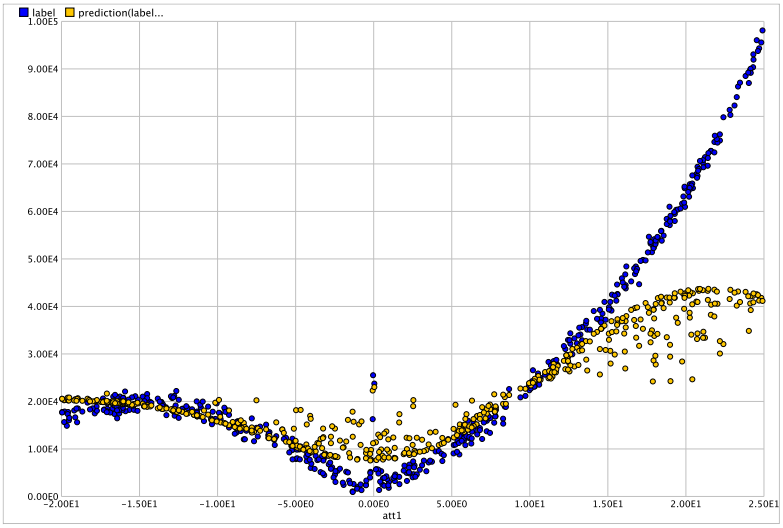



**Figure 6.4.:** A regression SVM (RBF kernel) model built on the discussed data set. The influence of the noise attribute can clearly be seen, also the missing support vectors at the data space edges affect the model quality.

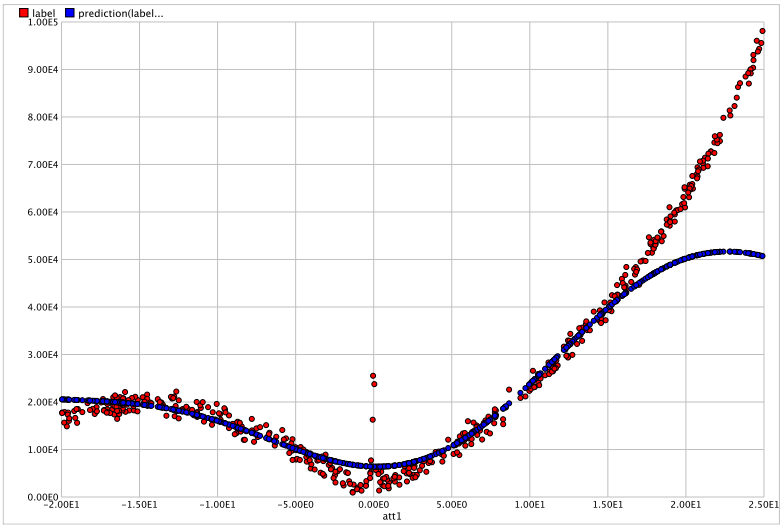



**Figure 6.5.:** A regression SVM (RBF kernel) model built on the discussed data set after the noise attribute was removed. The missing support vectors at the data space edges still affect the model quality.

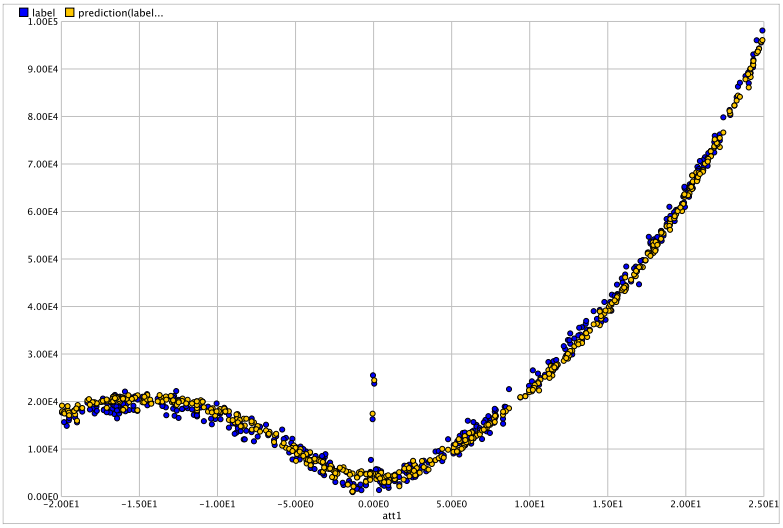



**Figure 6.6.:** A regression SVM (RBF kernel) model built on the discussed data set after the noise attribute was removed and the value for the parameter $C$ was drastically increased. Although the error at the edges vanished, the learning takes too long time now and overfitting to the label noise begins.

above into account at least for the parameter optimization approach and is hence not feasible for many learning tasks.

Even applying a powerful non-linear learning scheme like regression SVM with RBF kernels is not sufficient to deliver a good model for the underlying processes of this simple data set. Even worse is that analysts usually cannot understand the created model and no insight is gained by looking at it. Therefore, we now apply the multi-objective feature construction approach discussed in this section on the data set. We use a simple and fast linear regression learner (no kernel mapping) as the inner learning scheme and evaluated it with a fixed single split validation. The possible feature generators were multiplication, the calculation of reciprocal values, triangular functions, and the exponential function which turned out to be a good set of basic feature construction generators on a wide range of applications. The result is shown in Figure 6.7. It can clearly be seen that the constructed features help to get the global structure of the function. It even allows to predict the peak around 0. No influence of noise can be seen.

### 6.5.1. Interpretation of the Pareto Front

The corresponding Pareto front of this result is shown in Figure 6.8. The x-axis shows the negative squared error which should be maximized. The y-axis shows the negative

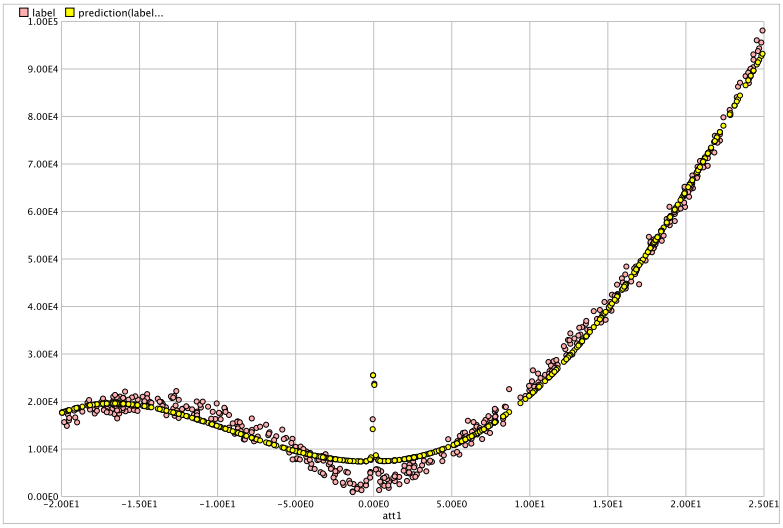



**Figure 6.7.:** A linear regression model based on the newly constructed features. The overall structure is found and overfitting does not occur. In contrast to the optimized SVM learning run of Figure 6.6, the learning procedure needs less than one second and the new attributes give additional insights.

number of used features, which also should be maximized. The negation again transforms the original minimization problems into maximization problems. This ensures that all multi-objective optimization in this thesis try to move the Pareto front as far to the upper right part of the plot as possible.

Each point in the plot corresponds to set of features which might consist of original features and / or those features constructed from existing ones. The first point, for example, corresponds to a very simple feature set consisting of one feature only, namely the original feature $X_1$. Moving to the lower right part of the curve, larger feature sets are depicted by the points. The curve in the Pareto front tells the analyst that not all features are equally important.

Instead of that, a feature ranking can clearly be seen starting with the single feature $X_1$ to a feature set containing 5 features, each of which describe one part of the generating process. Table 6.2 summarizes the found features. The analyst can read the most important terms from this table which of course is even more important for problems with higher dimensions than in this illustrating example. Please note that this ranking is very interesting, adding any feature after the third hardly improves the prediction performance which can directly be seen in the Pareto front: the negative squared error is hardly increased after the first three features.

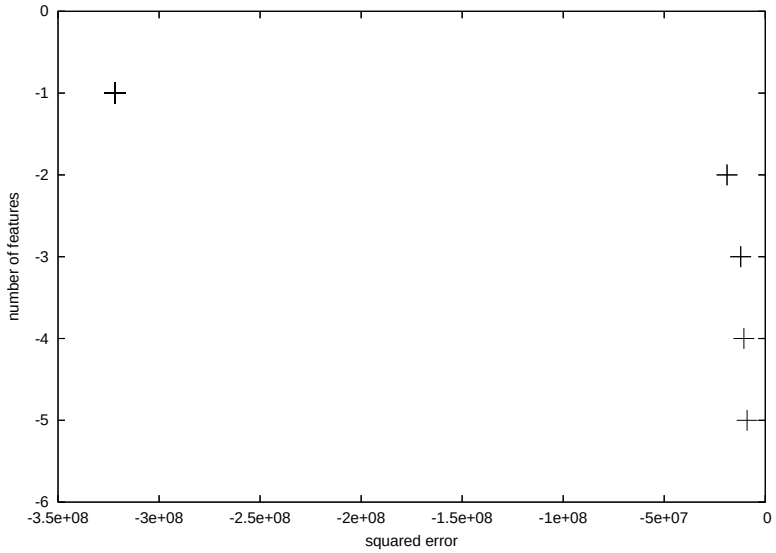



**Figure 6.8.:** The feature ranking induced by the multi-objective feature construction approach discussed in this chapter. It can clearly be seen that the first three features are most important for this data set.

It can also be seen, that the multi-objective selection scheme again was able to cover the complete possible range of sensible solutions. From a single feature (-1 on the y-axis) to a maximum of 5 features all possible feature numbers are part of the resulting Pareto set. Adding more features than 5 does not seem to help to reduce the prediction error since an almost vertical part is already reached for more than 3 features.

Finally, the Pareto front again starts with the smallest feature sets and larger feature sets are often build on top of the previous ones. For example, the optimal feature set with size 2 contains the attributes $X_1$ and $X_1^2$. The optimal feature set with size 3 consists of all features from the optimal feature set of size 2 and one additional feature $X_1^3$. This inherent structure in the Pareto front gives insight into the relevance ranking of the features even for the multi-objective feature construction case. However, just as for the mere feature selection case where such a ranking can also not be guaranteed for all cases, this also applies for feature construction. Experiments have shown [155] that greedy heuristics will more likely fail in the constructive setting due to the multi-modal optimization problems induced by feature interaction.

We can conclude that the Pareto front contains all sensible feature subsets in the trade-off region between the most simple model based on one feature only and more complicated models based on a set of partly constructed features.

| Rank | Number of Features | Feature Set |
|---|---|---|
| 1 | 1 | $X_1$ |
| 2 | 2 | $X_1$; $X_1^2$ |
| 3 | 3 | $X_1$; $X_1^2$; $X_1^3$ |
| 4 | 4 | $X_1$; $X_1^2$; $X_1^3$; $1/X_1^2$ |
| 5 | 5 | $X_1$; $X_1^2$; $X_1^3$; $1/X_1^2$; $X_1^4$ |

**Table 6.2.:** The feature sets of the Pareto front presented in Figure 6.8. The inherent structure can again be seen, adding more features than three hardly improves the prediction performance.

CHAPTER 7

# Multi-Objective Supervised Feature Extraction

This chapter describes how a set of describing features can be extracted from instances given in the form of series data. Since the data and the learning problems analyzed in this chapter derive from the field of musical classifications, we use this particular learning problem in order to motivate an adaptive feature extraction from series data. Later, we will also discuss a forecasting or prediction setting on stock data.

Firstly, this chapter presents a unifying framework for feature extraction from value series. Operators of this framework can be combined to feature extraction methods automatically, using a genetic programming approach. The construction of features is guided by the performance of the learning classifier which uses the features. Our approach to automatic feature extraction requires a balance between the completeness of the methods on one side and the tractability of searching for appropriate methods on the other side. In this chapter, some theoretical considerations illustrate the trade-off. After the feature extraction, a second process learns a classifier from the transformed data. The practical use of the methods is shown by two types of experiments: classification of musical genres and classification according to user preferences.

Secondly, we will show that $\Omega(X)$ discussed in the last chapter again can simply be defined as the number of used features and we will see that the quite more complex process of supervised feature extraction can also easily be turned into a multi-objective optimization problem.

## 7.1. Feature Extraction from Audio Data

Today, many private households as well as broadcasting or film companies own large collections of digital music plays. These are time series that differ from, e.g., weather reports or stocks market data. The task is normally that of classification, not prediction of the next value or recognizing a shape or motif. New methods for extracting features that allow to classify audio data have been developed. However, the development of appropriate feature extraction methods is a tedious effort, particularly because every new classification task requires tailoring the feature set anew.

Since music is stored in digital form and distributed via the internet, there is a need for the management and retrieval of audio data. How can we index large numbers of audio records? How can we structure music databases according to genre (e.g., classic, pop, hip hop) or occasions (e.g., dinner, party, wedding)? How can a system automatically recommend music records to users? Information retrieval has started several efforts to automatic indexing [92] and retrieval (e.g., querying by humming [58]). Machine learning has shown its benefits for text classification and ranked document retrieval with respect to user preferences [74]. It is straightforward to expect a similar benefit for the classification and personalized retrieval of music records.

Confronted with music data, machine learning encounters a new challenge of scalability:

- music databases store millions of records,
- given a sampling rate of 44100 Hz, a three minute music record has a length of about $8 \cdot 10^6$ values.

Moreover, current approaches to time series indexing and similarity measures rely on a more or less fixed time scale [79, 80]. Music plays, however, differ considerably in length. More general, time series similarity is determined with respect to some (flexible and generalized) shape of curves [77, 197]. However, the shape of the audio curve does not express the crucial aspect for classifying genres or preferences. The $i$-th value of a favorite song has no correspondence to the $i$-th value of another favorite, even if relaxed to the $(i \pm n)$-th value. The decisive features for classification have to be extracted from the original data. Some approaches extract features from music given in the form of Midi data, i.e. a transcription according to the 12 tone system [98][1]. This allows to include background knowledge from music theory. Usually, music data is given in the form of – possibly compressed – waves records, the audio data. Hence, feature extraction from audio data has become a hot topic recently [60, 97, 181, 201]. Several specialized extraction methods have shown their performance on some task and data set. It is now

[1] For an overview, see [143].

hard to find the appropriate feature set for a new task and data set. In particular, the problems are:

**Unifying framework missing:** The large set of extraction methods has not yet been systematically investigated, a unifying framework is still missing. This makes it hard to compare the proposed feature sets and to detect missing feature extraction methods.

**Variety of feature sets:** Different classification tasks ask for different feature sets (see Section 7.4). It is not very likely that a feature set delivering excellent performance on the separation of classical and popular music works well also for the separation of techno and hip hop music. Classifying music according to user preferences even aggravates the problem.

**Large search space for feature sets:** There is no concise feature set from which we would select an appropriate subset for a new task and data set by standard wrapper approaches [87]. Even if it existed, it would be most cumbersome to enumerate it.

In this chapter, we present our approach to tackle the problems. We present a unified framework for extraction methods in Section 7.2. The framework covers the known methods, and several new ones have been added. The repository of elementary extraction operators allows us to handle feature extraction as a sequence of data transformations which delivers a feature set in the end. Hence, we construct a feature set for each given task and data set, anew. Since it would be tedious to do so by hand, we apply a learning algorithm to construct the feature set for us. Section 7.3 describes the genetic programming approach to the automatic construction of (nested) sequences of data transformations, the method trees. The search within the universe of method trees is guided by a fitness function. Here, we embed a classification learner: the better the learning result using the transformed data, the higher the fitness of the feature set (i.e., the method tree). Genetic programming puts together the building blocks of feature extraction operators according to the targeted classification task and data set. It outputs a feature extraction method tree. Applying a method tree to the given audio data delivers a transformed data set, i.e., the examples rewritten by the corresponding feature set. This becomes the input to a second learning step, namely classifier learning. Figure 7.1 shows the overall process with the two learning steps, one using genetic programming, the other using the support vector machine mySVM [160] for classifier learning. Please note that the learning scheme used in the second learning step is also part of the feature extraction training. In contrast to the second learning step described here, the embedded learning scheme works on an excerpt of the examples to estimate the accuracy and provide a fitness value (fitness evaluation). Further details are explained in Section 7.3 and 7.4. The approach is tested on the learning tasks of genre classification and user preferences (Section 7.4).

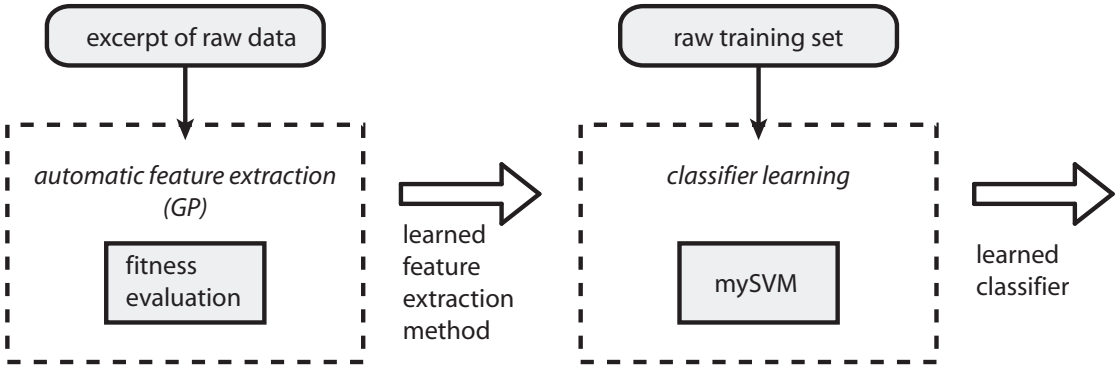



**Figure 7.1.:** The overall process of automatic feature construction for classification.

## 7.2. Methods for Feature Extraction

Audio data are time series, where the $y$-axis is the current amplitude corresponding to a loudspeaker's membrane and the $x$-axis corresponds to the time. They are univariate, finite, and equidistant. We may generalize the type of series which we want to investigate to *value series*. Each element $x_i$ of the series consists of two components. The first is the *index component*, which indicates a position on a straight line (e.g., time). The second component is a $m$-dimensional vector of values which is an element of the *value space*.

**Definition 7.1 (Value Series)** *A* VALUE SERIES *is a mapping* $v : \mathbb{N} \to \mathbb{R} \times \mathbb{C}^m$ *where we write* $v_i$ *instead of* $v(i)$ *and* $(v_i)_{i \in \{1,\ldots,n\}}$ *for a series of length* $n$*.*

This general definition covers time series as well as their transformations. All the methods described in the following refer to value series. They are not only applicable to audio data, but to value series in general. The usage of a complex number value space instead of a real number value space allows a convenient way to use basis transformations like the Fourier transformation. Finally, the introduction of the index component allows both equidistant and non-equidistant value series.

Feature extraction methods for value series can be described in a general framework. Such a framework offers the following advantages:

- Missing features for classification can be detected and their extraction methods be developed.
- Specialized methods can be decomposed into their general extraction methods:
  - the repository of general extraction methods can easily be implemented and extended, and

– combinations of extraction operators can be built, generating a large variety of feature sets.

In other words, organizing a repository of elementary feature extraction methods allows us to see the feature extraction for a certain learning task as a sequence of methods. The known methods are fixed sequences of such elementary extraction methods. Here, we give an overview of the building blocks, so that we later on can flexibly construct sequences. We now present the new notions of general windowing and interval mark-up together with the overall structure and some basic definitions.

### 7.2.1. Basis Transformations

Basis transformations map the data from the given vector space into another space. Audio data – like all univariate time series – are originally elements of the vector space $\mathbb{R}^2$. The basis $B$ of a vector space $V$ is a set of vectors which can represent all vectors in $V$ by their linear combination. The only required operation on vector spaces as the domain of transformations is the dot product. Since the most common basis transformation performed on audio data is the transformation into the infinite space of harmonic oscillations we assume *Hilbert spaces*.

**Definition 7.2 (Hilbert Space)** *Let $H$ be a vector space with an inner product $\langle f, g\rangle$. $H$ is called* HILBERT SPACE *if the norm defined by $||f|| = \sqrt{\langle f, f\rangle}$ turns $H$ into a complete metric space, i.e. any Cauchy sequence of elements of the space converges to an element in the space.*

The assumption of Hilbert spaces is no constraint, because all finite-dimensional spaces with a dot product (such as Euclidean space with ordinary dot product) are Hilbert spaces. In order to introduce the concept of Fourier transformations, we need an infinite-dimensional Hilbert space of functions.

**Definition 7.3 (Function Space)** *Let $P$ be a Hilbert space. If the elements $f \in P$ are functions, $P$ is called a* FUNCTION SPACE.

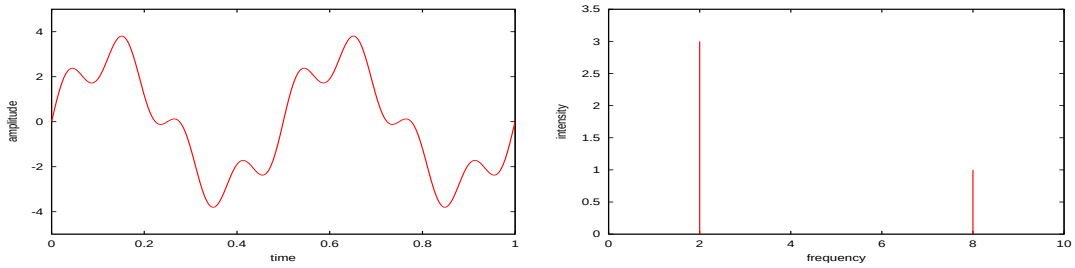
**Example 7.1 ($L^2$ as Function Space)** *The set of all functions $f : \mathbb{R} \to \mathbb{R}$ with a finite integral*

$$\int_{-\infty}^{\infty} f^2(a)da$$

*together with the inner product*

$$\langle f, g\rangle = \int_{-\infty}^{\infty} f(a)g(a)da$$

*form a well known function space: $L^2$.*




(a) Time space (b) Frequency space

**Figure 7.2.:** Overlay of two curves, $\nu_1 = 2Hz, a_1 = 3$ and $\nu_2 = 8Hz, a_2 = 1$, shown left in time space, right in frequency space after a Fourier transformation.

#### 7.2.1.1. Frequency Space

The goal of *Fourier analysis* is to write the series $(v_i)_{i\in\{1,\ldots,n\}}$ as a (possibly infinite) sum of multiples of the given base functions, which are $e^{-j\nu i}$ with $j$ as the imaginary unit, the (time) index $i$ and the frequencies $\nu$. A Fast Fourier Transformation [29] maps the given time space into this frequency space and is valid for audio data (Figure 7.2). The frequency space is a special case of a function space. The transformation uses the infinite number of complex valued dimensions of a Hilbert space. Complex numbers are necessary because Fourier transformations actually deliver two values: the intensity of occurring frequencies and the phase shifts. In general, the Fourier transformation can be seen as a projection of the series function on the base functions in the trigonometrical function space similar to that defined by the dot product in regular Euclidean spaces. The amplitudes (projection lengths) for all frequencies can be calculated by the following convolution:

$$FT(\nu) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} v_i e^{-j\nu i}\, di$$

where the real valued part of $FT(\nu)$ is the intensity corresponding to the frequency $\nu$ and the complex valued part of $FT(\nu)$ is the phase shift for this frequency.

#### 7.2.1.2. Correlation Space

The frequency space expresses a sort of correlation between values in terms of frequencies. For some features it would be more appropriate to express the correlation in terms of time dependencies. Therefore, the transformation into another space is used.

**Definition 7.4 (Correlation Space)** *The calculation of correlations of values between two points in time, $i$ and $i+k$, produce the* CORRELATION SPACE*, where for each lag $k$ their correlation coefficient in $[-1,+1]$ is indicated.*

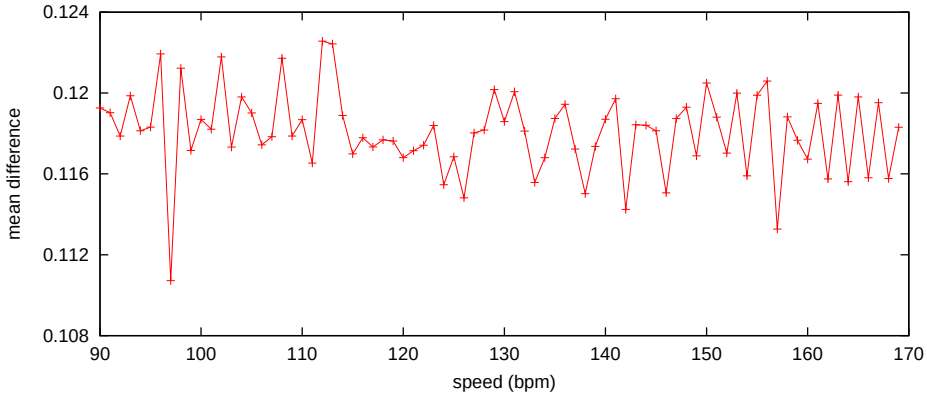



**Figure 7.3.:** Autocorrelation differences for a phase shift depending on speeds ranging from 90 to 170 beats per minute.

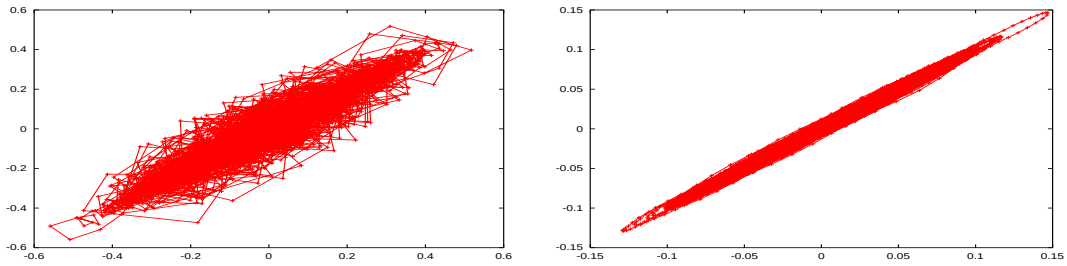
Transforming audio data into the correlation space eases the recognition of the speed of the music, measured in beats per minute. Assuming $T$ is the number of beats per measure (in pop music, the most commonly used value for $T$ is 4) and $SR$ the sampling rate of the audio file. If we shift the original time series by $shift = T \cdot SR \cdot 60/l$ for several values of $l$ in reasonable regions of $20 < l < 300$ we can determine the correlation between the original and the shifted time series. Maximal correlation corresponds to minimal difference between the shifted and the original series. Figure 7.3 shows the differences of original values with the shifted ones. Clearly, the difference at $l = 97$ beats per minute is minimal.

#### 7.2.1.3. Reconstruction of the State Space

Nonlinear dynamic systems can be described with the aid of non-linear differential equations. The number of variables which must be known to completely describe the behavior of such a system corresponds to the dimension of this system. These variables are called *state variables*.

**Definition 7.5 (State Space)** *The basis of the* STATE SPACE *of a dynamic system is given by the* STATE VARIABLES *of the system, i.e. the variables which must be known to describe the system. The elements of a state space represent the values of the state variables at the examined (time) points.*

The *state space* emphasizes characteristics which can hardly be seen in the original space. Since the state variables are often unknown, a topologically equivalent space is constructed [176]. This is known as *reconstruction of state space*.



(a) pop music (b) classic

**Figure 7.4.:** Phase space representation of a popular song (left) and a classical piece (right) created with the discussed state space reconstruction with $d = 1$ and $m = 2$.

**Definition 7.6 (Phase Space)** *Vectors within* PHASE SPACE *are constructed, where the components are parts of the original series:*

$$p_i = (v_i, v_{i+d}, v_{i+2d}, ..., v_{i+(m-1)d})$$

*where $d$ is the delay, and $m$ the dimension of the phase space. The set*

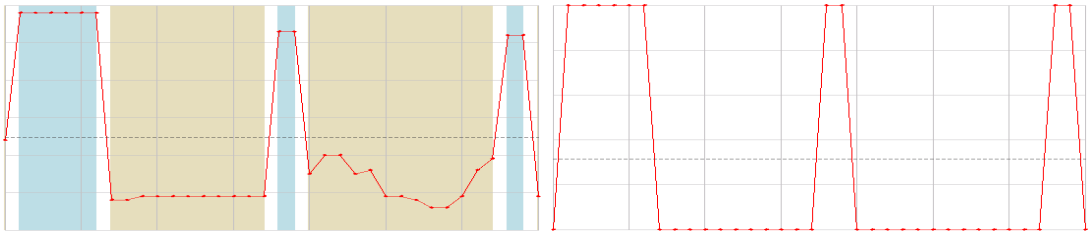
$$P_{d,m} = \{p_i \mid i = 1, ..., n - (m-1)d\}$$

*is the phase space representation of the original series $(v_i)_{i \in \{1,...,n\}}$.*

Within the phase space, several features can be extracted, e.g., the unordered angles between vectors. Small variances of angles indicate smooth changes of the state variables, large variances harsh changes. This is a dominant feature when separating classic from the more percussive pop music as shown in Figure 7.4.

#### 7.2.1.4. Reversibility

Basis transformations most often are reversible, because only the basis, not the position of the elements is changed. In contrast, if intervals in the index dimension are used, the transformation is not reversible. If, for instance, we summarize the original series by some time intervals and assign a value to each interval, the transformed series has still the same number of elements but fewer different values (see Figure 7.5 for illustration). We will discuss some possible ways to detect intervals in different dimensions of a value series in Section 7.2.3.



(a) Intervals in index dimension (b) Smoothing of intervals

**Figure 7.5.:** Intervals found in the index dimension are summarized.

### 7.2.2. Filters

Filters transform elements of a series to another location within the same space. Moving average and exponential smoothing, for instance, are filters. Many known transformations are subsumed by weighting functions. We consider the window functions Bartlett, Hanning, Hamming, Blackman-Harris, linear and exponential functions as particular instances of a function $f_w(i)$ which weighs the position within the window.

**Definition 7.7 (Weight Filter)** *Given a value series $(v_i)_{i\in\{1,\ldots,n\}}$, a filter $u_i = f_w(i) \cdot v_i$ is a* WEIGHT FILTER. *The weighting function $f_w$ only depends on the position $i$.*

For example, the often used Hanning filter can be written as

$$u_i = \left(0.5 - 0.5 \cdot \cos\frac{2\pi i}{n}\right) \cdot v_i$$

where $n$ is the total length of the series $(v_i)_{i\in\{1,\ldots,n\}}$. The Hanning filter gives high weights to the central parts of the series and lower weights to the border regions. This is for example desired to reduce the aliasing effects of Fourier transformations which are introduced by the finite lengths of the input series.

Other examples for this type of filters are frequency pass filters, filtering the extremes, the Bark-filter, and the ERB filter, which are all often used when analyzing music data.

### 7.2.3. Mark-up of Intervals

In analogy to mark-up languages for documents, which annotate segments within a text, also segments within a time series can be annotated.

**Definition 7.8 (Mark-Up)** *A* MARK-UP *$M : S \to C$ assigns an arbitrary characteristic $C$ to a segment $S$.*

We define special instances of mark-up by assigning characteristic value types to intervals in one of the dimensions of the considered space.

**Definition 7.9 (Interval)** *An* INTERVAL $I : S \to C$ *is a mark-up within one dimension. The segment* $S = (d, s, e)$ *is given by the dimension* $d$*, the starting point* $s$*, and the end point* $e$*. The characteristic* $E = (t, \varrho)$ *indicates a type* $t$ *and a density* $\varrho$*.*

Often clustering (e.g., k-means) is used in order to detect suitable intervals [63]. A clustering scheme is only usable in dimensions with a non-equidistant value distribution. Additionally, clustering in one or several dimensions is a batch process to be applied to the complete series. An incremental process is the signal to symbol process [131]:

**Signal to symbol processing:** Given the series $(v_i)_{i\in\{1,\ldots,n\}}$ with $n$ values, a decision function $f_e$ and an interval dimension,

initialize the interval counter with $t = 1$, start a new interval $I_t$ and add the first point.

For the remaining points of the series, do:

1. If $f_e(I_t, v_i) = 1$, then add $v_i$ to the current interval $I_t$.
2. Else close $I_t$, increase $t$ by 1, and add $v_i$ to the new $I_t$.

Typical examples of the decision function $f_e$ refer to the gradient, delivering characteristics such as, e.g., *increase, decrease*. Signal to symbol processing is applied to the index dimension (time). If intervals have already been found in the value dimension, these can be used to induce intervals in the index dimension. For instance, whenever a interval change in the value dimension has been found, the current interval in the index dimension is closed and a new one is started. Figure 7.6 illustrates this combination.

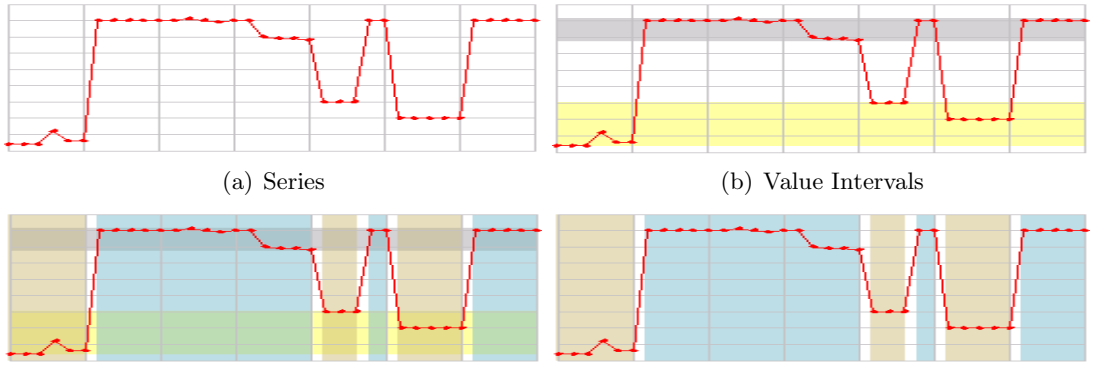
### 7.2.4. Generalized Windowing

Many known operators on times series involve windowing. Separating the notion of windows over the index dimension from the functions applied to the values within the window segment allows to construct many operators of the kind.

**Definition 7.10 (Windowing)** *Given the series* $(v_i)_{i\in\{1,\ldots,n\}}$*, a transformation is called* WINDOWING*, if it shifts a window of width* $w$ *over* $(v_i)_{i\in\{1,\ldots,n\}}$ *using a step size* $s$ *and evaluates in each window the function* $f$*:*

$$u_j = f((v_i)_{i\in\{j\cdot s+1,\ldots,j\cdot s+w\}}).$$

*All* $u_j$ *together form again a series* $(u_j)_{j\in\{0,\ldots,\lfloor(n-w)/s\rfloor\}}$*.*



(a) Series (b) Value Intervals

(c) Index intervals (d) Result

**Figure 7.6.:** The process of finding intervals in a series (a), first in the value dimension (b), then projected on the index dimension (c), delivering (d).

**Definition 7.11 (General Windowing)** *A windowing which performs an arbitrary number of transformations in addition to the function $f$ is called* GENERAL WINDOWING.

The function $f$ summarizes values within a window and thus prevents general windowing from enlarging the data set too much. Since the size of audio data is already rather large, it is necessary to consider carefully the number of data points which is handled more than once.

**Definition 7.12 (Overlap)** *The* OVERLAP *of a general windowing with step size $s$ and width $w$ is defined as $g = w/s$.*

Only for windowings with overlap $g = 1$ the function can be omitted. Such a windowing only performs transformations for each window and is called *piecewise filtering*. Section 7.2.6 investigates the runtime effects of transformations used within general windowing and the overlap.

Combining general windowing with the mark-up of intervals allows to consider each interval being a window. This results in an adaptive window width $w$ and no overlap, i.e. $g = 1$. Of course, this speeds up processing considerably.

### 7.2.5. Functions

Transformations convert a series into another series. In contrast, functions calculate single values from a series. The group of functions includes all kinds of statistics like different averages, variance and standard deviation. They refer to the value dimension. We may also consider the index dimension, for instance, the point with the largest value or highest amplitude. Often used functions are those indicating peaks.

**Definition 7.13 ($k$-Peaks function)** *The $k$-peak function delivers the position (index dimension), the height, and the width of the $k$ largest peaks of a series $(v_i)_{i\in\{1,\ldots,n\}}$.*

It is an instance of finding extremes (minimum, maximum). Similarly, the gradient of a regression line can be formulated. For audio data, the spectral flatness measure or the spectral crest factor can be expressed as an arithmetic combination of simple functions [71].

The widely used *Mel-frequency cepstral coefficients (MFCCs)* can also be constructed as a general windowing. The MFCCs are a spectrum of a specifically transformed frequency spectrum, hence the word *cepstral* as a reversed form of the word *spectral*. The difference between a traditional cepstrum (spectrum of a spectrum) and the mel-frequency cepstrum is that in the case of MFCCs the frequency bands are equally spaced on the so-called mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

1. apply a general windowing,
2. apply a window function like a Hanning or Hamming function on each window,
3. calculate the Fourier transform of each window,
4. map the values of the obtained spectrum for each window onto the mel scale with $mel = 1127.01048 \cdot \log(1 + \nu/700)$,
5. take the logs of the powers at each of the mel frequencies,
6. take the discrete cosine transform of the list of mel log powers as if it were a signal,
7. the MFCCs are the amplitudes of the resulting spectrum.

There can be variations on this process, for example, differences in the shape or spacing of the windows used to map the scale.

Figure 7.7 shows how the methods for feature extraction are put together to compute the cepstral coefficients. From these coefficients additional features can be extracted. It is easy to see how variants of this series can be generated, e.g., replacing the frequency spectrum and its logarithm by the gradient of a regression line.

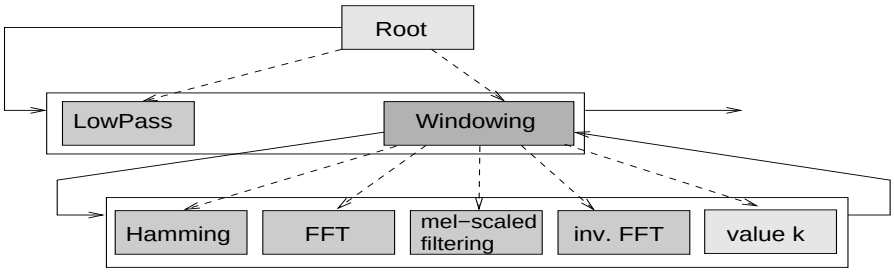



**Figure 7.7.:** Constructing the cepstral method from elementary extraction operators.

### 7.2.6. Some Properties of the Methods

Since large amounts of data have to be processed, each method must be fast. Of course, the mere sequence of methods is then fast, too. What needs to be checked is the effect of general windowing. How much does the runtime increase, if we apply methods running in $O(n^2)$ within the windows? How many data points are handled more than once? If the step size $s$ is 1, $k$ windows of width $n-k$ can be built. The number of values to be processed increases from $n$ to $k \cdot (n-k)$. The value $k = n/2$ maximizes this product. In other words, for a step size of 1 the worst case is a window size of $w = n/2$ for a series of length $n$. Then $g = n/2$ values are handled more than once.

In general, the number of windows for a series of length $n$ and a step size $s$ is

$$k = \frac{n-w}{s} + 1 = \frac{n}{s} - g + 1 \tag{7.1}$$

We assume that all transformations occur only once in a sequence of methods and that the number of methods is finite. The resulting runtime for methods in $O(n)$ is generally stated by Lemma 7.1.

**Lemma 7.1 (Windowing on Linear Methods)** *General windowing using methods in $O(n)$ has the worst case runtime of $O(gn - gw + w)$, where $g$ is the overlap.*

*Proof.* Processing data within a window costs $O(w)$. $k$ windows are to be processed. The overall runtime is $k \cdot w$:

$$\begin{aligned} k \cdot w &= \left(\frac{n}{s} - g + 1\right) \cdot w \\ &= \frac{n}{s}w - gw + w \\ &= gn - gw + w \end{aligned}$$

□

Lemma 7.1 means, that for $g = 1$, a series is processed in $O(n)$. For $g = 2$ the effort becomes $2n - w$ which is more expensive than processing the methods on the overall series since $w$ is always less than $n$. If $g = n/2$, the general windowing effort becomes $\frac{n^2}{2} - \frac{wn}{2} + w$ which is an order of magnitude larger than the runtime of the inner methods applied to the whole series.

Analogously, we state the runtime for general windowing for methods with logarithmic complexity, with quadratic complexity, and those with exponential complexity.

**Lemma 7.2 (Windowing on Methods with $O(n \log n)$)** *Assuming that each method is applied at most once, general windowing using methods in $O(n \log n)$ has the worst case runtime of $O(gn \log w - gw \log w + w \log w)$, where $g$ is the overlap, $n$ the length of the series, and $w$ the width of the windows.*

*Proof.* Processing data within a window costs $O(w \log w)$. $k$ windows are to be processed. The overall runtime is $k \cdot w \log w$:

$$\begin{aligned} k \cdot w \log w &= \left(\tfrac{n}{s} - g + 1\right) \cdot w \log w \\ &= \tfrac{n}{s} w \log w - wg \log w + w \log w \\ &= gn \log w - gw \log w + w \log w \end{aligned}$$

□

**Lemma 7.3 (Windowing on Quadratic Methods)** *Assuming that each method is applied at most once, general windowing using methods in $O(n^2)$ has the worst case runtime of $O(gnw - gw^2 + w^2)$, where $g$ is the overlap, $n$ the length of the series, and $w$ the width of the windows.*

*Proof.* Processing data within a window costs $O(w^2)$. $k$ windows are to be processed. The overall runtime is $k \cdot w^2$:

$$\begin{aligned} k \cdot w^2 &= \left(\tfrac{n}{s} - g + 1\right) \cdot w^2 \\ &= \tfrac{n}{s} w^2 - gw^2 + w^2 \\ &= gnw - gw^2 + w^2 \end{aligned}$$

□

**Lemma 7.4 (Windowing on Polynomial Methods)** *Assuming that each method is applied at most once, general windowing using methods in $O(n^p)$ has the worst case runtime of $O(gnw^{p-1} - gw^p + w^p)$, where $g$ is the overlap, $n$ the length of the series, and $w$ the width of the windows.*

*Proof.* Processing data within a window costs $O(w^p)$. $k$ windows are to be processed. The overall runtime is $k \cdot w^p$:

$$\begin{aligned} k \cdot w^p &= \left(\tfrac{n}{s} - g + 1\right) \cdot w^p \\ &= \tfrac{n}{s} w^p - g w^p + w^p \\ &= g n w^{p-1} - g w^p + w^p \end{aligned}$$

□

We discuss several values for the overlap $g$. For $g = 1$ the terms with $w^p$ cancel out and the total running time is $n \cdot w^{p-1}$ which is smaller than $n^p$ for all values $w < n$. For a more realistic windowing with $g = 2$ the total effort is $2nw^{p-1} - w^p$ which is also smaller than $n^p$ for all $w = n$. The total effort for an overlap of $\frac{n}{2}$ is the worst and is calculated as

$$\begin{aligned} g n w^{p-1} - g w^p + w^p &= n \cdot \frac{n}{2} \cdot \left(\frac{n}{2}\right)^{p-1} - \left(\frac{n}{2}\right)^p + \left(\frac{n}{2}\right)^p \\ &= n \cdot \left(\frac{n}{2}\right)^p - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= n \cdot \frac{n^p}{2^p} - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= \frac{n^{p+1}}{2^p} - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= \frac{2n^{p+1}}{2^{p+1}} - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= \frac{n^{p+1}}{2^{p+1}} + \left(\frac{n}{2}\right)^p \\ &= \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \end{aligned}$$

The total running time for this large overlap is always one degree larger than applying the same methods without windowing on the complete value series. However, values for $g$ larger than 2 are seldomly used for windowings for feature extraction.

**Lemma 7.5 (Windowing on Exponential Methods)** *Assuming that each method is applied at most once, general windowing using methods in $O(a^n)$ has the worst case runtime of $O(\frac{gn}{w} a^w - g a^w + a^w)$, where $g$ is the overlap, $n$ the length of the series, and $w$ the width of the windows.*

*Proof.* Processing data within a window costs $O(a^w)$. $k$ windows are to be processed.

The overall runtime is $k \cdot a^w$:

$$\begin{aligned} k \cdot a^w &= \left(\tfrac{n}{s} - g + 1\right) \cdot a^w \\ &= \tfrac{n}{s} a^w - g a^w + a^w \\ &= \tfrac{gn}{w} a^w - g a^w + a^w \end{aligned}$$

□

It can easily be seen that for all but the linear complexity methods, the general windowing is faster than applying the methods to the overall series, if $g = 1$, and at least as fast if $g = 2$. If $g = n/2$, the runtime for windowing is worse than the runtime for applying the methods to the overall series except for exponential inner methods[2].

## 7.3. Adaptive Construction of Method Trees

The elementary methods described above are combined in order to construct more complex features for classification tasks. Figure 7.7 already showed how elementary methods can be used for the reconstruction of known complex feature extraction methods. There are many more complex feature extraction methods which can be built using the framework described above (Section 7.2). For instance, the general windowing may apply a Fourier transformation $FT$ so that the peaks of the transformed series can be related with windows in time:

$$u_j = \max_{index}(FT(\{v_i\}_{i \in \{j \cdot s+1, \ldots, j \cdot s+w\}}))$$

The result is a value series, where the value of $u_j$ denotes the highest frequency for each window. From this series, the average and variance is built, yielding a good feature for the separation of techno and pop music – the variance is greater in pop music.

It is rather cumbersome to find such combinations that perform well for a classification task. We are looking for chains of method applications. Moreover, there might be some windowing within which such chains are applied. This is a rather large search space (see Section 7.3.3). It is too large to be inspected manually. Hence, *genetic programming* is applied in order to look for the best combination of methods [64, 91]. The result is a complex method consisting of a combination of the elementary methods presented above. Its use for the classifier learning will be shown in Section 7.4.

In order to structure the huge search space, we may separate functions, chains of method applications, and general windowing, where a chain of method applications is applied to each window.

[2] When constructing features from audio data, exponential methods or too large overlaps are not used.

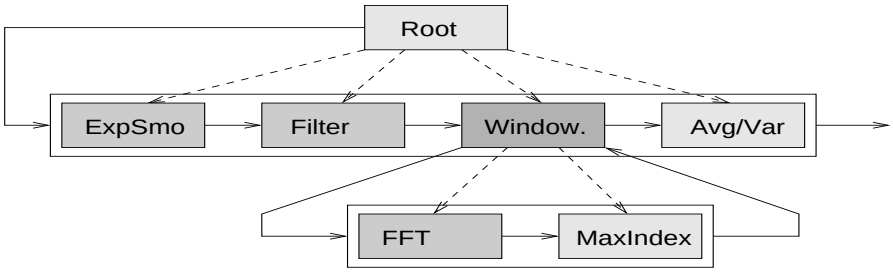



**Figure 7.8.:** A method tree for feature extraction built of elementary methods. Solid arrows show the data flow, dashed lines define the tree structure.

**Definition 7.14 (Chain)** *A* CHAIN *consists of an arbitrary number of transformations and a function at the end.*

A function is a chain with no transformations. It has the length 1. A longer chain consists of some transformations followed by a function. In any case, a chain delivers one value. Incorporating windowings leads to the concept of method trees:

**Definition 7.15 (Method Tree)** *A* METHOD TREE *is a general windowing whose children build a chain. If the chain entails a windowing, this becomes the root of a new, embedded method tree.*

The methods which are performed on each window can be seen as children of the windowing operator. Together they output a value series. The tree structure emerges from the nesting of windowing operators.

An example of a method tree is shown in Figure 7.8, where the root identifies the element within the search space. Its four children are exponential smoothing, a filtering, another method tree consisting of the chain just described (Fourier transformation with peaks applied to windows), and the average of the peaks. This last child returns the desired features.

Before the genetic programming approach is technically described, Figure 7.9 presents the process of automatically extracting features for a given classification task and data set. The picture details on the first box of Figure 7.1 above which shows the overall process. The search space within which the best method tree is to be found is called the universe of method trees. A population is a set of method trees. The navigation within the universe of method trees is a cycle of selecting a population, applying the method trees to the raw data, evaluating the fitness of the population, and enhancing the fittest method trees further to build a new population (Section 7.3.2). This cycle corresponds to the standard process of genetic algorithms. What differs from the standard is that method trees instead of binary vectors form the search space, that the search space is

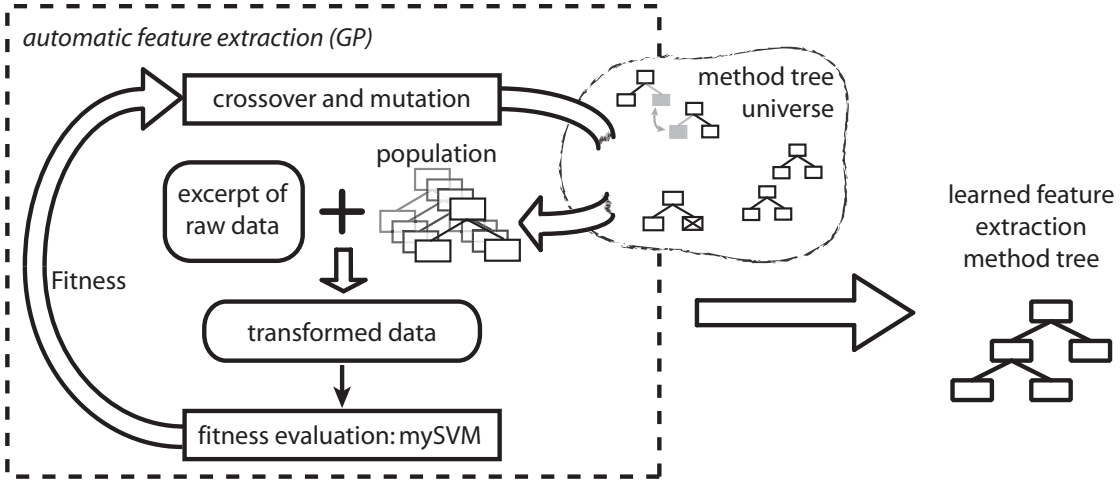



**Figure 7.9.:** Automatic feature extraction using genetic programming.

structured, and that the fitness evaluation is not merely a function but the result of running another learning algorithm.

### 7.3.1. Representation

Genetic programming constructs finite automata. Here, method trees are to be constructed. They are represented by XML expressions. Figure 7.10 shows the representation of the method tree from Figure 7.8. The RapidMiner system executes such trees and takes care of the syntactic well-formedness.

The restriction that chains are concluded by a function implies a level-wise structure of all possible method trees. The lowest level 1 entails only functions. These are chains of length 1. The next level, 2, covers chains with a concluding function. Levels 3 and above entail windowing. Method trees are constructed according to their levels. The level-wise growing means small changes to a current method tree. On the one hand, this reduces the probability of missing the optimal method tree. On the other hand, it may slow down the search, if the fitness of the lower levels does not distinguish between good and bad method trees.

### 7.3.2. Mutation, Crossover, and Selection

The operations of genetic programming are mutation and crossover. By random, mutations insert a new method, delete a method, or replace a method by one of the same class, i.e. by a function or transformation. Crossover replaces a sub-tree from one method tree

```
<operator name="Root" class="ValueSeriesPreprocessing">
  <operator name="Chain 1" class="OperatorChain">
    <operator name="ExpSm" class="ExponentialSmoothing" />
    <operator name="Filter" class="FilterTransformation" />
    <operator name="Windowing" class="Windowing">
      <parameter key="overlap"   value="2"/>
      <operator name="Chain 2" class="OperatorChain">
        <operator name="FFT" class="FastFourierTransform" />
        <operator name="MaxIndex" class="MaxIndexPoint" />
      </operator>
    </operator>
    <operator name="Avg" class="AverageFunction" />
  </operator>
</operator>
```

**Figure 7.10.:** XML method tree representation for RapidMiner.

by a sub-tree from another method tree, respecting the well-formedness conditions. This means that the roots of the sub-trees must be of the same type of methods.

For selection purposes, the fitness of all method trees is expressed by a *roulette wheel*, i.e. fitness proportional parts of a wheel's 360 degrees. The larger the portion, the more likely it becomes that the particular individual is selected for the next generation or crossover. Other possible selection schemes include *tournament selection*, where $t$ method trees are randomly chosen from the population and the winner of this set, i.e. the method tree with the largest fitness, is added to the next generation. The parameter $t$ allows the definition of selection pressure. In our experiments, we use a tournament selection with a tournament fraction size of 0.25.

#### 7.3.2.1. Fitness Evaluation

Since method trees serve classification in the end, the quality of classification is the ultimate criterion of fitness (wrapper approach). Individuals which provide better classification results when used as features for the classification task at hand should have a greater probability to survive into the next generation. To evaluate the fitness of each method tree in a population the following steps are performed:

1. Each individual method tree is applied to an excerpt of the raw data.
2. This method application returns a transformed data set, which is used by classifier learning.
3. A $k$-fold cross validation is executed to estimate the performances of the learning scheme in combination with the feature sets provided by the method trees.

4. The mean accuracy, recall, and/or precision of the result becomes the fitness value of the applied feature construction method trees.

5. The fitness values of the method trees are used to build the next population with one of the selection schemes described above.

### 7.3.3. Some Properties of the Search Space

Automatically constructing methods for feature extraction which deliver well suited feature sets for classifier learning is a demanding task. How fast can we expect a good result to be found? This general question can be split into three more specific ones. First, the size of the search space is important. Second, the complexity of processing one individual in the search space helps to bound the overall complexity of search. Third, the convergence to an optimum determines the speed of the genetic programming. The last issue is not yet solved. Even simple evolutionary algorithms demand complicated proofs [40]. Here, we answer the first and second question.

#### 7.3.3.1. Size of the Search Space

If all mathematical operations were allowed within method trees, the search space would become infinite, hence only a fixed set of transformations and functions are allowed. The following theorem states an upper bound for the size of the search space.

**Theorem 7.1 (Size of the Search Space)** *The size of the search space of all method trees is upper bounded by*

$$F \cdot \sum_{k=0}^{T_{n-1}} \frac{T_{n-1}!}{(T_{n-1}-k)!}. \tag{7.2}$$

*Proof.* Let $T_0$ be the number of transformations and $F$ the number of functions, $n$ the length of the input series. The number of possible methods at level 1 becomes $F$. At level 2, transformations could be applied once in any order. The two levels can be summarized. For chains of length $k$, there exist the following number of different chains:

$$K_0 = F \cdot \sum_{k=0}^{T_0} \frac{T_0!}{(T_0-k)!} \tag{7.3}$$

The higher levels are produced by windowing. Within the windows, a chain of transformations is executed, if no nested windowing is allowed. Hence, there exist as many

windowing operations as there are chains (equation 7.3). Adding the number of windowed transformation chains $K_0$ to the other transformations $T_0$ returns the number of method trees at level 3:

$$T_1 = T_0 + K_0 = T_0 + F \cdot \sum_{k=0}^{T_0} \frac{T_0!}{(T_0 - k)!} \tag{7.4}$$

For nested structures, the recursive structure can be illustrated by equation 7.5:

$$T_2 = T_0 + K_1 = T_0 + F \cdot \sum_{k=0}^{T_1} \frac{T_1!}{(T_1 - k)!} \tag{7.5}$$

The depth of nested windowing is restricted by the length of the series: after $n - 1$ levels of embedded windowing, there is no data of a series with $n$ points left for further windowings. Hence, the overall size of the search space of all method trees is upper bounded by:

$$F \cdot \sum_{k=0}^{T_{n-1}} \frac{T_{n-1}!}{(T_{n-1} - k)!} \tag{7.6}$$

□

Each element in the search space delivers as many features as are determined by the concluding function. If genetic programming has to construct more features, it can either be applied several times, or transformations can be applied more than once. In the latter case, equation 7.2 states the lower bound of the search space size.

#### 7.3.3.2. Processing a Method Tree

Until now we have ignored that the window size of embedded windowings must become smaller for increased depth of embedding. Regarding the embedded windowing operators leads to the notion of dynamic windowing.

**Definition 7.16 (Dynamic Windowing)** *Let $(v_i)_{i\in\{1,\dots,n\}}$ be the original value series of length $n$ and $d \in \{2, \dots n/2\}$. Windowing with overlap $g$, width $w = n/d$ and step size $s = n/gd$ is called* DYNAMIC WINDOWING.

The maximal depth of a method tree can now be determined.

**Lemma 7.6 (Maximal Depth)** *Given a value series $(v_i)_{i\in\{1,\dots,n\}}$ of length $n$, a method tree using dynamic windowing cannot exceed the depth of* $\log_d n - 1$.

*Proof.* Dynamic windowing splits the series into windows of width $n/d$. The width depends on the length of the series as well as on parameter $d$. For embedded windowing, only $n/d$ values are available. Windows on this smaller series have a window width of $n/d^2$. Only $\log_d n - 1$ repetitions are possible. The last embedding of windowing with width

$$\frac{n}{d^{\log_d n - 1}} = \frac{n}{\frac{d^{\log_d n}}{d}} = \frac{nd}{n} = d$$

does not allow any further windowing, because then only one value would remain for a window. □

Remember that the number of windows with overlap $g$ on a series of length $n$ is $n/s - g + 1$ (equation 7.1). Combining the maximal depth of a method tree with the number of windows and their computation efforts estimates the worst runtime complexity of a method tree.

**Theorem 7.2 (Runtime of a Method Tree)** *Let $CM(n)$ be the complexity of applying an internal method of at most quadratic time complexity to a series of length $n$. Using dynamic windowing, no method tree requires a runtime which is exponential in the length of the series $(v_i)_{i \in \{1,\dots,n\}}$.*

*Proof.* The number of windows times the effort per window determines the overall effort. Using equation 7.1 this is for a first level:

$$\left(\frac{n}{s} - g + 1\right) \cdot CM\left(\frac{n}{d}\right) = (g(d-1)+1) \cdot CM\left(\frac{n}{d}\right)$$

Embedding a further windowing delivers at level $i$ the following effort estimation:

$$(g(d-1)+1)^i \cdot CM\left(\frac{n}{d^i}\right)$$

Using Lemma 7.6 for the bound of $i$ results in the total effort:

$$\begin{aligned} & (g(d-1)+1)^{\log_d n - 1} \cdot CM\left(\frac{n}{d^{\log_d n - 1}}\right) \\ = \; & (g(d-1)+1)^{\log_d n - 1} \cdot CM\left(\frac{n \cdot d}{n}\right) \\ = \; & (g(d-1)+1)^{\log_d n - 1} \cdot CM(d) \end{aligned} \tag{7.7}$$

For the worst case of $CM(d) = d^2$, equation 7.7 becomes:

$$\begin{aligned} & (g(d-1)+1)^{\log_d n - 1} \cdot d^2 \\ = \; & \frac{d^2}{g(d-1)+1} \cdot n^{\frac{1}{\log_{g(d-1)+1} d}} \\ = \; & \frac{d^2}{g(d-1)+1} \cdot n^{\log_d (g(d-1)+1)} \end{aligned}$$

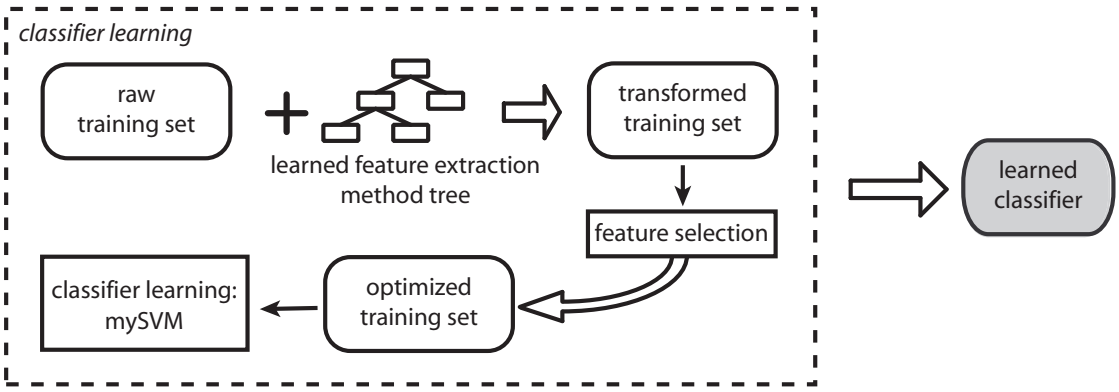



**Figure 7.11.:** Classifier learning step using the best method tree found by the genetic programming approach.

As is easily seen, the runtime is not exponential in the length of the series, but is limited by the overlap and the parameter $d$ and is therefore pseudo-polynomial. □

Dynamic windowing avoids a particular case with exponential effort, which otherwise could easily be constructed. If, for instance, the series was divided into two windows at each level with a fixed step size but dynamic window width, the effort would be $2^i \cdot O(\frac{n}{2^i})$ at the $i$-th level. After $n-1$ splits, no further embedding of windowing is possible, since only two values are left. Hence, the overall effort would be $2^n$. This exponential construction, however, does not obey dynamic windowing with fixed overlap and dynamic width. Hence, it cannot happen in our scenario which is one of the major results of this chapter.

## 7.4. Classification Using Learned Method Trees

Automatic feature construction aims at good results of a second learning step which uses the features, namely classifier learning. Remember Figure 7.1 from the introduction, where genetic programming were presented to deliver the input to classifier learning. Now, Figure 7.11 details the second box of the overall picture.

Feature construction is already guided by the classification task in that cross-validated learning determines the fitness of method trees (individuals of genetic programming). Now, also feature selection is performed by a simple evolutionary method, namely the (1+1)EA [6]. Again, the classification task decides upon the fitness. The feature set is built using a subset of the training data. The selected method trees are then applied to all the training data. The support vector machine mySVM is applied to these rewritten data and learns a classifier.

| | **Classic/Pop** | **Techno/Pop** | **Hiphop/Pop** |
|---|---|---|---|
| Accuracy | 100% | 93.12% | 82.50% |
| Precision | 100% | 94.80% | 85.27% |
| Recall | 100% | 93.22% | 79.41% |
| Error | 0% | 6.88% | 17.50% |

**Table 7.1.:** Classification of genres with a linear SVM using the task specific feature sets.

### 7.4.1. Classifying Genres

Since results are published for the genre classification task, we have applied our approach to this task, too. Note, however, that no published benchmark data sets exist. Hence, the comparison can only show that feature construction and selection leads to similar performance as achieved by other approaches. For the classification of genres, three data sets have been built.

- Classic/pop: 100 pieces for each class were available in Ogg Vorbis format.
- Techno/pop: 80 songs for each class from a large variety of artists were available in Ogg Vorbis format.
- Hiphop/pop: 120 songs for each class from few records were available in MP3 format with a coding of 128 kbits/s.

The classification tasks are of increasing difficulty. Using mySVM with a linear kernel, the performance was determined by a 10-fold cross validation and is shown in Table 7.1. Concerning classic vs. pop, 93% accuracy, and concerning hiphop vs. pop, 66% accuracy have been published [181, 182].

41 features have been constructed for all genre classification tasks. For the distinction between classic and pop, 21 features have been selected for mySVM by the evolutionary approach. Most runs selected features referring to the phase space (angle and variance). The use of features can also be inspected by restricting a top-down induction of decision trees to a few levels. For a one level stump, 93% accuracy could be achieved by just using the RMS volume, i.e. the root mean square average of the series.

For the separation of techno and pop, 18 features were selected for mySVM, the most frequently selected ones being the filtering of those positions in the index dimension where the curve crosses the zero line. The decision tree starts with a phase space feature, the average of angles. A one level stump uses the starting value of the second frequency band, giving a benchmark of 76% accuracy.

For the classification into hiphop and pop, 22 features were selected with the mere volume being the most frequently selected feature. The decision tree classifying hiphop

| | **Classic/Pop** | **Techno/Pop** | **Hiphop/Pop** |
|---|---|---|---|
| Accuracy | 96.50% | 64.38% | 72.08% |
| Precision | 94.12% | 60.38% | 70.41% |
| Recall | 95.31% | 64.00% | 67.65% |
| Error | 3.50% | 35.63% | 27.92% |

**Table 7.2.:** Classification performance using the same non-tailored standard feature set for all classification tasks (linear SVM).

| | **Classic/Pop** | **Techno/Pop** |
|---|---|---|
| SVM (linear) | 0.00% | 6.88% |
| SVM (rbf) | 1.50% | 14.38% |
| C4.5 | 0.00% | 7.50% |
| k-NN | 3.00% | 9.38% |
| Naive Bayes | 2.50% | 10.63% |

**Table 7.3.:** Classification errors with respect to different learning schemes.

against pop is rather complex. It starts with the length of the songs. Experiments with naive Bayes and k-NN did not change the picture: an accuracy of about 75% can easily be achieved, increasing the performance further demands better features.

In order to demonstrate the effect of tailored feature sets for each classification task we performed experiments with the same feature set for all data sets. We used only features which were used in at least 50% of all subsets produced by feature selection for all data sets to simulate a reasonable standard feature set. Table 7.2 shows the classification performance for a linear SVM estimated with a 10-fold cross validation. The performance is significantly lower than the performance which can be achieved using the tailored feature sets (see Table 7.1).

Table 7.3 shows the achieved classification errors with respect to different learning schemes. Since the extraction of features and the transformation in another feature space is performed by the applied method tree, the usage of a linear kernel function is actually no restriction. Therefore, we use a linear SVM for all our experiments and as inner learner to estimate the fitness of the method trees. The conclusions which can be drawn from Table 7.2 and 7.3 indicate that a tailored set of task specific features and not the quality of the learning scheme is the crucial aspect for the successful classification of audio data.

| | $\mathbf{User}_1$ | $\mathbf{User}_2$ | $\mathbf{User}_3$ | $\mathbf{User}_4$ |
|---|---|---|---|---|
| Accuracy | 95.19% | 92.14% | 90.56% | 84.55% |
| Precision | 92.70% | 98.33% | 90.83% | 85.87% |
| Recall | 99.00% | 84.67% | 93.00% | 83.74% |
| Error | 4.81% | 7.86% | 9.44% | 15.45% |

**Table 7.4.:** Classification according to user preferences.

### 7.4.2. User Preferences

Recommendations of songs to possible customers are currently based on the individual correlation of record sales. This collaborative filtering approach ignores the content of the music. A high correlation is only achieved within genres, because the preferences traversing a type of music are less frequent. The combination of favorite songs into a set is a very individual and rare classification. It is not a generalization of many instances. Therefore, the classification of user preferences beyond genres is a challenging task, where for each user the feature set has to be learned. Of course, sometimes a user is interested only in pieces of a particular genre. This does not decrease the difficulty of the classification task. In contrast, if positive and negative examples stem from the same genre, it is hard to construct distinguishing features. Genre characteristics might dominate the user-specific features. As has been seen in the difficulty of the data set for hiphop vs. pop, sampling from few records also increases the difficulty of learning. Hence, four learning tasks of increasing difficulty have been investigated.

Four users brought 50 to 80 pieces of their favorite music ranging through diverse genres. They also selected the same number of negative examples. User 1 selected positive examples from rock music with a dominating electric guitar. User 2 selected positive as well as negative examples from jazz music. User 3 selected music from classic over latin and soul to rock and jazz. User 4 selected pieces from different genres but only from few records. Using a 10-fold cross validation, mySVM was applied to the constructed and selected features, one feature set per learning task (user). Table 7.4 shows the results.

The excellent learning result for a set of positive instances which are all from a certain style of music corresponds to our expectation (user 1). The expectation that learning performance would strongly decrease if positive and negative examples are taken from the same genre is not supported (user 2). Surprisingly well is the learning result for a broad variety of genres among the favorites (user 3). This fact indicates that for this user the constructed feature set supports the building of preference clusters in feature space instead of dominating genre clusters. In contrast to this result the (negative) effect of sampling from few records can be seen clearly (user 4). Applying the learned decision function to a database of records allowed the users to assess the recommendations. They were found very reasonable. No particularly disliked music was recommended, but unknown plays and those, which could have been selected as the top 50.

Of course, this method of user preference recognition is just the first step. There are several ways to improve the results. For instance, users could indicate those examples that must be classified correctly, because they feel that it is an essential expression of their taste. Weighting examples as a further cost function for learning has been investigated in [86]. The inspection of the chosen features by the users themselves is not yet possible. Regular users are not acquainted with Fourier transformation and peaks, for instance, but would like to see understandable and interpretable features. For this reason, the author of this work has co-developed a large-scale feature construction based approach which delivers aggregated high-level descriptors based on massive amounts of phase space features [128].

## 7.5. Multi-Objective Feature Extraction

In the last chapter, we have defined a regularized risk for feature space transformations:

$$R_{reg}^{FST}(X) = R_{inner} + \lambda\Omega(X).$$

We have seen in this chapter that the used wrapper approach again delivers the risk $R_{inner}$. We have already used this performance as a guide for the evolutionary feature extraction. Now, an appropriate measurement for the *feature space complexity* $\Omega(X)$ has to be defined. For multi-objective supervised feature selection and multi-objective supervised feature construction, we have already seen that the number of used features is a simple and good measurement for the feature space complexity.

Since supervised feature extraction is only a special case of supervised feature construction, we can again use the number of extracted features as a complexity measure for the resulting feature space. Hence, we can still apply Lemma 6.1 and the number of features which is extracted should be minimized in order to minimize the regularized risk $R_{reg}^{FST}(X)$.

The final two criteria for multi-objective supervised feature extraction hence are the performance of the inner supervised learner $R_{inner}$ which should be maximized together with the number of used features $nf$ which should be minimized. We extend the genetic programming approach discussed above by using these criteria as a base for the non-dominated sorting selection scheme known from NSGA-II.

### 7.5.1. Experiments and Results

For the audio classification experiments, it turned out that a (1+1)EA working on one individual only already delivers good classification results. Unfortunately, the necessary runtime for this single-individual optimization approach already is very high on these

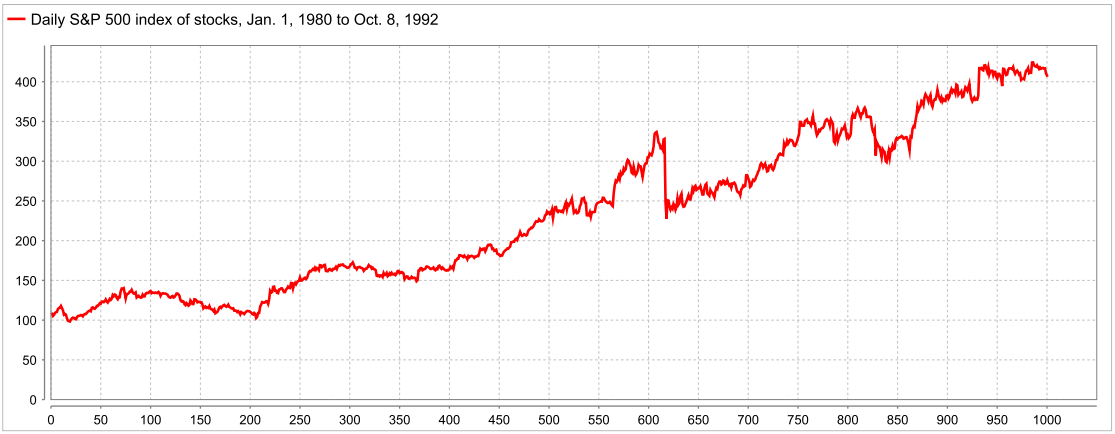



**Figure 7.12.:** The daily S&P 500 index between January 1st, 1980 and October 8th, 1992. The task is to predict the value for the next day from the values which were encountered in the past.

large amount of data and using an appropriate number of individuals in order to get complete Pareto fronts would no longer be feasible. Therefore, we use another and much smaller data set for the multi-objective feature extraction experiment discussed here, namely the daily S&P 500 data set consisting of all values of this index between January 1st, 1980 and October 8th, 1992. The task is that of a time series prediction, i.e. the value for the next day should be predicted from the values which were encountered in the past. Figure 7.12 shows the complete series.

A windowing is applied on this data set with window width 80 and step size 20. The next value after the last value in the window is used as label, i.e. the prediction horizon is 1. The original series has a length of 3333 values. After the windowing, the resulting example set has 163 examples with 80 attributes plus the label attribute. We applied the genetic programming approach discussed in this chapter on this data set. The number of individuals in the population is 15, the maximum number of generations was restricted to 50.

### 7.5.2. Interpretation of the Pareto Front

Figure 7.13 shows the Pareto front in generation 11. Although not all dominated points are removed yet, the Pareto front can already be seen. At the end of the optimization process, only non-dominated points are part of the final Pareto set.

The fitness is evaluated by a 10-fold back-testing validation, i.e. the data set is divided into 10 parts where the first $k$ parts are (cumulatively) used for training and the part $k+1$ is used for testing. The performance criterion root relative squared error is calculated as

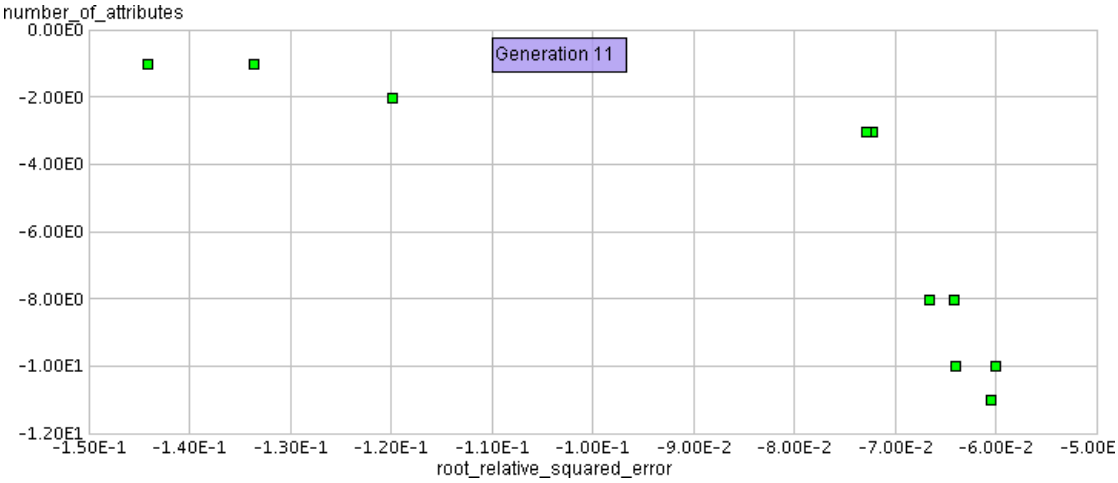



**Figure 7.13.:** The Pareto front after generation 11. Although not all dominated points are removed yet, the Pareto front can already be seen.

the average value of all 10 results. The second criterion is simply the number of features as it was proposed in the last section. As before, we again use the negative values as criteria in order to turn the problems into maximization problems. The error is shown on the x-axis and the number of features is shown on the y-axis.

Each point of the Pareto front then corresponds to a specific method tree with a certain performance and complexity. Figure 7.14 shows the method tree from this Pareto set which produces 8 features. Please note that each point in the Pareto set corresponds to a method tree for feature extraction as it was described in this chapter. The trees on the upper left side are pretty simple and produce only a single feature. Of course, they also do not provide good prediction performances. Walking to the lower right part, the method trees are growing and more features are constructed (up to 11 in this intermediate result) while the error decreases.

It can also be seen, that the multi-objective selection scheme again covers a wide range of sensible solutions. From a single feature (-1 on the y-axis) to the maximum number of features many possible feature numbers are part of the resulting Pareto set. The points in the resulting Pareto sets also have a property similar to the feature set rankings seen in the previous chapter. More complicated method trees are often build from subtrees defined by the less complex results.

Hence, beside the change in data structure for the search points, the basic interpretation of these Pareto fronts is equal to those for the simple multi-objective feature selection case (see Section 6.2.1) or for the similar multi-objective feature construction case (see Section 6.5.1).

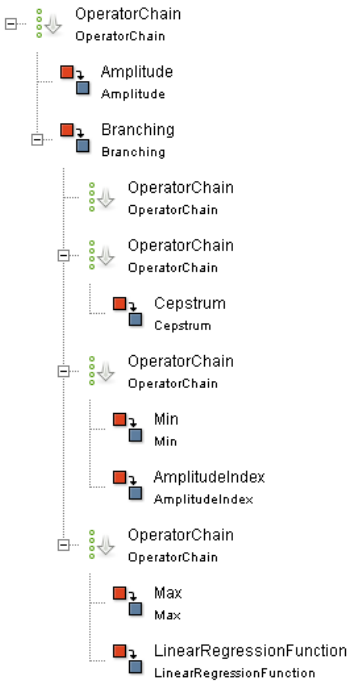



**Figure 7.14.:** A medium-sized feature extraction method tree from the Pareto front shown in Figure 7.13.

| Method | Root Relative Squared Error |
|---|---|
| Linear Regression | $0.035 \pm 0.013$ |
| GP Feature Extraction | **0.022 ± 0.005** |

**Table 7.5.:** Comparison of the genetic programming based method tree learning approach and a simple windowed linear regression approach for time series predictions. The results are obtained by a 10-fold back-testing validation. The bold font marks a significantly better result on a 1% confidence level for the feature extraction approach.

We finally compared a linear regression model learned on the features resulting from the best method tree with a linear regression model learned on the original windows. Table 7.5 shows the results. The bold font marks a significantly better result on a 1% confidence level for the feature extraction approach if all constructed features from the Pareto set are used. As it was expected, the best method tree delivered by the multi-objective feature extraction approach clearly outperforms the model without feature extraction.

CHAPTER 8

# Multi-Objective Unsupervised Feature Selection

The last chapters concentrated on the problem of supervised feature space transformations. We have seen that we can define a structural risk for feature space transformations similar to that known from statistical learning. The probably most simple instance for the model complexity is the number of features used in the resulting feature space. This even works for the quite complex task of supervised feature extraction. The definition of this structural risk for feature space transformations leads to a non-convex multi-objective optimization problem which we solved by means of evolutionary algorithms.

We will now change the underlying learning task and no longer deal with supervised learning problems. For unsupervised learning, the definition of a structural risk (Chapter 6) is not possible, neither for the learning problem nor the feature space transformations. The main reason for this of course is that no loss function can be defined due to the lack of a label. Nevertheless, we will show in the next two chapters that the problem of unsupervised feature space transformations again has a multi-objective nature. We will use known clustering criteria which try to measure the quality of clusters instead of the empirical risk. We also transfer the idea of feature set size to the unsupervised setting. This will lead to a surprising change which must be performed in order to define a valid multi-objective optimization problem for unsupervised feature space transformations.

Feature selection for unsupervised learning is a challenging new application in machine learning. In this chapter, we show how the rigorous definition of competing criteria leads to a novel and sound theoretical framework based on multi-objective optimization. This approach creates Pareto sets which share an interesting property: depending on the number of inherent patterns each front shows several kinks. These kinks allow an interpretable segmentation from which the user can select few prototypes which drastically reduces the effort of selecting a final solution. This turns unsupervised feature selection

into a case study for automatic segmentation of highly complex Pareto sets. Another important consequence of the necessary paradigm change discussed in this chapter is a method which segments the Pareto sets produced by our approach. Inspecting only prototypical points from these segments drastically reduces the amount of work for selecting a final solution. In the last section of this chapter, we will compare our methods against existing approaches on eight data sets.

## 8.1. Unsupervised Feature Selection

Chapter 2 discussed two different machine learning tasks, namely supervised and unsupervised learning. For supervised learning, a set of labeled data points must be given. The learning method should merely find a function which *predicts* the label for unseen data points. Supervised learning methods cannot be applied if no information is known beforehand. Unsupervised machine learning should rather *describe* the data set. Hence, the task is to automatically find the inherent, natural patterns of the data. Such natural patterns can express useful information for a decision maker. Typical application areas include customer segmentation, information retrieval, and image analysis [69, 133].

We have seen that the search for a proper supervised prediction function can usually be formulated as an optimization problem where the number of wrong predictions for the known data points should be minimized. A similar criterion for validity does not exist in the unsupervised setting. The optimization function of an unsupervised algorithm cannot rely on given patterns in order to decide if a found pattern is "correct" or "wrong". The validity of discovered patterns also depends on the background knowledge and intention of the user. It is therefore often desirable that unsupervised learning methods present more than one solution to the user.

The main problems for both supervised and unsupervised learning algorithms is to decide which dimensions of the data space should be taken into account. The prediction accuracy of a learned decision function can be dramatically increased if redundant or noisy features are omitted during learning. Although the problem is the same for both learning paradigms, the consequences might be rather different.

The last chapters have shown that supervised feature selection and extraction problems can be solved by minimizing the number of used features while prediction accuracy is preserved. The search for the best feature subset out of all $2^m$ possible subsets usually requires heuristics for larger dimensions $m$. Genetic algorithms have demonstrated their ability to solve this problem several times before [149, 196] and this thesis extended these approaches by explicitly defining the trade-offs between several conflicting criteria. This turns the minimization of the number of features and the maximization of the prediction accuracy into a multi-objective optimization problem since removing necessary features from the data set will decrease accuracy.

The same problem exists for unsupervised learning. The existence of noisy or redundant features can cover inherent data clusters and omitting those features might reveal the actual natural patterns. There are several approaches which try to directly identify promising feature subsets for clustering [158]. However, these approaches do not reflect the multi-objective character of the problem setting. Although no loss function $L(f(X), Y)$ exist and hence the definition of a regularized risk in the way we know will not work, we will see that the problem of unsupervised feature selection is still multi-objective. Therefore, state of the art feature selection approaches for unsupervised learning use multi-objective optimization. They transfer the idea of minimizing the number of features while the clustering optimization criterion should be preserved [81, 82, 132].

However, the definition of the optimization problem in this way is not appropriate. Under very weak assumptions we will show that the Pareto set will collapse into one singular point if the number of features should be minimized while the existing cluster evaluation measures should also be optimized. Even if the population does not collapse it tends to cover only a small fraction of the solution space.

Nevertheless, there *is* a trade-off between the number of features and cluster validity. However, the number of features must be *maximized* instead of minimized in order to achieve this competition. Although this might sound surprising at first, the change of the optimization direction has a natural origin in the aim of unsupervised learning. In order to describe the data set at hand, the amount of information which could be derived from the used feature set should be preserved during the feature selection process.

We will see in the experiments that the resulting Pareto sets are more beneficial for users since they provide a larger coverage of possible candidate solutions. The Pareto fronts produced by our selection approach can also be segmented into meaningful regions. This eases the selection of a final solution from the set of Pareto optimal points. We enable feature selection for density based clustering schemes as well. In contrast to combinatorial clustering algorithms like $k$-means, these clustering algorithms are able to find non-Gaussian clusters like rings or spirals. In the following, we propose an improved feature selection approach which is applicable to a wide variety of clustering algorithms and provides interpretable segmentations of the complex Pareto set.

## 8.2. Data Clustering

One of the most important approaches to unsupervised learning is data clustering. The aim of cluster analysis is to group data points into sets of similar data points. Given a data set $X$, which is an unlabeled set of individual data points $x_i \in X$ (observations). A cluster is a subset of data points $C_q \subseteq X$. In principle, clusters may overlap. However, most clustering algorithms are designed to produce partitions of data points, i. e. a set

of clusters $C_1 \dots C_k$ such that $C_i \cap C_j \neq \emptyset \Rightarrow C_i = C_j$ (clusters do not overlap) and $\bigcup_{q=1}^{k} C_q = X$ (each data point is covered by a cluster).

### 8.2.1. Combinatorial Clustering Algorithms

Cluster analysis aims at assigning items to clusters where elements within a cluster are more similar to each other than to the elements of other clusters. This notion can be expressed as an optimization problem by using a distance measure $d(x_i, x_j)$ on the set of data points:

$$\text{minimize} \quad \omega_d = \sum_{q=1}^{k} \sum_{x_i \in C_q} \sum_{x_j \in C_q} d(x_i, x_j).$$

It can be shown that by minimizing $\omega_d$ the mean pairwise difference between pairs of points in different clusters is maximized [63]. A very efficient approach for optimizing this function is $k$-means clustering. It is based on the squared Euclidean distance. In the following, we assume that the data points are represented by a set of $m$ real valued features, i. e. $x_i \in \mathbb{R}^m$, and that $x_{ip}$ is the value of the $p$-th feature for data point $x_i$. The Euclidean distance of two points $x_i$ and $x_j$ is

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{m} (x_{ip} - x_{jp})^2}.$$

It can be shown that optimizing $\omega_d$ with respect to the squared Euclidean distance is equivalent to optimizing the function

$$\omega = \sum_{q=1}^{k} \sum_{x_i \in C_q} \sum_{p=1}^{m} (x_{ip} - c_{qp})^2$$

where $c_{qp}$ is the $p$-th value of the centroid of cluster $C_q$. The centroid is the point with the smallest distance to all points in $C_q$. It can be calculated as

$$c_{qp} = \frac{\sum_{x_i \in C_q} x_{ip}}{|C_q|}.$$

The $k$-means algorithm uses this relationship by applying an alternating optimization procedure [62]. In each step, every data point is assigned to the cluster with the nearest centroid. Then, the centroids are recalculated for each cluster and data points are newly assigned to clusters based on the new centroids. This alternation stops if there is no further change in cluster assignment or after a maximal number of steps. The

centroids are initialized with random data points drawn from $X$. $k$-means does not guarantee to find an optimal solution and is sensitive to the choice of initial points. Therefore, a common strategy is to start $k$-means several times with different random initializations. For its simplicity and efficiency, $k$-means is one of the most popular clustering algorithms.

A natural choice for evaluating a set of clusters produced by $k$-means is $\omega$, i. e. the function it optimizes. This measure has several drawbacks. Most important, it is not normalized with respect to the feature values and to the number of clusters. With an increasing number of clusters, $\omega$ decreases monotonically. $\omega$ decreases as well for a decreasing number of features. For these reasons, it is not well suited as criterion for unsupervised feature selection problems.

Several other evaluation measures for $k$-means were proposed. Probably the most important is the Davies-Bouldin (DB) index [34]. It is calculated as

$$\omega_{DB} = \frac{1}{k}\sum_{q=1}^{k} \max_{q,r\neq q} \left\{ \frac{s_q + s_r}{d(c_q, c_r)} \right\}$$

where $s_q$ and $s_r$ are the average within cluster distances for cluster $C_q$ and $C_r$ respectively which are defined as

$$s_q = \frac{1}{|C_q|} \sum_{x_i \in C_q} d(x_i, c_q)$$

and

$$s_r = \frac{1}{|C_r|} \sum_{x_i \in C_r} d(x_i, c_r)$$

with the centroids $c_q$ and $c_r$.

$\omega_{DB}$ takes into account only the relative separation of the two clusters which are worst separated. This value is normalized as it is divided by the distance between the corresponding centroids. Therefore, it is less sensitive to the number of clusters. And it is also less sensitive to the number of features since the dimension of the space is also part of the centroid-based normalization. Focusing on the clusters that are worst separated allows for a very fine grained optimization, as the parts of the clustering that are well separated do not overshadow these important parts. Please note that $\omega_{DB}$ is only used in order to evaluate a cluster structure. There is no efficient algorithm which optimizes $\omega_{DB}$ directly.

### 8.2.2. Gaussian Mixtures

Gaussian mixture clustering assumes that the underlying data generating process consists of a mixture of different overlapping Gaussian distributions and that each distribution represents an individual cluster. Clustering is achieved by estimating the parameters

of the underlying distributions and assigning each data point to the distribution that most likely produced it. This notion can be formalized by

$$\omega_{GM} = \log \sum_{x_i \in X} \sum_{q=1}^{k} p_q g(x_i | \mu_q, \Sigma_q)$$

where $g$ is a multivariate Gaussian distribution representing cluster $C_q$ with mixture parameters $\mu_q$ and $\Sigma_q$. The probability $p_q$ is the apriori probability of a data point belonging to cluster $C_q$.

The expectation maximization (EM) approach employs an alternating optimization procedure similar to the one applied in $k$-means clustering [38]. First, each data point is assigned to the distribution that most likely generated it, then the parameters of all distributions (and thus clusters) are recalculated based on the data points in each cluster. This procedure is repeated until no further change occurs or a maximum number of steps is reached.

### 8.2.3. Graph-Based Clustering

Instead of assigning data points to clusters directly, we can first impose a graph $(X, E)$ on these points consisting of the nodes (data points) $X$ and a set $E$ of edges between them. Two data points $x_i$ and $x_j$ are connected if they are sufficiently similar to each other. For a maximum distance threshold $d_{max}$, $x_i$ and $x_j$ are connected if $d(x_i, x_j) \leq d_{max}$. The by far most common clustering criterion is to regard clusters as connected components in the distance graph. This approach is denoted as *single link clustering*. Finding such connected components in a graph is usually achieved by a graph search [30].

The value $d_{max}$ determines the coherence of the resulting clusters. If a fixed number of clusters $k$ is given, we can find a maximal value for $d_{max}$ such that the graph contains at most $k$ connected components and thus clusters. In this case, the resulting $d_{max}$ can be used as evaluation measure as it denotes the strength of the weakest link in the cluster structure. The weakest link is the point at which a given cluster would split up into two components if $d_{max}$ would be decreased. If we use $d_{max}$ as a cluster quality evaluation measure, we denote it similar to the other measurements as $\omega_{d-max}$.

Graph based clustering is very popular and can be combined with many advanced connectivity criteria as in *density based clustering* [45] or *support vector clustering* [13]. In contrast to $k$-means and Gaussian mixtures, these algorithms can detect clusters of any shape while $k$-means is limited to find spherical clusters boundaries. Figure 8.1 shows an example of a structure that cannot be clustered with neither EM nor $k$-means. Structures like these play an important role in applications as image recognition or astronomical data analysis [161].

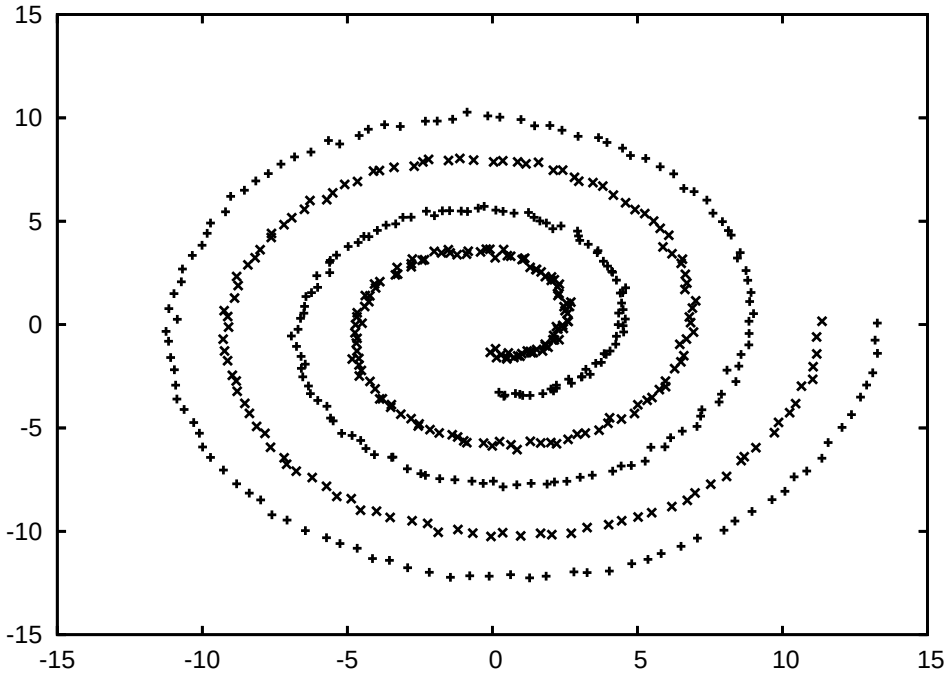



**Figure 8.1.:** The SPIRAL data set created for single link clustering.

## 8.3. Multi-Objective Feature Selection for Clustering

Multi-objective optimization is a natural choice for selecting appropriate feature subsets for clustering problems. The current state of the art is represented by the work described in [81, 82] and [132]. In the following, we will describe both approaches and show that they are both limited in several ways. These limitations are a result of the way the multi-objective optimization problem is posed.

Kim, Street, and Menczer introduce four performance criteria for $k$-means clustering[1] [81, 82]. The first one is a variant of $\omega$ that is normalized by the number of features $nf$:

$$\omega_{norm} = \frac{1}{nf}\omega.$$

A variant of between cluster distance is used as a second measure. However, this measure behaves essentially in the same way as normalized $\omega$ (minimizing within cluster distance is equivalent to maximizing between cluster distance). The third measure represents the maximum number of clusters $k$ which should be minimized. The last measure captures the number of features $nf$ that should be minimized as well.

In the following theorem we show that for a given number of clusters $k$ minimizing $\omega_{norm}$ and the number of features leads to exactly one Pareto optimal point. This optimal point always selects one single feature from the dataset, in particular the one that leads to a minimal loss with respect to the used clustering performance criterion.

**Theorem 8.1 (Pareto Front Collapse)** *Minimizing $\omega_{norm}$ and the number of features $nf$ leads to one single Pareto optimal point.*

*Proof.* Let $m$ be the total number of available features. For $\omega_{norm}$ we can denote the loss $l_p$ of an individual feature $p$ as

$$l_p = \sum_{q=1}^{k} \sum_{x_i \in C_q} (x_{ip} - c_{qp})^2.$$

In order to minimize the number of features selecting only one feature out of the $m$ possible features is optimal. We show that always

$$\omega_{norm} \geq \min_{1 \leq p \leq m} \{l_p\}.$$

[1] In the original work all criteria are normalized by a constant. This, however, has no influence on Pareto optimality.

That means that the performance can only decrease by adding any feature but the one that optimizes $l_p$. It can easily be seen that

$$\begin{aligned}\omega_{norm} =& \frac{1}{m}\sum_{p=1}^{m} l_p \\ \geq& \frac{1}{m}\sum_{p=1}^{m} \min_{1\leq p\leq m}\{l_p\} \\ =& \min_{1\leq p\leq m}\{l_p\}.\end{aligned}$$

□

Using $\omega_{norm}$ for the optimization is not a well suited approach for feature selection in clustering problems as it leads to trivial solutions. A similar proof can be given for normalized between cluster distance.

In [132] a normalized variant of $\omega_{DB}$ is proposed as alternative performance criterion to $\omega_{norm}$:

$$\omega_{DB,norm} = \frac{1}{nf}\omega_{DB}.$$

This approach is better suited, as $\omega_{DB}$ is normalized with respect to the feature space. However, this criterion is very sensitive. If the feature set contains for example a real valued feature that takes discrete values only, then choosing this one feature is again Pareto optimal. However, this one feature does almost certainly not represent the complete dataset in the descriptive sense mentioned in the introduction. In Section 8.5, we will see several examples for which the Pareto set collapses into a single trivial solution.

Furthermore, using normalized $\omega_{DB}$ does lead to a "build-in" competition between the number of features $nf$ and the cluster quality measure $\frac{1}{nf}\omega_{DB}$. Therefore, even clustering random data produces a Pareto front that exhibits an artificially introduced inversely proportional relationship (see Figure 8.2 (c)).

The major problem of both approaches is that they do not pose the problem correctly from the point of view of multi-objective optimization. In the next section we give an alternative problem formulation that solves the described difficulties.

## 8.4. Information Preserving Feature Selection

In the last sections, we discussed several quality measurements for different clustering schemes. In the following, we assume that all criteria should be maximized during feature selection. Criteria which should be minimized in the original problem definition are therefore multiplied by $-1$.

In contrast to the existing approaches discussed in Section 8.3 we do not minimize the number of features but *maximize* $nf$. This prevents the algorithm from selecting trivial solutions and leads to more complete Pareto sets of diverse natural clusterings. Although this change of the optimization direction might sound surprising from a supervised learning point of view, it is a direct consequence from the definition of the clustering evaluation criteria. The value of these criteria typically increases with the density of the data set which decreases with an increasing number of features. We have already seen that this relationship leads to the collapse of Pareto sets for certain criteria. Hence, the decision for maximizing the is well motivated if any competition between the criteria should be established. Otherwise, Pareto sets collapses will occur like it was proved in the previous section.

The fitness evaluation is done by performing a clustering scheme on the reduced feature sets. Depending on the used scheme we use $\omega_{DB}$ (equation 8.2.1) for $k$-means clustering and $\omega_{d-max}$ for single link clustering. Since there is a natural competition between maximizing the number of features $nf$ and the selected cluster criterion we do not need to apply an artificial normalization factor.

The unsupervised feature selection problem is inherently multi-objective and cannot be solved with single-objective evolutionary algorithms. In the clustering setting the user has no idea of criteria weights and, furthermore, there exist no simple decision about correct or wrong clusterings. Such a decision would totally depend on the amount of information the user can obtain from different clusterings. Therefore, we try to maintain as much information as possible and aim at finding all solutions which are optimal for arbitrary criteria weight vectors, i.e. the complete Pareto set. It was mentioned before that multi-objective evolutionary algorithms do not strongly depend on form and continuity of the Pareto-optimal set. We will see that for clustering with non-normalized optimization criteria the Pareto front is neither nicely shaped nor continuous.

We again use NSGA-II as the evolutionary selection technique for the unsupervised multi-objective feature selection task. Individuals are bit vectors of length $m$ indicating if a feature should be selected or not. The population size is set to $2m$, the maximal number of generations is 1000. A bit flip mutation is performed with probability $1/m$ and uniform crossover with probability 0.9.

### 8.4.1. Finding Interesting Points in the Pareto Front

The Pareto plots derived by the multi-objective optimization procedure described in the last section show a clear structure. This structure is not accidental but reflects structure in the underlying data. We can exploit this Pareto plot structure in order to discover patterns in the underlying data set. Moreover, exploiting this structure drastically reduces the effort of selecting a final solution from the Pareto set.

The basic idea is to find points at which the trade-off between the number of features and the cluster quality significantly changes. As this trade-off is represented by the slope of the Pareto plot at a given point, we want to find points where the change in slope is maximal. In the following, this notion is formalized. Since the following does not depend on the actual cluster quality evaluation measure, we will use the term $\omega$ if we refer to one of the cluster quality measures, for example $\omega_{DB}$ or $\omega_{d-max}$. We assume that all Pareto optimal points are sorted and that the $p$-th point is denoted by the pair $(\omega_p, nf_p)$. The value $\eta_p$ then represents the slope at point $p$ by

$$\eta_p = \arctan\left(\frac{\omega_{p+1} - \omega_p}{nf_{p+1} - nf_p}\right).$$

As we are interested in points at which the slope significantly changes we further calculate

$$\Delta\eta_p = \frac{\eta_p}{\eta_{p-1}}$$

with $\eta_0 = \pi/2$.

A value of $\Delta\eta_p$ greater than 1 indicates that adding an additional feature has a significant smaller negative influence on the cluster coherence than the proceeding features. A value smaller than 1 indicates that an additional feature has a stronger negative influence. The points between a strong increase and a strong decrease in slope often represent redundant features or very coherent sets of features (vertical parts). Adding an individual feature does not change the performance much. The features between a decrease and an increase represent areas with many noisy and incoherent features (horizontal parts). Adding such features has a direct negative influence on the cluster quality for each of the features that is added. For an example, please refer to Section 8.5.3 and Figure 8.6.

## 8.5. Experiments and Results

In this section, we will show results for both the approach proposed in this chapter and the normalized minimization approach discussed in Section 8.3 as it was proposed in [81, 82, 132]. We applied both algorithms to several synthetic and real world data sets.

### 8.5.1. The Data Sets

In order to measure the effect of the artificial normalization factor necessary for feature set minimization we applied the algorithms on a grid data set (GRID) and a random data set (RANDOM) containing only white noise. Another artificial data set (GM)

consisting of 32 Gaussian clusters with random standard deviations between 0.0 and 1.0 in five dimensions was created. This data set was enriched with ten additional single Gaussian noise features with standard deviation 0.5.

We also applied both algorithms on two clustering benchmark datasets, namely the IRIS data set [49] and the WPBC (Wisconsin Prognostic Breast Cancer) data set [192]. The latter is especially interesting because of many redundant features. This allows us to check how well both approaches are able to cope with redundancy. The IRIS data set is also used in two enriched variants containing additional noise features, either with nominal noise (IRIS-NN; a discrete-valued feature with randomly selected nominal values was added) or with Gaussian noise (IRIS-GN).

In order to evaluate feature selection for non-standard cluster boundaries we also generated an artificial data set consisting of two spirals and apply single link clustering to it. Such clusterings cannot be found be combinatorial clustering algorithms as $k$-means. Figure 8.1 shows the two non-noise dimensions of the created data set.

Table 8.1 summarizes the properties of all data sets. All experiments were performed with the freely available machine learning environment RapidMiner [125].

### 8.5.2. Interpretation of the Pareto Fronts

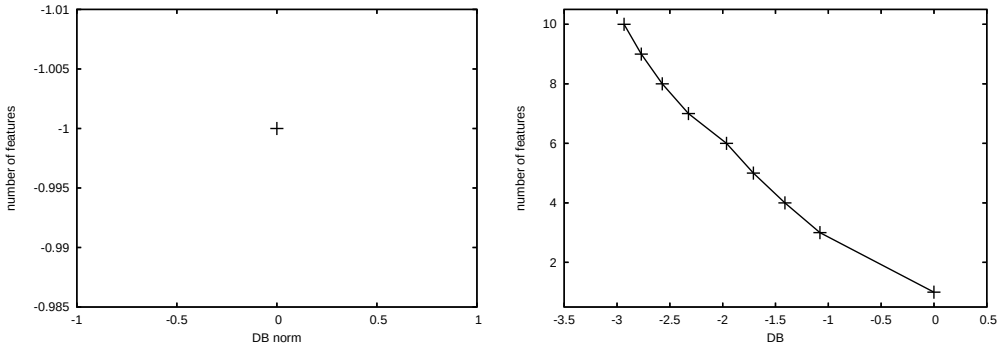
Figures 8.2, 8.3, and 8.4 shows all Pareto sets for the simultaneous optimization of the used cluster criterion and the feature set size. It should be noted that in most cases the population converges to the final front after less than 20 generations. Moreover, NSGA-II selection was again able to sustain the found solution until the end of optimization.

The basic interpretation of the resulting Pareto fronts is equal to that for the simple multi-objective feature selection case (see Section 6.2.1). We hence concentrate our discussion on the differences between the two approaches.

It can clearly be seen that in all cases the Pareto sets provided by our information preserving approach contain more points than the results of the normalized minimization approach. If there is only one feature with a relative small standard deviation, the Pareto set of the minimization approach will still collapse (GRID, IRIS-NN, and SPIRAL). Moreover, the artificial normalization factor $1/nf$ introduces a convex front although there is nothing to optimize at all. This effect can be seen for the random data set, where the minimization approach finds a convex Pareto front while the front provided by our approach is still linear. For both the normal IRIS data set and the IRIS-GN data set enriched with noise features the proposed approach finds the complete Pareto set while the minimization approach was only able to find a small number of feature subsets. Since it is not clear which clustering is "correct" beforehand, the user should be

| **Abbr.** | **Properties** | **n** | **m** | **Noise** | $\sigma_o$ | $\sigma_n$ | **k** | **Results** |
|---|---|---|---|---|---|---|---|---|
| GRID | equidistant values in all dimensions | 3125 | 5 | 0 | n.a. | n.a. | 0 | (a) and (b) |
| RANDOM | uniformly distributed values | 500 | 10 | 10 | n.a | $\infty$ | 0 | (c) and (d) |
| GM | Gaussian mixture with 32 clusters | 1000 | 15 | 10 | 0.5 | 0.5 | 32 | (e) and (f) |
| IRIS | Iris data set without noise features | 150 | 4 | 0 | 0.8 | n.a. | 3 | (g) and (h) |
| IRIS-NN | Iris data set with nominal noise | 150 | 5 | 1 | 0.8 | 0.01 | 3 | (i) and (j) |
| IRIS-GN | Iris data set with Gaussian noise | 150 | 14 | 10 | 0.8 | 0.8 | 3 | (k) and (l) |
| WPBC | WPBC data set without noise features | 198 | 34 | 0 | 33.2 | n.a. | ? | (m) and (n) |
| SPIRAL | Two spirals around the origin | 500 | 7 | 5 | 5.5 | 5.5 | 2 | (o) and (p) |

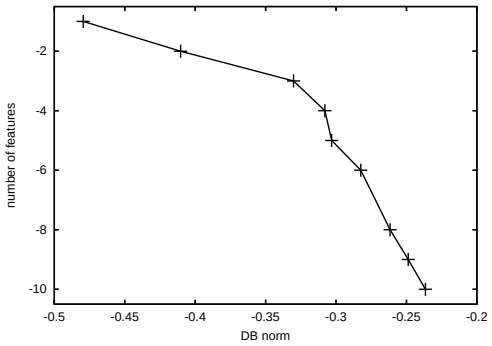
**Table 8.1.:** The used data sets. The first column summarizes the abbreviations used in the text, the second summarizes some properties of the data set. $n$ is the total number of examples, $m$ the total number of features. The column *noise* defines how many features of $m$ where explicitly added noise features. The next columns define the mean standard deviation of the original features ($\sigma_o$) and the noise features ($\sigma_n$). The column $k$ indicates the number of clusters if it is known. The last column indicates which Pareto sets were found for the data set with both approaches.



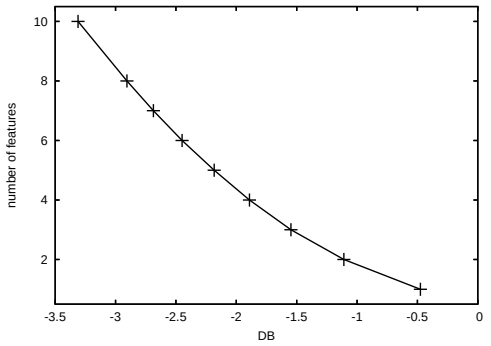

(a) GRID Min vs. $\omega_{DB,norm}$



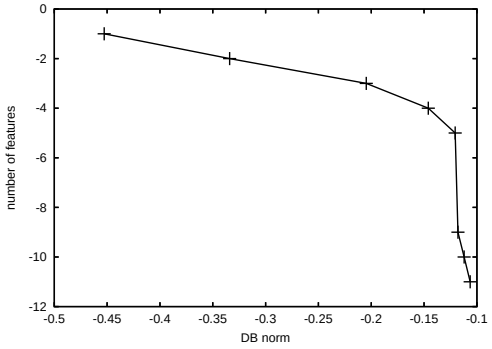

(b) GRID Max vs. $\omega_{DB}$



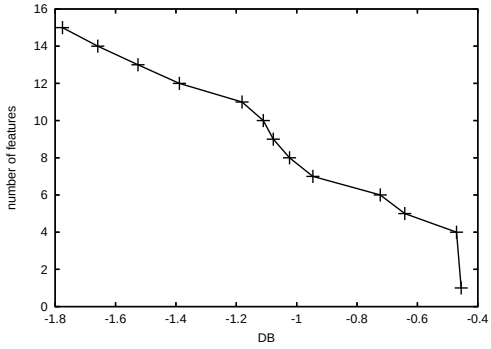

(c) RANDOM Min vs. $\omega_{DB,norm}$




(d) RANDOM Max vs. $\omega_{DB}$

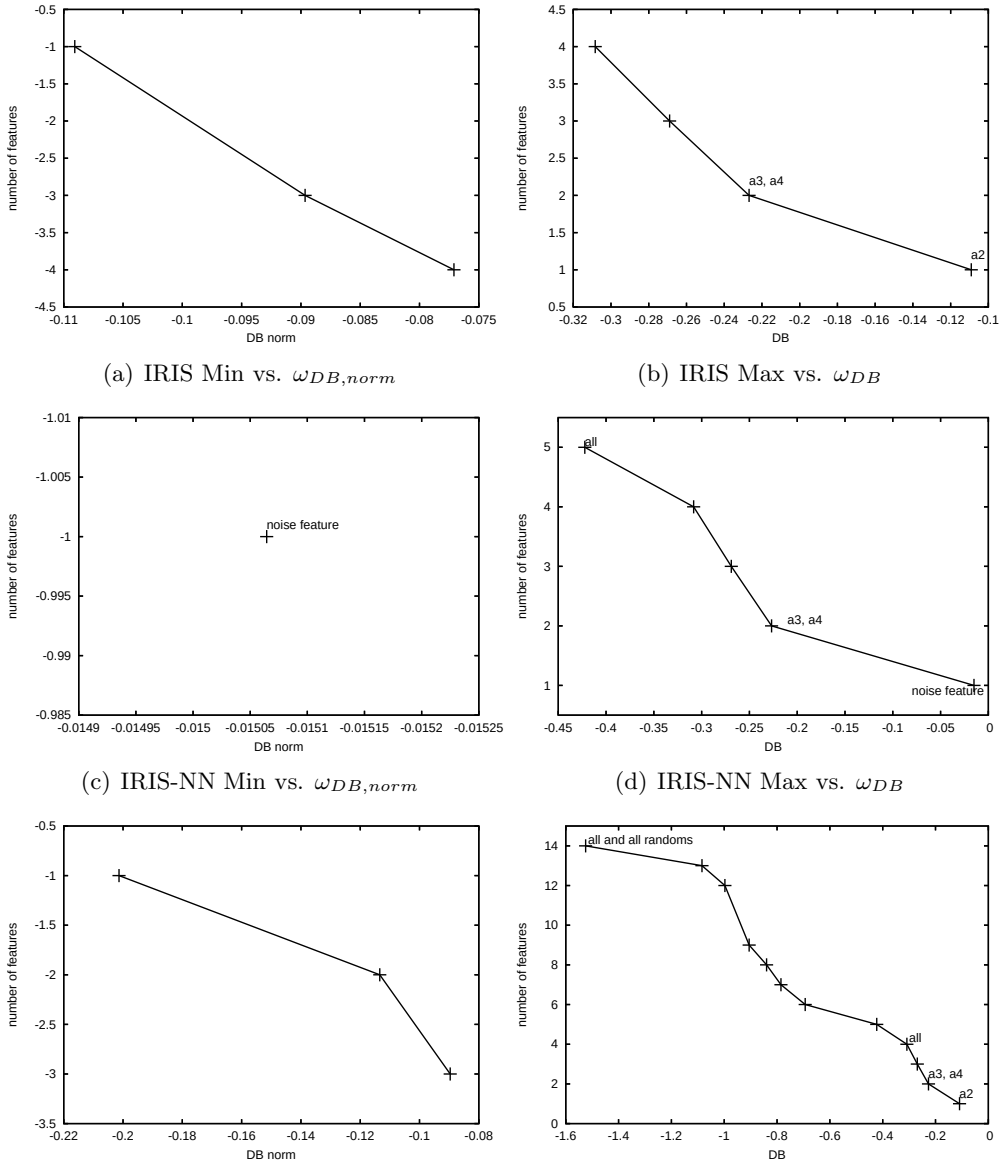



(e) GM Min vs. $\omega_{DB,norm}$




(f) GM Max vs. $\omega_{DB}$

**Figure 8.2.:** The Pareto fronts for all data sets. The left result for each dataset is achieved by the approach discussed in section 8.3 for a normalized value $\omega_{DB,norm}$ ($DB_{norm}$). It can clearly be seen that these results are not as complete and that kinks are covered by the artificial inversely proportional structure. The results on the right are achieved by our information preserving maximization approach. Part 1 of the results.




(a) IRIS Min vs. $\omega_{DB,norm}$

(b) IRIS Max vs. $\omega_{DB}$

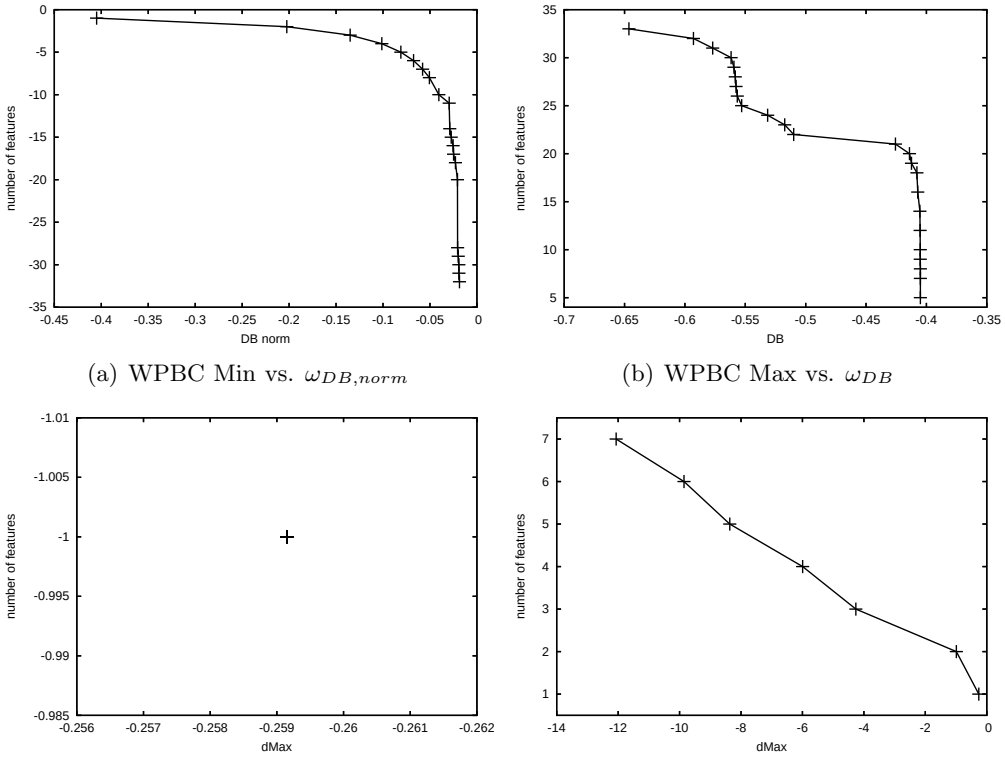
(c) IRIS-NN Min vs. $\omega_{DB,norm}$

(d) IRIS-NN Max vs. $\omega_{DB}$

(e) IRIS-GN Min vs. $\omega_{DB,norm}$

(f) IRIS-GN Max vs. $\omega_{DB}$

**Figure 8.3.:** The Pareto fronts for all data sets. The left result for each dataset is achieved by the approach discussed in section 8.3 for a normalized value $\omega_{DB,norm}$ ($DB_{norm}$). It can clearly be seen that these results are not as complete and that kinks are covered by the artificial inversely proportional structure. The results on the right are achieved by our information preserving maximization approach. Part 2 of the results.




(a) WPBC Min vs. $\omega_{DB,norm}$

(b) WPBC Max vs. $\omega_{DB}$

(c) SPIRAL Min vs. $d_{max}$

(d) SPIRAL Max vs. $d_{max}$

**Figure 8.4.:** The Pareto fronts for all data sets. The left result for each dataset is achieved by the approach discussed in section 8.3 for a normalized value $\omega_{DB,norm}$ ($DB_{norm}$). It can clearly be seen that these results are not as complete and that kinks are covered by the artificial inversely proportional structure. The results on the right are achieved by our information preserving maximization approach. Part 3 of the results.

able to select from the complete Pareto front. The same applies for the other real-world data set WPBC (Figure 8.4 shows the results for $k = 2$).

We did not focus on the simultaneous optimization of the number of clusters although this is of course possible. Since our approach does not differ from existing approaches in this respect we concentrated on the usability of information preservation. However, for the real world data set WPBC we simultaneously optimize $k$ in the range of $[2, 10]$ which leads to a three dimensional Pareto set (Figure 8.5). The Pareto plot shows the influence of the number of clusters on the kinks in the Pareto fronts. There are two regions with distinct deviations from the convex hull of the Pareto set. One small kink for $k = 8$ in the rear region and one bigger kink for $k = 2, 3, 4$ in the front region. These kinks were also totally covered by the inversely proportional structure introduced by the normalization approach and it would not be possible to detect a structure at all. Furthermore, redundant features can easily be determined in the almost vertical parts of the front.

### 8.5.3. Pareto Front Segmentation

Besides the fact that the Pareto sets are more complete, there is another advantage of the non-normalized maximization: kinks are not covered by any artificially introduced inversely proportional relationship structure. In contrast, these kinks can easily be discovered in the result of our approach. We applied the segmentation algorithm described in section 8.4.1 on the Pareto set delivered for WPBC and $k = 2$. The kinks between neighbors can clearly be seen if the number of features is maximized (Figure 8.4 (b)) instead of minimized (Figure 8.4 (a)) and the clustering criterion was not normalized. Selecting the five points with highest absolute deviances of $\Delta\eta_p$ to 1 leads to an interpretable segmentation of the result. Figure 8.6 shows the selected points and the segments.

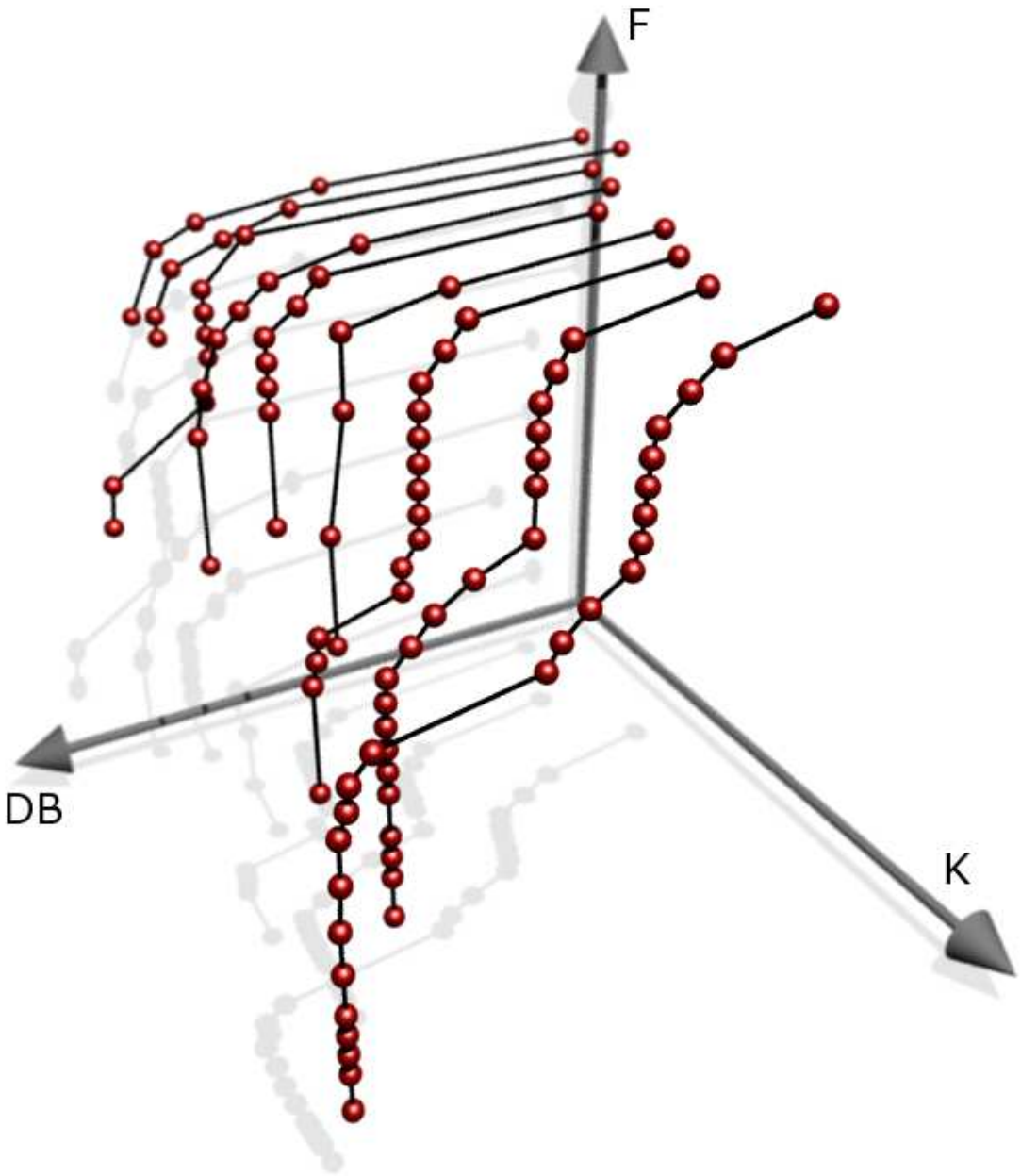



**Figure 8.5.:** We applied information preserving feature selection on the real-world data set WPBC. The number of features $nf$ ($F$ in the plot), the Davies Bouldin clustering criterion $\omega_{DB}$ ($DB$ in the plot), and the number of clusters $k$ ($K$ in the plot) should be simultaneously optimized. The result is a three dimensional Pareto set containing all necessary information allowing a decision about the best clustering. The kinks could be used to segment the Pareto set and ease the analysis of the front.

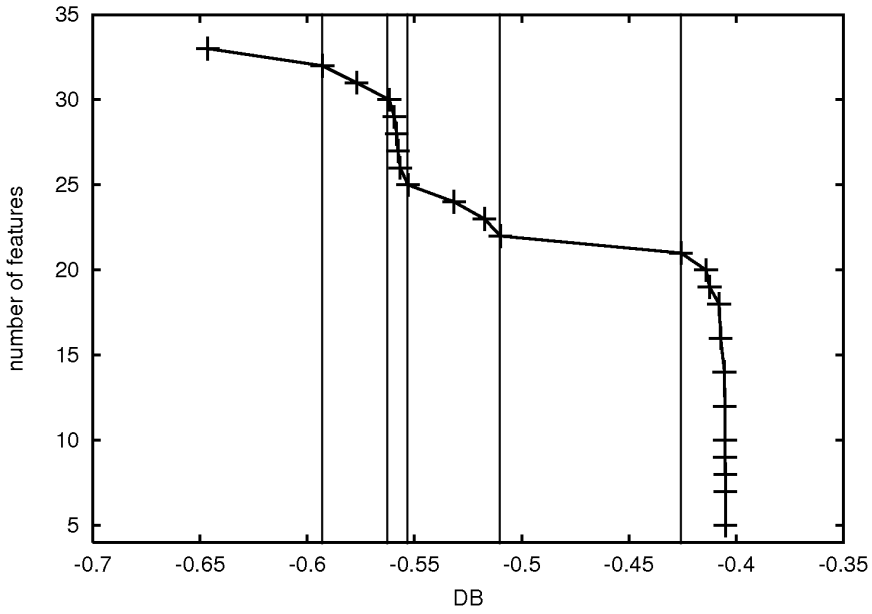



**Figure 8.6.:** The five points with the highest absolute deviation of $\Delta\eta_p$ to 1 are marked with perpendicular lines. This leads to an interpretable segmentation of the Pareto front which eases the process of selecting a final solution from the Pareto set.

CHAPTER 9

# Multi-Objective Feature Space Transformation for Clustering

The last chapter presented a novel multi-objective evolutionary framework for feature selection in unsupervised machine learning settings. Clustering is an ideal test case for evolutionary computation methods for several reasons. First, it is an inherently multi-objective problem. There is usually not one correct result as for supervised learning. Users rather explore the space of results interactively to gain insight into the natural patterns within the data set. Second, the approach proposed in this work yields Pareto sets that show significant inner structures. These structures are not accidental but reflects patterns in the underlying data. The last chapter also discussed a generic method for an automatic Pareto set segmentation and showed that the discovered segments can be interpreted with respect to unsupervised feature selection.

These benefits can only be achieved if the optimization problem is posed in a sound way. Although maximizing the number of features during feature selection might sound surprising at first, this paradigm change can be motivated by the aim of unsupervised learning: the search for descriptive, natural patterns. In particular, we have shown that existing approaches to multi-objective unsupervised feature selection based on minimization are not appropriate as they produce trivial or incomplete solution sets. Another contribution of this approach is its applicability to other clustering algorithms than EM or $k$-means. This is essential in applications in which clusters of any shape must be found, e. g. by means of density based clustering algorithms.

This chapter will show an extension of these basic ideas for the case of unsupervised feature construction. We will discuss a novel, generalized framework for feature space transformation in unsupervised knowledge discovery settings. Unsupervised feature space transformation is also inherently multi-objective. To facilitate data exploration, transformations should increase the quality of the result and should still preserve as much of

the original data set information as possible. We exemplify this relationship on the problem of data clustering. We have already seen that existing approaches to multi-objective unsupervised feature selection do not pose the optimization problem in an appropriate way. This was corrected in the previous chapter. However, using feature selection only is often not sufficient for real-world knowledge discovery tasks. We propose a new, generalized framework based on the idea of information preservation presented in the last chapter. This framework enables feature selection as well as a simple form of feature construction for unsupervised learning.

## 9.1. Motivation for Feature Space Transformations

Many knowledge discovery problems cannot be solved accurately by using the original feature space. This is due to several factors as noise, redundancy, sparsity or the fact that standard learning algorithms cannot represent complex relationships. Both supervised and unsupervised knowledge discovery therefore depend on methods that transform the feature space in an appropriate way.

We have discussed a solution for unsupervised feature selection in the previous. However, feature selection is only a limited form of feature space transformation. It can for example not solve the problems of sparsity and feature interaction. Methods for feature space reduction can be applied to reduce noise and sparsity before applying unsupervised learning, e. g. Kernel-PCA [164]. Selecting appropriate parameters for new data sets is non trivial. This is especially problematic as such methods lead to feature spaces that are hard to interprete and resulting patterns are even harder to analyze.

The main limitation of these approaches is, however, that they do not reflect that feature space transformation for unsupervised learning inherently is a multi-objective optimization problem. We have seen, that for the task of feature selection the number of features must be maximized instead of minimized in order to define a sound multi-objective unsupervised feature selection setting. This change in the optimization direction also means that the original space should not be completely changed - which perfectly corresponds to the descriptive goal of the unsupervised learning.

We now try to transfer the basic idea of information preservation to the task of feature construction. The general idea is to optimize the quality of the resulting patterns by partially transforming the original feature space, i.e. the original data should be modified as little as possible. This goal is clearly conflicting to the goal of optimizing a cluster evaluation measure. It will, for example, be very easy to create a constant feature yielding a global optimum for all clustering criteria known today.

## 9.2. Information Preserving Feature Aggregation

We have motivated that merely selecting features is often not sufficient. First, sparse data is a severe problem in many applications areas. For example in text clustering, generalizing terms by adding superordinate terms can significantly improve the quality of the result [67]. The same holds for association rule mining. Adding generalized features which combine individual items to classes, enables the algorithm to find patterns, which would not be valid in the original data space [171]. Second, many datasets contain features produced by similar underlying processes, e.g. time series data. Popular preprocessing approaches as moving average replace neighboring values by a generalized value exploiting the assumption that neighbors are similar.

### 9.2.1. Definition of Feature Aggregation

In the following, we present a general formalism for feature aggregation. The goal is to pairwise aggregate features as long as performance increases. On the other hand, the feature space is not allowed to change too drastically because otherwise the found clusters would hardly provide any insight into the actual structures in the data set.

The formalism proposed here is a straightforward generalization of the feature selection framework presented in the previous chapter and should fulfill several requirements. First, the constructed feature space should be easily interpretable in order to allow for a quick inspection of the results. Second, the optimization problem should be posed in a way that it can be solved efficiently. Third, as for selection, trivial solutions must be avoided. This directly leads to the fact that aggregated values should deviate as little as possible from both given feature values.

**Definition 9.1 (Feature Aggregation Function)** *Let $X$ denote the data set and $X_r$ a single feature. A* FEATURE AGGREGATION FUNCTION *is a function $f : X_r \times X_s \rightarrow X_t$ that maps two features to a new feature.*

Please note that the newly aggregated feature replaces the arguments. In the following, we state formal conditions that an aggregation function should fulfill in order to meet the requirement stated above. As point of departure, we use the concept of a *t-conorm*, that captures the notion of combining values by disjunction very naturally.

**Definition 9.2 (t-Conorm)** *A function is a* T-CONORM *if it fulfills the following constraints:*

1. *Boundary condition:* $f(x, 0) = x$
2. *Commutativity:* $f(x_r, x_s) = f(x_s, x_r)$

3. *Associativity:* $f(f(x_r, x_s), x_t) = f(x_r, f(x_s, x_t))$
4. *Monotonicity:* $x_r \geq x_s \Rightarrow f(x_r, x_t) \geq f(x_s, x_t)$

Associativity and commutativity ensure that the feature aggregation is order independent, thus that the order in which features are aggregated does not have an influence on the result. This considerably reduces the search space and leads to results that are easier to interprete, as the system produces sets of features instead of trees. The boundary condition ensures that the aggregation follows the notion of a disjunctive merging. Monotonicity preserves the ordinal information in the data.

### 9.2.2. Information Preserving Feature Aggregation

The formalization of the feature aggregation presented in the last section, however, is not sufficient to capture the notion of a minimal deviation. For example, it would still be possible that $f(x, x) \neq x$. Thus, even if both features have the same value, the resulting value would be different. This clearly violates the concept of merging two features and altering them minimally. We therefore add an additional constraint that excludes such functions:

**Definition 9.3 (Information Preserving t-Conorm)** *A function is an* INFORMATION PRESERVING T-CONORM *if it is a t-conorm and fulfills the following minimal deviation condition:*

$$\forall x, x' \in X : \neg\exists f'(x, x') : |f'(x, x') - x| + |f'(x, x') - x'| < |f(x, x') - x| + |f(x, x') - x'|. \tag{9.1}$$

This condition states that the aggregation function should always yield a merged value that has a minimal deviation from both original values. In the following, we show that from the t-conorm conditions and condition 9.1, two additional conditions can be derived analytically. The first one is idempotence:

**Definition 9.4 (Idempotence)** *A function fulfills* IDEMPOTENCE *iff* $f(x, x) = x$.

Aggregation functions definitely should fulfill idempotence since it could hardly be motivated why an aggregated value should differ from identical function arguments. This would not meet our notion of aggregation functions as it was motivated above. The next lemma states the formal connection between information preservation and idempotence:

**Lemma 9.1 (Information Preservation and Idempotence)** *An information preserving t-conorm* $f(x, x')$ *(fulfills condition 9.1) also fulfills idempotence.*

*Proof.* Trivial. □

### 9.2.3. Domain Preserving Feature Aggregation

We need another property for our aggregation functions which will ensure that the domain of the feature spaces will not change:

**Definition 9.5 (Domain Preservation)** *A function fulfills* DOMAIN PRESERVATION *iff* $min(x, x') \leq f(x, x') \leq max(x, x')$.

Thus the merged value must be in the domain spanned by the input values. This is again something we would expect from an aggregation function. The following theorem states that our notion of information preserving t-conorms also already captured this property:

**Theorem 9.1 (Information and Domain Preservation)** *An information preserving t-conorm $f(x, x')$ (fulfills condition 9.1) also fulfills domain preservation.*

*Proof.* We will prove by contradiction. For $x = x'$ the condition is trivially violated. We have to prove four cases and assume that $f(x, x') > max(x, x')$ and $x > x'$. Then:

$$\begin{aligned} |f(x, x') - x| + |f(x, x') - x'| &= (f(x, x') - x) + (f(x, x') - x') \\ &> (f(x, x') - x) + (x - x') \geq (x - x') \\ &= |max(x, x') - x| + |max(x, x') - x'| \end{aligned}$$

Thus condition 9.1 is violated. The other cases can be shown analogously. □

We have now stated the definition of feature aggregation functions and also the conditions for information preserving and domain preserving aggregation functions. Both conditions are very important to ensure only minimal deviations of the original feature space which is desired for unsupervised feature space transformations. It it now very interesting that the above conditions constraint the set of possible aggregation functions to exactly a single one, the maximum function:

**Theorem 9.2 (Maximum as Information Preserving t-Conorm)** *The maximum function is the only aggregation function fulfilling all conditions stated above.*

*Proof.* We will first show that for all t-conorms: $f(x, x')$, $max(x, x') \leq f(x, x')$. This is actually a consequence from the boundary condition, the commutativity condition, and the monotonicity condition in Definition 9.2. We prove by contradiction by assuming that there exists a t-conorm $f(x, x')$ with $f(x, x') < max(x, x')$. We will discuss the case where $x < x'$, the other cases can be analogously shown. If $x < x'$, it holds that $f(x, x') < x'$ in our contradiction setting. Let now $x > x_0 = 0$, then we can follow from the monotonicity constraint that $f(x, x') \geq f(x_0, x') = f(0, x') = x'$ which is a contradiction to $f(x, x') < x'$.

On the other hand, the domain preservation conditions requires that $f(x, x') \leq max(x, x')$, hence $f(x, x') = max(x, x')$. □

## 9.3. Criteria for Multi-Objective Unsupervised Feature Space Transformations

Given the aggregation function, we still need to extend the performance measure proposed in the previous chapter in order to capture feature aggregation as well. We have seen that for mere feature selection the number $nf$ of selected features is sufficient for measuring the degree of feature space preservation. One of the surprising results of this work is that this number should be maximized instead of minimized in the unsupervised setting. We want to extend the proposed framework in a way that feature selection is a special case of the more generic feature space transformation setting. We give two conditions which must be fulfilled by this generalized cost measure:

**Conditions 9.1 (Unsupervised Feature Space Transformation Measure)** *Let $nf_o$ be the number of selected original, i.e. non-aggregated, features. Let $nf_a$ be the number of aggregated features in the transformed feature set. For an unsupervised feature space transformation measure $nf$ the following must hold:*

1. *$nf = nf_o$ if the feature set does not contain any aggregated features.*
2. *Every aggregation must lead to a loss of $-a$ with $a > 0$.*

In the following we will assume $a = 1$. A very simple measure fulfilling these conditions is given by $nf = nf_o + nf_a$. If no features were aggregated $nf_a$ is 0 and $nf = nf_o$. Since all aggregation functions must replace the input features, aggregating two original base features reduces $nf_o$ by 2 and increases $nf_a$ by 1. Hence $nf$ is totally increased by 1. The same applies in the case of two already aggregated features or in the case of a merge of one base feature with an already aggregated features. Hence, every aggregation leads to the same loss of $-1$. Just as for the mere feature selection case the number $nf$ should be maximized.

Allowing the aggregation of features induce a representation change for the individuals of the evolutionary algorithm. Individuals are still represented by vectors $\vec{v}$ of length $m$. In contrast to the feature selection case, each coefficient of this vector is a number $v_i \in [-1, \max(v_1, \ldots, v_m)]$. This number $v_i$ represents the state of the $i$-th feature. $-1$ means that the feature is not selected at all. 0 means that the feature is used in its original form. Any number greater than 0 means that the feature should be aggregated with other features with the same number. This ensures that each feature is used at most once in the complete set. The mutation operator performs a uniformly distributed random change of each coefficient in the interval $[-1, max(v_1, \ldots, v_m) + 1]$. This mutation is

| Abbr. | Properties | n | m | k | Results |
|---|---|---|---|---|---|
| IRIS-M | Iris data set with divided features | 150 | 8 | 3 | (a) |
| KDDCUP-2004 | quantum physics data (KDD 2004) | 5000 | 78 | 2 | (b) |
| NEWSGROUPS | articles from three newsgroups | 3000 | 1052 | 16 | (c) |

**Table 9.1.:** The used data sets for unsupervised feature space transformation.

performed with probability $1/m$ for each coefficient. The other parameters are the same as in the special case of feature selection.

One important property of our approach is that the number of features is strictly monotonically decreasing. This is important for the efficiency of the proposed method, as increasing the number of features will decrease the runtime of the inner clustering algorithm. In contrast to other feature construction approaches the used vector representation also ensures that the amount of memory is restricted to the start individual size since the length $m$ of the individuals does not change during the feature aggregation process. Therefore, our approach can also be used for large scale unsupervised feature selection and aggregation and is feasible even for large data sets with many features.

## 9.4. Experiments and Results

We analyze the properties of our generalized feature space transformation on several synthetic and real-world data sets. The essential requirement is that the resulting Pareto sets are as broad as possible. The worst case is a Pareto front that collapses into a single point. All experiments were performed with the freely available machine learning environment RapidMiner [125].

We applied the new unsupervised feature aggregation algorithm on one synthetic and two real-world data sets. The properties of these data sets are summarized in Table 9.1. For the data set IRIS-M we divided the values of the four Iris features into two parts A and B resulting in a total of eight features. For each of the original feature values, we randomly select one of the corresponding two new features as target and use the original value as the value for this randomly chosen target feature. The value of the non-chosen feature is set to a random value between 0 and the original value. This way the complete original information can only be reconstructed by aggregating the correct features by means of the maximum function. The KDDCUP-2004 data set consists of a sample of 5000 examples drawn from the quantum physics data of the KDD cup 2004. For the data set NEWSGROUPS we combined three newsgroups of the well known 20-newsgroups data set which results in 3000 examples.

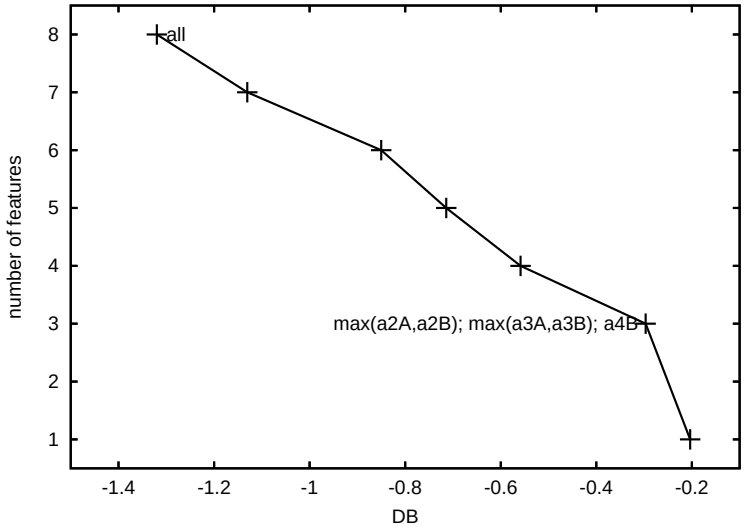



**Figure 9.1.:** The Pareto fronts delivered by the unsupervised multi-objective feature aggregation experiments on the IRIS-M data set. The Pareto sets still cover the complete range of possible solutions from 1 until 8 features for the IRIS-M data set. Additionally, features were only aggregated if this combination was necessary.

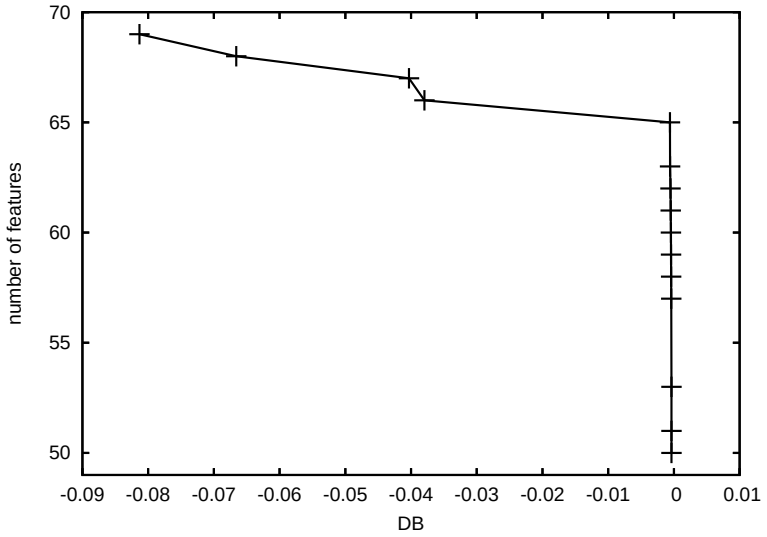
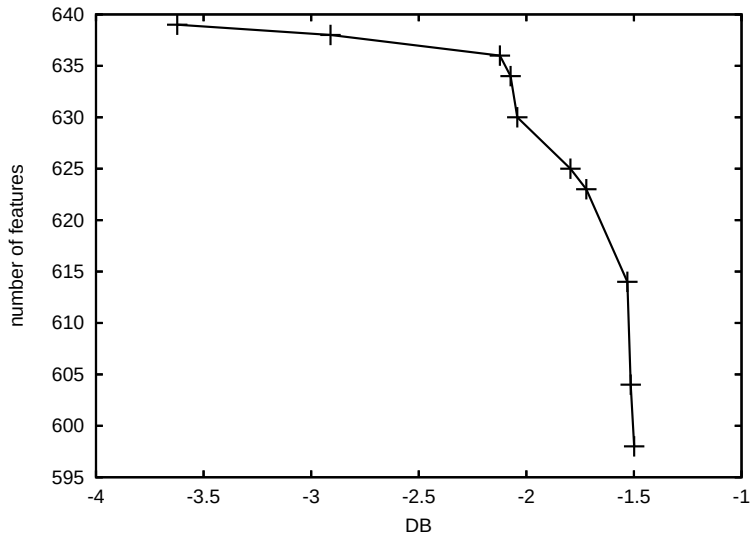



**Figure 9.2.:** The Pareto fronts delivered by the unsupervised multi-objective feature aggregation experiments on the KDDCUP-2004 data set.




(a) NEWSGROUPS

**Figure 9.3.:** The Pareto fronts delivered by the unsupervised multi-objective feature aggregation experiments on the NEWSGROUPS data set.

### 9.4.1. Interpretation of the Pareto Fronts

Figure 9.1, 9.2, and 9.3 shows the results for unsupervised feature space transformation. The basic interpretation of the resulting Pareto fronts is equal to that for the simple multi-objective feature selection case (see Section 6.2.1). The x-axis shows the negative cluster criterion which is to be maximized and the y-axis shows the number of features which also should be maximized. Hence, the complete front should again be moved to the upper right part of the multi-objective optimization scheme.

For the IRIS-M data set, the complete range of solutions is covered by the resulting Pareto set and the necessary features $a2$ and $a3$ were reconstructed by aggregation. For KDDCUP-2004 a clear kink can be seen indicating redundant features which are aggregated in the lower part of the vertical line. For both real-world data sets again a broad range of the feature space is covered by the result which supports our claim that our approach yields robust and useful solutions.

In the following paragraphs, we conclude our results for unsupervised feature space transformations. We presented a novel multi-objective framework for feature space transformation in clustering settings which plays an important role in a wide variety of applications ranging from pattern recognition to customer relationship management and web search. Clustering is an inherently multi-objective problem. There is usually

not one correct result as for supervised learning. Users rather explore the space of results interactively to gain insight into the natural patterns within the data set.

The results in this chapter are based on the results in the previous one on multi-objective feature selection for clustering. Since merely selecting features is not sufficient in many settings, we extended the proposed approach to the task of feature construction. Especially the problems of sparse data and feature interactions cannot be properly solved by feature selection only.

The last two chapters presented an approach that is based on the idea of information preservation. As much of the original data space should be preserved as possible, while the validity of the resulting clusters is optimized. We show that this approach yields complete and useful Pareto sets. The original feature set and clusterings were found in all cases. These Pareto sets moreover show a strong inner structure which can be used to explore the set of solutions even more efficiently by inspecting only these interesting points.

Furthermore, we extended our approach to allow for a limited form of feature construction as well. Aggregation is used to derive new features, that generalize over two or more features in the original data set. A set of basic conditions limits feature construction to the $t$-conorm maximum which summarizes two features with minimal alteration. We show that even for feature aggregation our approach leads to robust Pareto sets.

Also, our approach is very generic. As it essentially adopts a wrapper approach, it can be combined with a large variety of problems and algorithms. It is therefore easy to adapt to new problems and application domains.

And this is the major result of the second part of this thesis: posing the optimization problem of unsupervised feature selection and construction in a sound way by maximizing the number of features solves the unsupervised feature space transformation problem for a very first time.

CHAPTER 10

# Feature Set Transfers

It is widely known that feature selection and construction is essential for solving many complex learning problems. We have seen in the previous chapters how multi-objective optimization can help to improve the results and increase the insight into the underlying processes.

Unfortunately, the construction of features usually implies searching a very large space of possibilities and is often computationally demanding. We now propose a case based approach to feature construction which should overcome the issues connected to computation time. Learning tasks are stored together with a corresponding set of constructed features in a case base and can be retrieved to speed up feature construction for new tasks. The essential part of our method is a new representation model for learning tasks and a corresponding distance measure. Learning tasks are compared using relevance weights on a common set of base features only. Therefore, the case base can be built and queried very efficiently. In this respect, our approach is unique and enables us to apply case based feature construction not only on a large scale, but also in distributed learning scenarios in which communication costs play an important role. We derive a distance measure for heterogeneous learning tasks by stating a set of necessary conditions. Although the conditions are quite basic, they constrain the set of applicable methods to a surprisingly small number.

## 10.1. Motivation for Feature Set Transfers

Many inductive learning problems cannot be solved accurately by using the original feature space. This is due to the fact that standard learning algorithms cannot represent complex relationships as induced for example by trigonometric functions or ratios. For example, if only base features $X_1$ and $X_2$ are given but the target function depends

highly on $X_c = \sin(X_1 \cdot X_2)$, the construction of the feature $X_c$ would ease learning and most often is necessary to enable any reasonable predictions at all [17, 33, 89].

Unfortunately, feature construction is a computationally very demanding task often requiring to search a very large space of possibilities [115, 114, 194]. In this chapter, we consider a scenario in which several learners face the problem of feature construction on different learning problems. The idea is to transfer constructed features between similar learning tasks to speed up the generation in such cases in which a successful feature has already been generated by another feature constructor. Such approaches are usually referred to as Meta Learning [187].

Meta Learning was applied to a large variety of problems and on different conceptual levels. The importance of the representation bias, which is closely related to feature construction, was recognized since the early days of Meta Learning research [10, 11]. The key to many Meta Learning methods is the definition of similarity between different learning tasks [12, 178]. Here, we propose a Meta Learning scheme that compares two learning tasks using only relevance weights assigned to a set of base features by the individual learners.

This is motivated by a set of constraints found in many distributed Meta Learning scenarios. Firstly, the retrieval of similar learning tasks and relevant features usually has to be very efficient, especially for interactive applications. This also means that methods should enable a best effort strategy, such that the user can stop the retrieval process at any point and get the current best result. Secondly, the system should scale well with an increasing number of learning tasks. Also, it has to deal with a large variety of heterogeneous learning tasks, as we cannot make any strict assumptions on the individual problems. Finally, since many Meta Learning systems are distributed, communication cost should be as low as possible. As a consequence, methods that are based on exchanging examples or many feature vectors are not applicable.

Considering the given constraints, we develop and analyze a method for feature construction based on Meta Learning that estimates the distance of two learning tasks using only base feature weights. All learners weigh a common set of base features according to their relevance to the given learning task. Some of the learners, those that have already performed feature construction, send these relevance weights and the constructed features to a case base. From this case base, learners can retrieve possibly relevant features by sending their own base feature weights. The central challenge is to find a measure that compares two learning tasks using only base feature weights.

## 10.2. Basic Concepts

Before we state the conditions which must be met by any method comparing learning tasks using feature weights only, we first introduce some basic definitions. Let $\tau$ be the set of all learning tasks, a single task is denoted by $T_i$. Let $X_i$ be a vector of numerical random variables for task $T_i$ and $Y_i$ another random variable, the target variable. These obey a fixed but unknown probability distribution $P(X_i, Y_i)$. The components of $X_i$ are called *features* $X_{ik}$. Please note that we have to add the index $i$ in order to identify the instance spaces $X_i$ and also the features $X_{ik}$ which was not necessary in previous chapters and thus omitted. We need this additional index variable since we now have to distinguish between different learning tasks $T_i$ represented by possibly different feature sets $X_i$.

The objective of every *learning task* $T_i$ is to find a function $f_i(X_i)$ which predicts the value of $Y_i$. We assume that each set of features $X_i$ is partitioned in a set of *base features* $X_B$ which are common for all learning tasks $T_i \in \tau$ and a set of *constructed features* $X_i \setminus X_B$.

We now introduce a very simple model of feature relevance and interaction. The feature $X_{ik}$ is assumed to be irrelevant for a learning task $T_i$ if it does not improve the classification accuracy:

**Definition 10.1 (Irrelevant Features)** *A feature $X_{ik}$ is called* IRRELEVANT *for a learning task $T_i$ iff $X_{ik}$ is not correlated to the target feature $Y_i$, i. e. if $P(Y_i|X_{ik}) = P(Y_i)$.*

The set of all irrelevant features for a learning task $T_i$ is denoted by $\mathcal{I}_i$.

Two features $X_{ik}$ and $X_{il}$ are alternative for a learning task $T_i$, denoted by $X_{ik} \sim X_{il}$, if they can be replaced by each other without affecting the classification accuracy. For linear learning schemes, this leads to the linear correlation of two features:

**Definition 10.2 (Alternative Features)** *Two features $X_{ik}$ and $X_{il}$ are called* ALTERNATIVE *for a learning task $T_i$ (written as $X_{ik} \sim X_{il}$) iff $X_{il} = a + b \cdot X_{ik}$ with $b > 0$.*

In the following, we will denote those features which are either irrelevant or alternative to one of the base features (and hence also irrelevant - at least they do not need to be constructed) with the symbol $\mathcal{A}$. Please note that the definition above is a very limited definition of alternative features. However, we will show that most weighting algorithms are already ruled out by conditions based on this simple definition.

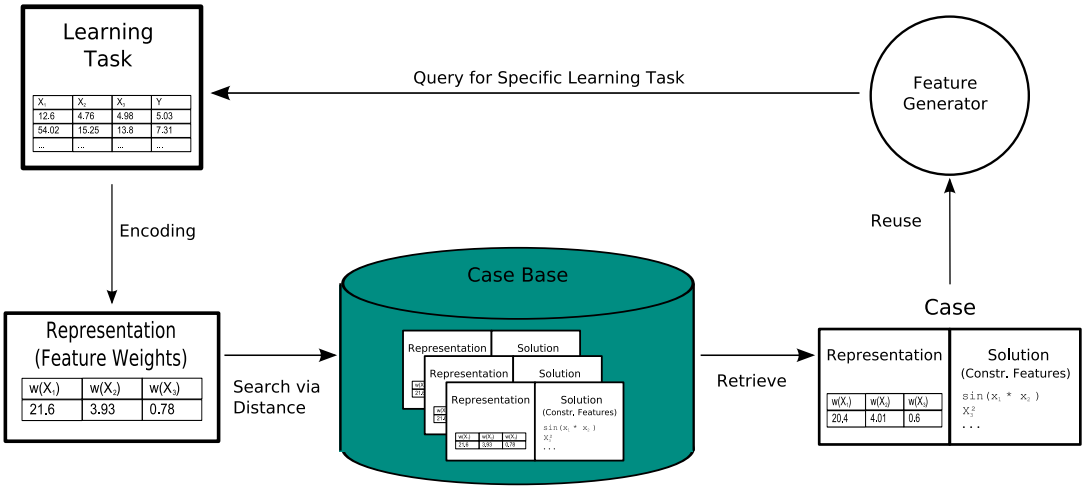



**Figure 10.1.:** Overview of the case-based feature construction process. Source: [138].

## 10.3. Comparing Learning Tasks Efficiently

The objective of our work is to speed up feature construction and improve prediction accuracy by building a case base containing pairs of learning tasks and corresponding sets of constructed features. We assume that a learning task $T_i$ is completely represented by a feature weight vector $w_i$. The vector $w_i$ is calculated from the base features $X_B$ only. This representation of learning tasks is motivated by the idea that a given learning scheme approximate similarly constructed features by a set of base features in a similar way, e. g. if the constructed feature $\sin(X_{ik} \cdot X_{il})$ is highly relevant the features $X_{ik}$ and $X_{il}$ are relevant as well.

Our approach works as follows: for a given learning task $T_i$ we first calculate the relevance of all base features $X_B$. We then use a distance function $d(T_i, T_j)$ to find the $k$ most similar learning tasks. Finally, we create a set of constructed features as union of the constructed features associated with these tasks.

This set is then evaluated on the learning task $T_i$. If the performance gain is sufficiently high (above a given fixed threshold) we store task $T_i$ in the case base as additional case. Otherwise, the constructed features are only used as initialization for a classical feature construction that is performed locally. If this leads to a sufficiently high increase in performance, the task $T_i$ is also stored to the case base along with the locally generated features. See Figure 10.1 for an overview.

While feature weighting and feature construction are well studied tasks, the core of our algorithm is the calculation of $d$ using only the relevance values $w$ of the base features $X_B$. In a first step, we define a set of conditions which must be met by feature weighting schemes. In a second step, a set of conditions for learning task distances is defined which makes use of the weighting conditions.

**Weighting Conditions 1 (Weighting Function Axioms)** *Let $w$ be a* WEIGHTING FUNCTION $w : X_B \to \mathbb{R}$. *Then the following must hold:*

*(W1)* $w(X_{ik}) = 0$ *if* $X_{ik} \in X_B$ *is irrelevant*

*(W2)* $F_i \subseteq X_B$ *is a set of alternative features. Then*

$$\forall S \subset F_i, S \neq \emptyset : \sum_{X_{ik} \in S} w(X_{ik}) = \sum_{X_{ik} \in F_i} w(X_{ik}) = \hat{w}$$

*(W3)* $w(X_{ik}) = w(X_{il})$ *if* $X_{ik} \sim X_{il}$

*(W4) Let* $\mathcal{A}$ *be a set of features where*

$$\forall X_{ik} \in \mathcal{A} : (X_{ik} \in \mathcal{I}_i \vee \exists X_{il} \in X_B : X_{ik} \sim X_{il}) .$$

*Then*

$$\forall X_{il} \in X_B : \nexists X_{ik} \in \mathcal{A} : X_{il} \sim X_{ik} \wedge w'(X_{il}) = w(X_{il})$$

*where $w'$ is a weighting function for $X'_B = X_B \cup \mathcal{A}$.*

These conditions state that irrelevant features have weight 0 and that the sum of weights of alternative features must be constant independently of the actual number of alternative features used. Together with the last condition this guarantees that a set of alternative features is not more important than a single feature of this set. Obviously, this is a desired property of a weighting function used for the comparison of learning tasks. In the following we assume that for a modified space of base features $X'_B$ the function $w'$ denotes the weighting function for $X'_B$ according to the definition in (W4).

Additionally, we can define a set of conditions which must be met by distance measures for learning tasks which are based on feature weights only:

**Distance Conditions 1 (Distance Function Axioms)** *A* DISTANCE MEASURE $d$ *for learning tasks is a mapping $d : \tau \times \tau \to \mathbb{R}^+$ which should fulfill at least the following conditions:*

*(D1)* $d(T_1, T_2) = 0 \Leftrightarrow T_1 = T_2$

*(D2)* $d(T_1, T_2) = d(T_2, T_1)$

*(D3)* $d(T_1, T_3) \leq d(T_1, T_2) + d(T_2, T_3)$

*(D4)* $d(T_1, T_2) = d(T_1', T_2')$ *if* $X_B' = X_B \cup \mathcal{I}$ *and* $\mathcal{I} \subseteq \mathcal{I}_1 \cap \mathcal{I}_2$

*(D5)* $d(T_1, T_2) = d(T_1', T_2')$ *if* $X_B' = X_B \cup \mathcal{A}$ *and* $\forall X_k \in \mathcal{A} : \exists X_l \in X_B : X_k \sim X_l$

(D1)–(D3) represent the conditions for a metric. These conditions are required for efficient case retrieval and indexing, using e. g. M-Trees [26]. (D4) states that irrelevant features should not have an influence on the distance. Finally, (D5) states that adding alternative features should not have an influence on distance.

## 10.4. Negative Results

In this section we will show that many feature weighting approaches do not fulfill the conditions (W1)–(W4). Furthermore, one of the most popular distance measures, the Euclidian distance, cannot be used as a learning task distance measure as it was introduced above.

**Lemma 10.1 (Feature Selection)** *Any feature selection method does not fulfill the conditions (W1)–(W4).*

*Proof.* For a feature selection method, weights are always binary, i. e. $w(X_{ik}) \in \{0, 1\}$. We assume a learning task $T_i$ with no alternative features and $X_B' = X_B \cup \{X_{ik}\}$ with $\exists X_{il} \in X_B : X_{il} \sim X_{ik}$, then either $w'(X_{il}) = w'(X_{ik}) = w(X_{il}) = 1$, leading to a contradiction with (W2), or $w'(X_{il}) \neq w'(X_{ik})$ leading to a contradiction with (W3). □

**Lemma 10.2 (Greedy Feature Weighting)** *Any (greedy) feature weighting method for which* $w(X_{ik})$ *is calculated independently of* $X_B \setminus X_{ik}$ *does not fulfill the conditions (W1)–(W4).*

*Proof.* We assume a learning task $T_i$ with no alternative features and $X_B' = X_B \cup \{X_{ik}\}$ with $\exists X_{il} \in X_B : X_{il} \sim X_{ik}$. If $w$ is independent of $X_B \setminus X_{ik}$ adding $X_{ik}$ would not change the weight $w'(X_{il})$ in the new feature space $X_B'$. From (W3) follows that $w'(X_{ik}) = w'(X_{il}) = w(X_{il})$ which is a violation of (W2). □

Lemma 10.2 essentially covers all feature weighting methods that treat features independently such as Information Gain [147] or Relief [84].

The next theorem states that the Euclidian distance cannot be used as a distance measure based on feature weights.

**Theorem 10.1 (Euclidean Distance)** *Euclidean distance does not fulfill the conditions (D1)–(D5).*

*Proof.* We give a counterexample. We assume that a weighting function $w$ is given which fulfills the conditions (W1)–(W4). Further assume that learning tasks $T_i, T_j$ are given with no alternative features. We add an alternative feature $X_{ik}$ to $X_B$ and get $X'_B = X_B \cup \{X_{ik}\}$ with $\exists X_{il} \in X_B : X_{il} \sim X_{ik}$. We infer from conditions (W2) and (W3) that

$$w'(X_{ik}) = w'(X_{il}) = \frac{w(X_{il})}{2} \quad \text{and} \quad w'(X_{jk}) = w'(X_{jl}) = \frac{w(X_{jl})}{2}$$

and from condition (W4) that

$$\forall p \neq k : w'(X_{ip}) = w(X_{ip}) \quad \text{and} \quad \forall p \neq k : w'(X_{jp}) = w(X_{jp}).$$

In this case, the following holds for the Euclidian distance

$$\begin{aligned} d(T'_i, T'_j) &= \sqrt{S + 2\left(w'(X_{ik}) - w'(X_{jk})\right)^2} = \sqrt{S + 2\left(\frac{w(X_{ik})}{2} - \frac{w(X_{jk})}{2}\right)^2} \\ &= \sqrt{S + \frac{1}{2}\left(w(X_{ik}) - w(X_{jk})\right)^2} \neq \sqrt{S + \left(w(X_{ik}) - w(X_{jk})\right)^2} = d(T_i, T_j) \end{aligned}$$

with

$$S = \sum_{p=1, p\neq k}^{|X_B|} \left(w'(X_{ip}) - w'(X_{jp})\right)^2 = \sum_{p=1, p\neq k}^{|X_B|} \left(w(X_{ip}) - w(X_{jp})\right)^2. \qquad \square$$

## 10.5. Positive Results

We have shown that many common feature weighting algorithms and distance measures cannot be used as learning task distance in our scenario. In this section we will prove that a combination of feature weights delivered by a linear Support Vector Machine with the Manhattan distance obeys the proposed conditions.

We have seen that SVMs aim to minimize the regularized risk $R_{reg}(w, b)$ of a learned function $f$ which is the weighted sum of the empirical risk $R_{emp}(w, b)$ and a complexity term $||w||^2$:

$$R_{reg}[f] = R_{emp}(w, b) + \lambda ||w||^2.$$

The model parameters $w$ and $b$ correspond to the function class parameters $\gamma$ known from the introduction. The result is a linear decision function $y = \text{sgn}(\langle w, x \rangle + b)$ with a minimal length of $w$. The vector $w$ is the normal vector of an optimal hyperplane with a maximal margin to both classes. One of the strengths of SVMs is the use of kernel functions to extend the feature space and allow linear decision boundaries after efficient non-linear transformations of the input. Since our goal is the construction of (non-linear)

features during preprocessing we can just use the most simple kernel function which is the dot product. In this case, the components of the vector $w$ can be interpreted as weights for all features.

**Theorem 10.2 (SVM Weighting)** *The feature weight calculation of SVMs with linear kernel function meets the conditions (W1)–(W4).*

*Proof.* Since these conditions can be proved for a single learning task $T_i$ we write $X_k$ and $w_k$ as a shortcut for $X_{ik}$ and $w(X_{ik})$.

*(W1)* We assume that the SVM finds an optimal hyperplane. The algorithm tries to minimize both the length of $w$ and the empirical error. This naturally corresponds to a maximum margin hyperplane where the weights of irrelevant features are 0 if enough data points are given because otherwise the length $||w||^2$ of the normal vector $w$ would not be minimal.

*(W2)* SVMs find the optimal hyperplane by minimizing the weight vector $w$. Using the optimal classification hyperplane with weight vector $w$ can be written as $y = \operatorname{sgn}(w_1 x_1 + \ldots + w_i x_i + \ldots + w_m x_m + b)$. We will show that this vector cannot be changed by adding the same feature more than one time. We assume that all alternative features can be transformed into identical features by normalizing the data. Adding $k-1$ alternative features will result in

$$y = \operatorname{sgn}\left( \ldots + \underbrace{\left(w_i^1 + \ldots + w_i^k\right)}_{\text{alternative features}} x_i + \ldots + b \right).$$

However, the optimal hyperplane will remain the same and does not depend on the number of alternative attributes. This means that the other values $w_j$ will not be changed. This leads to

$$w_i = \sum_{l=1}^{k} w_i^l$$

which proofs condition (W2).

*(W3)* The SVM optimization minimizes the length of the weight vector $w$. This can be written as

$$w_1^2 + \ldots + w_i^2 + \ldots + w_m^2 \stackrel{!}{=} \min.$$

We replace $w_i$ using condition (W2):

$$w_1^2 + \ldots + \left(\hat{w} - \sum_{j \neq i} w_j\right)^2 + \ldots + w_m^2 \stackrel{!}{=} \min.$$

In order to find the minimum we have to partially differentiate the last equation for all weights $w_k$:

$$\begin{aligned} & \frac{\partial}{\partial w_k}\left(\ldots+\left(\hat{w}-\sum_{j\neq i} w_j\right)^2+w_k^2+\ldots\right)=0 \\ \Leftrightarrow \quad & 2w_k-2\left(\hat{w}-\sum_{j\neq i} w_j\right)=0 \\ \Leftrightarrow \quad & w_k+\sum_{j\neq i} w_j=\hat{w}. \end{aligned}$$

The sum on the left side contains another $w_k$ for each derivation. This leads to a system of linear equations of the form

$$\begin{gathered} \vdots \\ \ldots+1\cdot w_{i-1}+0\cdot w_i+1\cdot w_{i+1}+\ldots+1\cdot w_{k-1}+2\cdot w_k+1\cdot w_{k+1}+\ldots=\hat{w} \\ \vdots \end{gathered}$$

The coefficient 0 always stays at the same $i$-th position, the coefficient 2 moved across all other positions *but* the $i$-th position. Solving this system of equations leads to $w_p = w_q$ (condition (W3)) for all pairs $p$ and $q$.

*(W4)* We again assume that a SVM finds an optimal hyperplane given enough data points. Since condition (W1) holds adding an irrelevant feature would not change the hyperplane and thus the weighting vector $w$ for the base features will remain. The proofs of conditions (W2) and (W3) state that the optimal hyperplane is not affected by alternative features as well. □

In order to calculate the distance of learning tasks based only on a set of base feature weights we still need a distance measure that met the conditions (D1)–(D5).

**Theorem 10.3 (Manhattan Distance)** *Manhattan distance does fulfill the conditions (D1)–(D5).*

*Proof.* The conditions (D1)–(D3) are fulfilled due to basic properties of the Manhattan distance. Therefore, we only give proofs for conditions (D4) and (D5).

*(D4)* We follow from the definition of the Manhattan distance that

$$d(T_i', T_j') = \sum_{X_{ip}, X_{jp} \in X_B} |w_i'(X_{ip}) - w_j'(X_{jp})| + \underbrace{\sum_{X_{iq}, X_{jq} \in \mathcal{I}} |w_i'(X_{iq}) - w_j'(X_{jq})|}_{0}$$

$$= d(T_i, T_j)$$

from (W4).

*(D5)* We show the case for adding $p$ features with $\forall X_{ik} : X_{ik} \sim X_{il}$ for a fixed $X_{il} \in X_B$:

$$d(T_i', T_i') = \sum_{p=1, p \neq k}^{|X_B|} |w_i'(X_{ip}) - w_j'(X_{jp})| + (p+1) \cdot |w_i'(X_{ik}) - w_j'(X_{jk})|$$

$$= \sum_{p=1, p \neq k}^{|X_B|} |w_i(X_{ip}) - w_j(X_{jp})| + |w_i(X_{ik}) - w_j(X_{jk})| = d(T_i, T_j)$$

from (W4) and (W2). □

Therefore, we conclude that SVM feature weights in combination with Manhattan distance fulfill the necessary constraints for an efficient learning task distance measure based on feature weights.

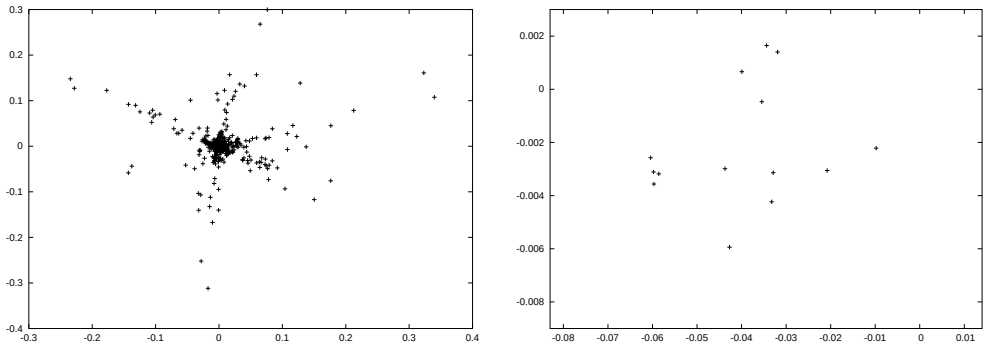
## 10.6. Experiments and Results

All experiments were performed with the machine learning environment RapidMiner [125].

### 10.6.1. Synthetical Data

In this section, we describe two experiments which were performed on synthetical regression problems. All cases have an important property: linear regression schemes are not able to predict the target function without applying some feature construction. Two case bases containing maximal 1000 cases was generated. For each case the following was done:

**Example set generation:** 300 examples with five base features were generated (plus four alternatives for the second data set). The target variable obeys a randomly created regression function containing the building blocks $+$, $*$, sin and exp. The maximal depth of the target function was 3, the probability for leafs which does




(a) Synthetical base weights (b) Real world base weights

**Figure 10.2.:** The base feature weights of the synthetical test cases and for the real world cases after a dimensionality reduction on two dimensions.

not contain another function was 0.3. These parameters could for example lead to functions like $X_5 \cdot e^{X_2} + \sin(X_3)$, the features $X_1$ and $X_4$ were irrelevant for this learning task example.

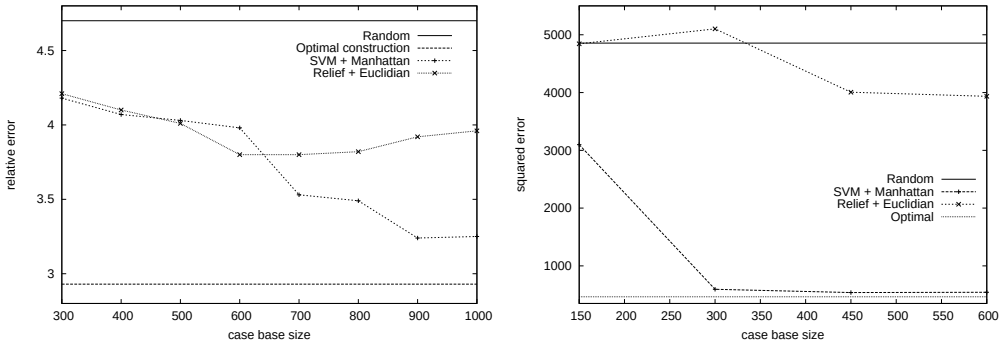
**Weight calculation:** The weights of the base features were calculated. We tried both linear SVM weights and weights calculated by Relief [84].

**Feature construction:** An evolutionary approach for feature construction was used to generate new features from the base features [115]. We used the following parameters: 200 generations, population size 10, and crossover probability 0.5.

**Adding the case:** the weights and the constructed features were added to the case base. Please note that the case base does not contain the data itself, information about the target variable, or the learned hypothesis.

The first learning task does not contain any alternative features. The second data set contains five alternatives for each feature to demonstrate the influence of the alternatives. Figure 10.2 (a) shows the base feature weights of all cases of the first learning task after a dimensionality reduction on two dimensions. The five main "axes" (the rays originating from the center of the data set) represent the five base features, the outermost points on each axis derive from the cases $\exp(\exp(\exp(X_i)))$. Cases between two axes contain functional parts depending on both base features.

After creation of the case base, 200 test cases were created in the same manner but without feature construction. We only performed feature construction on the test set to determine the optimal relative error which can be achieved in the optimal feature space. This minimal error was 2.93% and builds the bottom baseline for our comparison experiments. The upper bound for the error is the average error for a regression model on the base features only without any feature construction. In this case, the benchmark




(a) Result for non-alternative data set

(b) Result for alternative data set

**Figure 10.3.:** The results of case based feature construction. The averaged relative error of all 200 test cases is plotted against the number of cases used as case base. The combination SVM weights plus Manhattan distance clearly outperforms the combination Relief plus Euclidian distance. This applies especially for data sets containing alternative features.

error is 4.70% and this error should of course be reduced by means of feature construction and feature transfers.

The case based feature construction was performed with the constructed features of the 10 cases with minimal distance. Figure 10.3 shows the results. The averaged relative errors of all 200 test cases was plotted against the number of cases used for the case base. The plots show two curves for the combination of SVM feature weighting with Manhattan distance and Relief weighting with Euclidian distance. For the first learning task without alternative features, both approaches reduced the error beneath the benchmark error of 4.70% for all examined case base sizes. As expected the combination SVM plus Manhattan distance outperforms the combination of Relief plus Euclidian distance, at least for sufficiently large case bases. This combination almost reaches the minimal error 2.93% achieved by evolutionary feature construction. For data sets with several alternative features, the combination Relief and Euclidian distance does not significantly improve the performance.

Our approach fulfills the constraints presented above and can speed up the process of feature construction considerably. The average time needed for the automatic construction of an optimal feature set without using the case base was 76.8 seconds. Using the feature construction induced by the 20 most similar cases reduces the time needed for learning to 0.8 seconds.

| Method | Accuracy |
|---|---|
| Base features only | 64.7 |
| Random | 68.4 |
| Euclidean | 73.37 |
| Manhattan | 73.37 |

**Table 10.1.:** The achieved accuracy using base features only and feature recommendations (based on Random, Manhattan, and Euclidian distances of base feature weights).

### 10.6.2. Real World Data

We performed additional experiments using real world data based on the distributed music retrieval and structuring application described in Section 10.1. Our experiments are based on 39 taxonomies created by a group of students. The underlying audio data consists of about 2000 audio clips from the Garageband site[1]. We used 24 randomly chosen taxonomies for training and the remaining 15 for testing. The hierarchical classification problems are split into a set of flat, binary classification problems, by using each inner node with two or more children as a split point. If there are more than two sub nodes, two of them are selected at random. Also, all selected sub nodes have to contain more than 30 examples, as very small example sets would add a considerable amount of noise.

Random recommendation of features was compared with recommendations based on Manhattan and Euclidian distance. Ten base features were used to compare the cases. Figure 10.2 (b) shows the base feature weights of all cases after a singular value decomposition. The accuracy was estimated by 10-fold cross validation. Table 10.1 shows the results for the different approaches.

As can be seen, case based feature construction significantly improves the accuracy of the learners. However, there is no difference between Manhattan and Euclidian distance. This is due to the fact, that an optimized set of base features was used, that by construction contains only minimal redundancy. However, alternative features were the reason to prefer Manhattan over Euclidian distance which might be important in cases where alternative features exist.

## 10.7. Exploiting the Similarity of Constructed Features

In the previous sections, we discussed an approach to feature construction that is based on Meta Learning. Learning tasks are stored together with a corresponding set of con-

[1] http://www.garageband.com

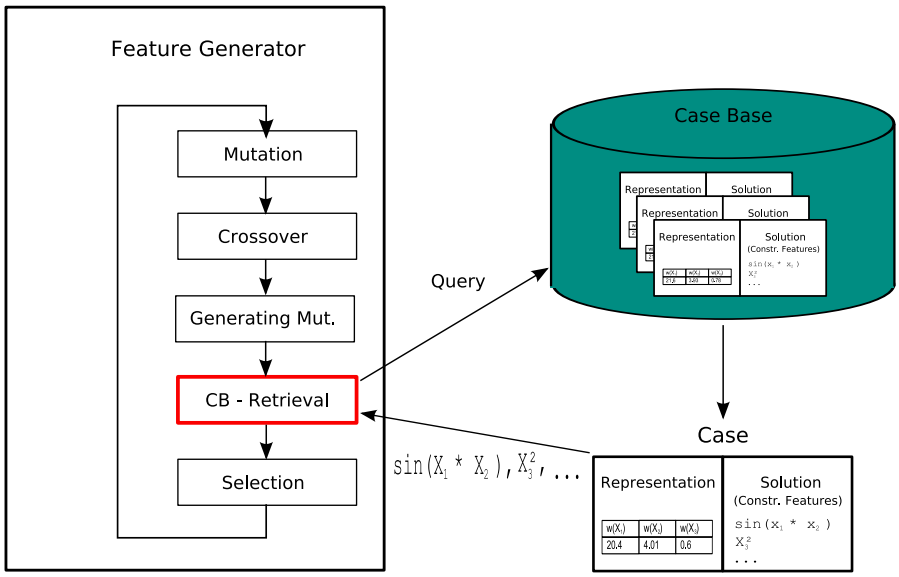



**Figure 10.4.:** We improve the usual evolutionary based feature construction algorithms like those described in Chapter 6 and 7 by adding the discussed case base feature retrieval as additional mutation. Source: [138].

structed features in a case base. This case base is then used to constrain and guide the feature construction for new tasks.

Now we want to also take into account the information about the already constructed features. This is for example necessary, if the case base should be queried multiple times. This new query method consists essentially of the new data task representation model from the previous section and a corresponding two step distance measure. Learning tasks are first compared using relevance weights on a common set of base features only. Such a case base can be built and queried very efficiently. In a second step, sampling is used to compare the additionally constructed features to the base features.

This two-phase approach is unique as it enables us to apply case based feature construction not only on a large scale, but also in distributed learning scenarios in which communication cost plays an important role and as a part of generation based randomized search heuristics like evolutionary algorithms. The idea is simply to add an additional mutation performing the case base retrieval to the search process (see Figure 10.4). Using the two step process, the accuracy of recommendations can be increased while not loosing the benefits of efficiency. The theoretical results are also confirmed by experiments on both synthetical data and data obtained from a distributed learning scenario on audio data.

```
Given: learning tasks t_i and t_j

total = 0;
for all constructed features X_ik \ X_B do {
  dist = Infinity;
  for all constructed features X_jl \ X_B do {
    dist = min(dist, calc_distance(X_ik, X_jl));
  }
  total = total + dist;
}
return total;
```

**Figure 10.5.:** The main routine to calculate the distance between two different feature sets including constructed attributes.

### 10.7.1. Similarity of Constructed Features

In the last section, we discussed a distance measure on a set of common base features. The calculations can be performed in linear time of both the number of training cases and the number of features. Furthermore, since the measure is a metric, efficient indexing of cases is possible using data structures like M-Trees.

However, if the base features are not sufficient to approximate the target function, the base weights alone might be a poor indicator for learning task distance. In the following, we introduce two extensions of the case based feature construction approach. First, we incorporate the similarity of constructed features into the distance measure presented above. This follows the idea that for similar learning tasks similar features must be constructed. Without loss of generality, the constructed features can be modeled as function trees. In the case of value series data the constructed features can be given as method trees (see Chapter 7). In order to take the similarity of constructed features into account, the distance between the feature trees must be calculated. It has been shown that calculating the difference of two unordered labeled trees is an NP-complete problem [200]. Approximations for this problem using syntactical heuristics are still not feasible for large feature trees and case bases [85]. Additionally, the data distribution is not considered by syntactical approaches. For example, $\sin(x)$ is very similar to $x$ for small absolute values of $x$ but of course not for $x \to \infty$. Therefore, we are using a probabilistic sampling approach considering the ranges of interest instead of syntactical heuristics. The main routine just compares each constructed feature of the first learning task with all constructed features of the second learning task. The minimum distances for all features are summed up and returned (Figure 10.5).

We employ a sampling approach for the routine `calc_distance` which ensures that the feature construction distance is calculated in some range of interest, i.e. with respect

```
Given: - base feature std. dev. σ_i and σ_j
       - features X_ik and X_jl

// calculate ranges
for (p = 1 until |X_B|) do {
  σ_p = min(σ_ip,σ_jp);
}
// generate small artificial data set
artificial_data = empty;
for (i = 1 until m) do {
  generate base feature data using σ;
  construct X_ik and X_jl on generated data;
}
// calculate correlation and return as distance
r = correlation(X_ik,X_jl) on artificial_data;
return 1 - (r * r);
```

**Figure 10.6.:** The routine `calc_distance` which calculates the distance between two constructed features using a range sensitive sampling and the squared correlation on a small artificially generated data set with size $m$.

to the data distributions in the base dimensions. Since the data does not need to be accessible, the basic idea is to generate $m$ artificial data points in the base space $X_B$ and construct both features $X_{ik}$ and $X_{jl}$ for this small artificial data set. The distance can then be calculated with help of the correlation coefficient or the absolute deviation of both constructed features. In order to ensure that the $m$ artificial points are generated in interesting ranges of all dimensions, the data ranges of all features must be submitted to the case base in addition to the base feature weights. If the data is mean standardized only the standard deviation must be transfered. Figure 10.6 shows the sampling based algorithm `calc_distance` using the squared correlation as feature distance for constructed features.

Using a probabilistic sampling approach allows an estimation how the distance calculation varies with different possible samples. This way we are able to say how close or far from the actual distance our estimation is likely to be [134]. Since the construction of features can be done in $O(1)$ (at least in the non-series case), the runtime of this sampling approach is quadratic in the number of constructed features and linear in the number $m$ of sample points and cases, i.e. $O(|X_i \setminus X_B| \cdot |X_j \setminus X_B| \cdot m)$ for each case. Hence, this sampling based distance measure does not only consider the feature ranges but is also feasible for large numbers of constructed attributes and cases.

The second extension to the case based feature construction approach introduced above is to allow learners to query the case base repeatedly. We start using base features only

and add further features in each iteration until a given stopping criterion is fulfilled. This can be a desired performance or a maximal amount of time. In each step, a learner can construct features on its own, e.g. with help of an evolutionary feature construction approach. This can also be seen the other way round: for an evolutionary approach the retrieval of similar cases and construction of new features is just another mutation of the input data.

### 10.7.2. Decreasing Runtime using a 2-Phase Distance Calculation

We have seen that the runtime of the discussed learning task similarity using constructed features is more efficient than other approaches calculating the syntactical tree distance of constructed features. However, a quadratic runtime might be too slow for real world applications using huge case bases. In this section, we combine both the simple base weight distance using SVM weights with Manhattan distance and the sampling based construction distance. In a first phase, we employ only the base feature weight distance to find a candidate set of the $k$ most promising cases. In a second phase, we use a weighted combination of both, the weight distance and the construction sampling distance introduced in the last section. This leads to a number of $k'$ cases from this candidate set whose constructed features are also constructed for the case at hand. Of course these steps can be embedded in the iterative learning process described above. The parameter $k$ allows an adaption of the trade-off between runtime and accuracy depending on the application.

## 10.8. Experiments and Results

Again, all experiments were performed with the machine learning environment RapidMiner [125]. We use data sets similar to those described in Section 10.6.1. A base containing maximal 1500 cases was generated. For each case the following was done:

**Example set generation:** We employ the same data generation processes as in Section 10.6.1 but generated 500 instead of 300 examples.

**Weight calculation:** The weights of the base features were calculated with a linear SVM.

**Feature construction:** An evolutionary approach for feature construction was used to generate new features from the base features [155] with the following parameters: 400 generations, population size 10, and crossover probability 0.5.

**Adding the case:** the standard deviations of the base features, the base weights, and a syntactical description of the constructed features were added to the case base.

| Size | No. | Random | Euclidean | Manhattan | 2-Phase |
|---|---|---|---|---|---|
| 50 | $\infty$ | $\infty$ | 0.228 | 0.213 | 0.145 |
| 100 | $\infty$ | $\infty$ | 0.201 | 0.196 | 0.106 |
| 250 | $\infty$ | 8.720 | 0.125 | 0.125 | 0.102 |
| 500 | $\infty$ | $\infty$ | 0.126 | 0.125 | 0.105 |
| 1000 | $\infty$ | $\infty$ | 0.120 | 0.119 | 0.106 |
| 1500 | $\infty$ | 4.010 | 0.121 | 0.116 | 0.106 |

**Table 10.2.:** The averaged relative errors for the different approaches. The symbol $\infty$ indicates that no result was produced in a reasonable amount of time.

Please note that the case base does not contain the data itself, information about the target variable, or the learned hypothesis.

After creation of the case base the 241 test cases were created in the same manner but without feature construction. It was not possible to reliably determine the averaged relative error for these test cases without using feature construction since most SVM runs does not converge. The relative error is defined as the absolute error divided by the minimum of the prediction and the true value. The case based feature construction was performed with the constructed features of the most similar case only. If this case was randomly selected from the case base without using a distance measure, an averaged relative error of 4.010 and 8.720 respectively could only be achieved for 1500 and 250 cases. For the other case base sizes it was also not possible to calculate an error in a reasonable amount of time. All error estimations were done with a 10-fold cross validation. Table 10.2 summarizes the results and Figure 10.7 plots the averaged relative errors of all 241 test cases against the number of used training cases. The plot shows the errors for the combination of SVM feature weighting and Manhattan distance and the new 2-phase construction approach with Manhattan distance on the base feature weights in the first phase (20 cases) and a construction sampling weight of 50. Both approaches dramatically reduced the error compared to the randomly selected cases and the 2-phase distance outperforms the base weight only distance. However, further experiments have shown that increasing the number $k'$ of similar cases reduce the performance gain of the 2-phase approach.

Our approach fulfills the constraints presented above and can speed up the process of feature construction considerably. The average time needed for the automatic construction of an optimal feature set without using the case base was 216.9 seconds. Using the feature construction induced by the most similar case reduces the time needed for learning to 2.18 seconds for the Manhattan distance and 3.67 seconds for the 2-phase approach. Using only the sampling construction distance also provides a very competitive error of 0.109 but needs a runtime of 58.73 seconds for each case.

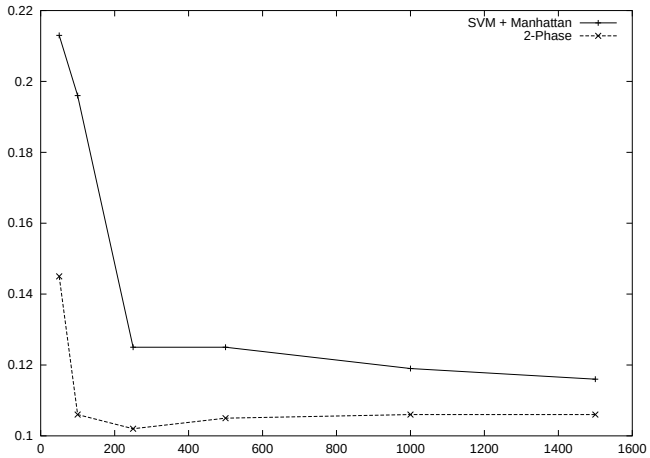



**Figure 10.7.:** The results of case based feature construction. The averaged relative error of all 241 test cases is plotted against the number of cases used as case base. The 2-phase approach clearly outperforms the SVM plus Manhattan distance.

The master thesis of Michael Nöthe provides additional insights into the performance gain which can be achieved by using a case base feature retrieval as part of an evolutionary feature construction algorithm [138]. In this work, the author extended the supervised feature construction approach presented in Chapter 6 (called YAGGA2) and added an additional case base feature retrieval mutation to YAGGA2. The resulting algorithm, called YAGGA3, queries the case base for the first time in the 5th generation and then in each 10th generation anew by using the discussed 2-phase retrieval approach. Figure 10.8 shows the performance of both algorithms depending on the current generation. It can clearly be seen that YAGGA3 converges much faster and receives huge performance gains in each lookup generation.

Another set of experiments was performed by Michael Nöthe on 10 different data sets which show that the new YAGGA3 approach is clearly better on most data sets. In the first set of experiments (Figure 10.9), the total runtime of both approach was limited to 100 seconds. The average results of 10 runs is reported for both algorithms. It can clearly be seen that the new YAGGA3 approach leads to smaller errors for most of the data sets.

In a second set of experiments (Figure 10.10), the number of generations was limited instead of the total runtime. We also used a larger number of cases which was retrieved from the case base by using the 2-phase approach. For 2 cases, the results are already much better than the YAGGA2 approach without feature transfer as it was proposed in Chapter 6. For 5 cases, the error is further reduced and for some data sets, errors cannot be measured any longer.

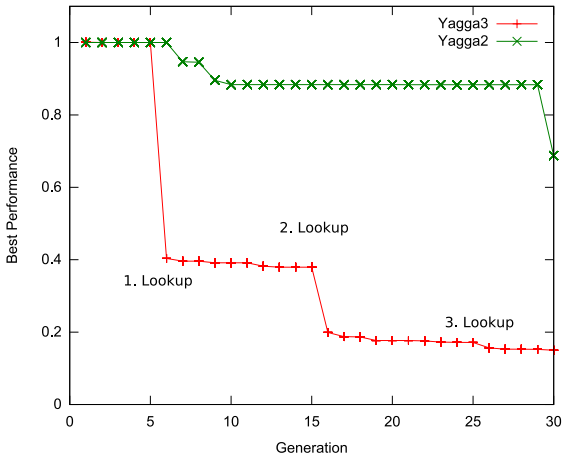



**Figure 10.8.:** The performance of YAGGA2 and YAGGA3 with respect to the number of generations. YAGGA3 performs a case base lookup in the 5th, the 15th, and the 25th generations which results in a clear performance gain and faster convergence times. Source: [138].

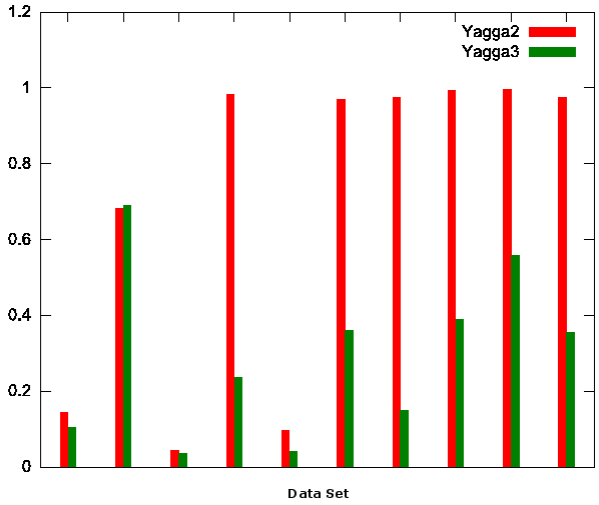



**Figure 10.9.:** Average performance of 10 runs for 10 different data sets after a runtime of 100 seconds. Both algorithms are performed 10 times on each data set. The bars denote the average performance (root mean squared error) and, hence, lower bars are better. It can clearly be seen that YAGGA3 leads to much smaller errors for most of the data sets. Although the standard deviations are not shown in this plot, YAGGA3 delivers significantly better results in 7 out of the 10 cases. Source: [138].

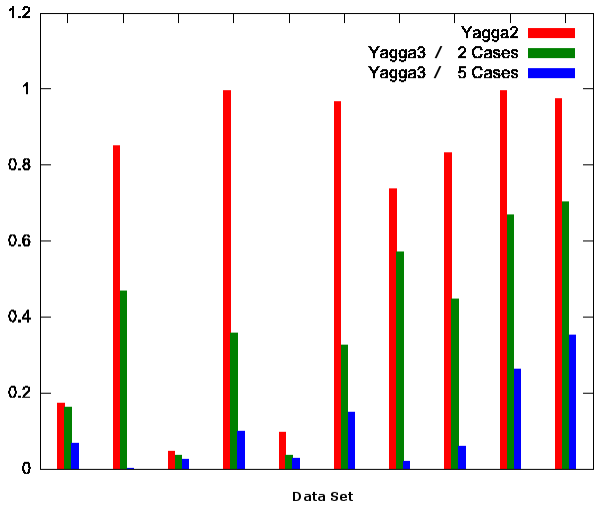



**Figure 10.10.:** Another comparison on the same 10 data sets between YAGGA2 and YAGGA3. The generation number was limited to 100. The best 2 (5) cases delivered for each retrieval process formed the base for feature construction. Although not shown, YAGGA3 based on 5 cases delivers significantly better results in 9 out of the 10 cases. Source: [138].

Please note that the presented approach for feature transfers mainly depends on the selection of a set of base features common for all learning tasks. But even if such a common base feature set is found it must be ensured that the base features in the case base are correctly mapped to (a subset of) features of a new learning task. A follow-up work [138] presents a simple solution for this problem and discusses empirical results demonstrating the success of this base feature mapping approach.

CHAPTER 11

# Conclusion

The main goal of this work was to connect optimization techniques, especially those suitable for non-convex and multi-objective optimization problems, with data mining. We have seen that statistical learning, which today is among the most successful data mining paradigms, inherently defines a multi-objective optimization problem. We made this problem explicit and removed the trade-off factor which has to be used otherwise. Since there is no clear border between the actual learning phase and the preprocessing phase, we tried to transfer the positive results from learning to feature space transformation problems. This led to some interesting results, especially for the task of unsupervised feature space transformations.

## 11.1. Multi-Objective and Non-Convex Learning

We started with a connection between evolutionary computation and statistical learning theory. The idea of large margin methods was very successful in many machine learning and data mining applications. We used the most prominent representative of this paradigm, namely Support Vector Machines, and employed evolution strategies and particle swarm optimization in order to solve the constrained optimization problem at hand. We developed a hybrid mutation which decreases convergence time while the classification accuracy is preserved.

We have seen that *evolutionary SVMs* are at least as accurate as their quadratic programming counterparts. For practical values of $C$, the evolutionary SVM variants frequently outperformed their competitors. With respect to the original fitness function, the evolutionary approach always outperform the traditional SVM. We can conclude that evolutionary algorithms proved as reliable as other optimization schemes for this type of problems.

We then demonstrated how the trade-off between training error and model complexity can be made explicit. We divided the optimization problem of SVMs in two parts and transformed both parts into its dual form of its own. These transformations reduce the runtime for fitness evaluation and provide space for other well-known improvements like incorporating arbitrary kernel functions for non-linear classification tasks which is the key idea of SVMs.

We exploited the new objectives by employing a *multi-objective evolutionary algorithm* after some consequences of the explicit trade-off optimization were discussed. These include the possibility of further reducing the runtime by using only parts of the objectives and the optional usage of a hold-out set in order to produce a hint which areas of the resulting Pareto front should be inspected by the user. This turns the Pareto front of all solutions between minimal training error and minimal model complexity into a powerful tool for controlling the overfitting of machine learning methods. For a first time, this overfitting control does not have to be performed by the user (like setting the parameter $C$ for traditional SVM) but is automatically performed by a learning method. Please note also that all information about these plots are collected in one single run of the algorithm in contrast to wrapper approaches where the learner must be performed once for each point of such an overfitting plot.

The idea of statistical learning theory, i.e. taking the model complexity into account, is simple and appealing. Current approaches, however, did not make use of the inherent trade-off but demanded the definition of a weighting factor of the conflicting criteria from the user. The multi-objective evolutionary SVM proposed in this work is the first solution explicitely solving the basic problem of statistical learning theory.

Besides the inherent advantages of evolutionary algorithms (e. g. parallelization, multi-objective optimization of training error and capacity) it is now also possible to employ non-positive semidefinite or *indefinite kernel functions* which would lead to unsolvable problems for other optimization techniques. As the experiments have shown, an SVM based on evolutionary computation is the first practical solution for this type of problem and outperforms both traditional SVMs as well as Relevance Vector Machines.

Finally, we discussed a problem of high practical relevance which as well leads to a non-convex optimization problem and can also be improved by multi-objective optimizations: *transductive learning*. We defined and transformed all objectives into the dual form and developed a new multi-objective evolutionary transductive SVM. Besides the fact, that this solution again is able to deliver all results in one single optimization run there is a quite more important conclusion: the multi-objective transduction SVM can be seen as a formal connection between supervised and unsupervised learning. In the optimization function of this SVM there is no difference between classification and clustering. It is as simple as this: if no training data is given, we automatically get a clustering scheme. If no test data is given, we end up with the traditional SVM for supervised learning. If both is given, we have a semi-supervised learning scheme lying in between. Thus,

analysts always get the full trade-off between these corner points in form of a 3D Pareto front. This again is achieved with only one single optimization run instead of multiple optimizations.

This formal connection between statistical classification and clustering together with the automatic overfitting control possible in a single optimization run is the major result of the first part of this thesis.

## 11.2. Multi-Objective Feature Space Transformations

In the second part of this work, we tried to transfer the idea of the regularized risk to the problem of feature space transformations. We first discussed a measure for *feature space complexity* before we employed this measure and introduced a simple solution for multi-objective *feature construction* for supervised learning problems. This also includes the problem of multi-objective supervised feature selection.

Thereafter, we discussed a novel way of *supervised feature extraction* based on genetic programming. At first, operators for the analysis of large collections of series data have been presented in a unifying structure. Some new operators have been developed, for instance those in the phase space. Other operators have been generalized, for instance, the windowing and mark-up operators. The operators are organized by method trees, which extract complex features. All known feature extraction methods for the used audio data sets are covered, either directly as an operator, or as the result of a method tree. Many different method trees (features) can be built from the presented primitives. The method trees are automatically generated for a certain classification task by a genetic programming approach.

The construction of method trees has been restricted to functions at the first level, chains concluded by a function at the second level, and to windows embedding chains at higher levels. The complexity of windowings including methods of a certain complexity has been investigated. It was shown under which circumstances windowing decreases runtime, compared to processing the overall value series. Dynamic windowing with reasonable parameters prevents the approach from becoming infinite or exponential in the length of a series. The countable search space for method trees has been shown to be very large but at least not infinite if each transformation is only allowed once. Despite of this large search space, we can again employ a multi-objective selection scheme and the result will be the Pareto front of possible extractions from the most simple to the most accurate ones.

We then moved forward to the task of *unsupervised feature selection* and presented a novel multi-objective evolutionary solution for feature selection in unsupervised machine learning settings. We exemplified this approach on the task of data clustering which plays

an important role in a wide variety of applications ranging from pattern recognition to customer relationship management and web search.

First, it turned out that clustering inherently is a multi-objective optimization problem. There is usually not one correct result as for supervised learning. Users rather explore the space of results interactively to gain insight into the natural patterns within the data set. Second, the approach proposed in this work yields Pareto sets that show *significant inner structure*. This structure is not accidental but reflects patterns in the underlying data. We presented a generic method for an automatic Pareto set segmentation and showed that the discovered segments can be interpreted with respect to unsupervised feature selection. This turns clustering into a reference application of automatic Pareto set analysis.

We argued that these benefits can only be achieved if the optimization problem has been posed in a sound way. Although *maximizing the number of features* during feature selection might sound surprising at first, this paradigm change can be motivated by the aim of unsupervised learning: the search for descriptive, natural patterns. In particular, we have shown that existing approaches do not pose the optimization problem in a sound and robust way. We showed both analytically and empirically, that the corresponding sets of Pareto optimal solutions collapse to a single, trivial solution.

Finally, we presented a multi-objective framework for *feature space transformation* in clustering settings. This new approach is based on the idea of information preservation. As much of the original data space should be preserved as possible, while the validity of the resulting clusters is optimized. We extended our unsupervised feature selection approach to allow for a limited form of feature construction as well. Aggregation is used to derive new features, that generalize over two or more features in the original data set. t-Conorms are a class of theoretically and empirically established generic aggregation functions. They are a natural extension of disjunctions for continuous values, which have proven to be essential for many data mining applications, e.g. generalized association rules. A set of basic conditions limits feature aggregation to the $t$-conorm maximum which summarizes two features with minimal alteration. We show that even for feature aggregation our approach leads to robust Pareto sets. Our experiments support this claim.

The last contribution of this work is a Meta Learning approach to feature construction in order to speed-up the expensive feature space optimization runs. This *feature transfer* approach compares tasks using relevance weights on a common set of base features only. After stating some very basic conditions for such a distance measure, we have shown that an SVM as base feature weighting algorithm and the Manhattan distance fulfill these conditions, while several other popular feature weighting methods and distance measures do not.

We introduced some enhancements for this case based feature construction approach. These enhancements include the distance calculation of already constructed features using a sampling procedure in a 2-phase distance measure. We have then presented experimental results indicating that our method can speed up feature construction considerably. Our approach is therefore highly relevant for practical problems involving feature construction.

The definition of a feature space complexity and the first solution for unsupervised feature space transformations by posing a sound multi-objective optimization problem together with the runtime improvements achieved by using feature transfers are the major results of the second part of this thesis.

In total, we can conclude that the usage of multi-objective evolutionary algorithms can enhance the power of both data mining (Part I) and preprocessing techniques (Part II). The proposed algorithms can cope with non-convex optimization problems. The inherent trade-off between error and complexity can now be made explicit and also solved by employing multi-objective evolutionary optimization techniques. This led not only to a formal connection between classification and clustering but also to an automatic overfitting control for both learning and preprocessing as well.

# Joint Work and Other Work by the Author

This chapter summarizes other work by the author and states the collaborations and contributions of other authors to this work.

## Joint Work

Some parts of this work are adapted versions of joint publications. In the following, the individual contributions are described in detail.

**Chapter 7**
Automatic feature extraction from series data by means of genetic programming was already the topic of the author's masters thesis [106]. Large parts of this chapter builds on the thesis and on a following joint publication with Katharina Morik [115]. For this work, the author has improved smaller parts of the former publications and added the feature space complexity measurement as well as the multi-objective optimization approach for the method tree creation.

**Chapter 8**
Multi-objective feature selection [123] is based on joint work with Michael Wurst who contributed his highly acknowledged knowledge in the field of clustering schemes and clustering evaluation methods.

**Chapter 9**
Multi-objective feature space transformations [124] is joint work with Michael Wurst who again contributed his valuable knowledge of clustering schemes, evaluation measures, and aggregation methods.

**Chapter 10**
The idea for recommendations of feature transfers based on feature weights is the result of joint work with Michael Wurst [120, 121, 122]. The weighting axioms and the corresponding proofs are sole work of the author. The distance axioms and the corresponding proofs are work of Michael Wurst. Some of the figures and empirical results were taken from the master thesis of Michael Nöthe as it was stated in the text.

## Other Publications

This section describes a selection of other publications of the author which were not used as a base of this thesis. Since these publications are at least loosely connected to this work this overview should serve as a starting point for further reading.

**Collaborative Use of Features in a Distributed System for the Organization of Music Collections [118]** This work describes the singe-objective feature extraction approach which was the basis for Chapter 7 together with the feature transfer known from Chapter 10 and uses these techniques in order to support the collaborative structuring of music collections.

**Understandable models of music collections based on exhaustive feature generation with temporal statistics [128]** This work employs a large-scale feature construction approach based on the phase space features discussed in Chapter 7. The goal was to produce understandable models even in this high-dimensional data space.

**A Benchmark Dataset for Audio Classification and Clustering [65]** This work presents a new benchmark data set consisting of freely accessible music files together with extracted features and 39 structures created during a student project.

**Beatles vs. Bach: Merkmalsextraktion im Phasenraum von Audiodaten [103]** This work (German) discusses some new features which can be extracted from the phase space (reconstruction of the state space) of audio data. The resulting features perform quite well for different audio genre classification tasks.

**Localized Alternative Cluster Ensembles for Collaborative Structuring [195]** The LACE algorithm is a novel solution for collaborative structuring tasks exploiting the locally valid structures defined by others.

**Handling Local Patterns in Collaborative Structuring [119]** This work defines a taxonomy for supervised and unsupervised learning tasks and introduces new learning tasks, among them the LACE algorithm mentioned in this work.

**Optimizing Process Plant Layouts [90]** A comparison of different optimization techniques for a layout problem of chemical plants. It turns out that for this type of problems multi-objective evolutionary algorithms deliver the best results.

**Incorporating Fuzzy Knowledge into Fitness: Multi-objective Evolutionary 3D Design of Process Plants [107]** Introduces the multi-objective evolutionary optimization for layouts and combines it with a fuzzy logic based fitness function for 3D process plant layouts.

**On the Automated Creation of Understandable Positive Security Models for Web Applications [18]** This work presents a representation for the normal way of web services accesses and discusses a simple approach to learn such a positive security models from web log data.

**A Flexible Platform for Knowledge Discovery Experiments: Yale – Yet Another Learning Environment [113, 125]** The official publications for the learning environment RapidMiner (formerly Yale) which is the base of all experiments of this work.

# Bibliography


[1] D. W. Aha and Bankert R. L. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and H.-J. Lenz, editors, *Learning from Data*, chapter 4, pages 199–206. Springer, 1996.

[2] David Aha, Dennis Kibler, and Marc Albert. Instance-based learning algorithms. *Machine learning Journal*, 6:37–66, 1991.

[3] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proc. of the 9th National Conference on Artificial Intelligence*, pages 547–552. MIT Press, 1991.

[5] Gerhard Arminger and Norman Götz. Asymmetric loss functions for evaluating the quality of forecasts in time series for goods management systems. SFB475–Report 22, Universität Dortmund, 1999.

[6] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Trans. on Evolutionary Computation*, 1(1):3–17, 1997.

[7] C. Bahlmann and H. Burkhardt. The writer independent on-line handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Trans. Pattern Anal. and Mach. Intell. (TPAMI)*, 26(3):299–310, 2004.

[8] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming – an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufman, 1998.

[9] S. Baumann and T. Pohle. A comparison of music similarity measures for a p2p application. In *Proc. of the 6th International Conference on Digital Audio Effects*, London, UK, 2003.

[10] J. Baxter. Learning internal representations. In *Proc. of the Eighth Annual Conference on Computational Learning Theory (COLT 1995)*, pages 311–320. ACM Press, 1995.

[11] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[12] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Proc. of the Sixteenth Annual Conference on Learning Theory (COLT 2003)*, 2003.

[13] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2(1):125–137, 2001.

[14] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: a comprehensive introduction. *Journal Natural Computing*, 1(1):2–52, 2002.

[15] E. Bloedorn and R. Michalski. Data-driven constructive induction: methodology and applications. In Huan Liu and Hiroshi Motoda, editors, *Feature Extraction, Construction, and Selection – A Data Mining Perpective*, chapter 4, pages 51 – 68. Kluwer, 1998.

[16] E. Bloedornm, J. Wnek, and R. Michalski. Multistrategy constructive induction. In *Proc. of the Second International Workshop on Machine Learning (MSL93)*. Morgan Kaufman, 1993.

[17] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, pages 245–271, 1997.

[18] C. Bockermann, I. Mierswa, and K. Morik. On the automated creation of understandable positive security models for web applications. In *Proc. of the Pervasive Computing (PERCOM 2008)*, 2007.

[19] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A traininig algorithm for optimal margin classifiers. In *Proc. of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[20] E. Bradley. *Intelligent Data Analysis: an introduction*, chapter Time-Series Analysis. Springer, 1999.

[21] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[22] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[23] C. Burges. Geometric methods for feature extraction and dimensional reduction: A guided tour. Technical report, Microsoft Research, 2004.

[24] G. Camps-Valls, J.D. Martin-Guerrero, J.L. Rojo-Alvarez, and E. Soria-Olivas. Fuzzy sigmoid kernel for support vector classifiers. *Neurocomputing*, 62:501–506, 2004.

[25] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[26] P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proc. of 23rd International Conference on Very Large Data Bases (VLDB 1997)*, pages 426–435. Morgan Kaufmann, 1997.

[27] C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

[28] R. Collobert, Sinz, J. F. Weston, and L. Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7(1):1687–1712, 2006.

[29] J. W. Cooley and J. W. Tukey. An algorithm for the machine computation of the complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[31] D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676–1695, 1990.

[32] G. B. Dantzig. *Linear programming and extensions*. Princeton University Press, 1966.

[33] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3):131–156, 1997.

[34] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[35] K. Deb and A. Kumar. Light beam search based multi-objective optimization using evolutionary algorithms. In *Proc. of the Congress on Evolutionary Computation (CEC 2007)*, pages 2125–2132, 2007.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. Technical report, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, 2002.

[37] K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proc. of the 8th annual conference on Genetic and evolutionary computation (GECCO 2006)*, pages 635–642, New York, NY, USA, 2006. ACM.

[38] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[39] P. Deuflhard. *Newton methods for nonlinear problems. Affine invariance and adaptive algorithms.* Springer, 2004.

[40] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. Technical Report Reihe CI 21/98, SFB 531, Universität Dortmund, Germany, 1998.

[41] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, New York, 1973.

[42] J. P. Eakins, J. Edwards, J. Riley, and P. Rosin. A comparison of the effectiveness of alternative feature sets in shape retrieval of multi-component images. In *Proc. of the SPIE 4315, Storage and Retrieval for Media Databases*, pages 196–207, 2001.

[43] R. El-Yaniv, D. Pechyony, and V. Vapnik. Large margin vs. large volume in transductive learning. *Machine Learning*, 72(3):173–188, 2008.

[44] C. Emmanouilidis, A. Hunter, and J. MacIntyre. A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *Proc. of the Congress on Evolutionary Computation (CEC 2000)*, pages 309–316, 2000.

[45] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the International Conference on Knowledge Discorvery in Databases (KDD 1996)*, pages 226–231, 1996.

[46] J. P. Evans and R. E. Steuer. A revised simplex method for linear multiple objective programs. *Mathematical Programming*, 5:375–377, 1973.

[47] T. E. Fawcett and P. E. Utgoff. Automatic feature generation for problem solving systems. In *Proc. of the 9th International Workshop on Machine Learning*, pages 144–153, 1992.

[48] S. Fischer, R. Klinkenberg, I. Mierswa, and O. Ritthoff. YALE: Yet Another Learning Environment – tutorial. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, 2002.

[49] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[50] N.S. Flann and T.G. Dietterich. Selecting appropriate representations for learning from examples. In *AAAI-86*, pages 460–466, Philadelphia, 1986.

[51] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2000.

[52] F. Fleuret and H. Sahbi. Scale-invariance of support vector machines based on the triangular kernel. In *Proc. of the Third International Workshop on Statistical and Computational Theories of Vision (part of ICCV 2003)*, 2003.

[53] J. Foote. Content-based retrieval of music and audio. In *In Multimedia Storage and Archiving Systems II, Proc. of SPIE*, pages 138–147, 1997.

[54] A. Freitas. *Data mining and knowledge discovery with evolutionary algorithms*, chapter 3, pages 55–58. Springer, 2002.

[55] F. Friedrichs and C. Igel. Evolutionary tuning of multiple SVM parameters. In *Proc. of the 12th European Symposium on Artificial Neural Networks (ESANN 2004)*, pages 519–524, 2004.

[56] H. Fröhlich, O. Chapelle, and B. Schölkopf. Feature selection for support vector machines using genetic algorithms. *International Journal on Artificial Intelligence Tools*, 13(4):791–800, 2004.

[57] T. Gevers and H. M. G. Stokman. Robust histrogram construction from color invariants for object recognition. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 26:113–118, 2004.

[58] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: musical information retrieval in an audio database. In *Proc. of ACM Multimedia*, pages 231–236, 1995.

[59] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Compuation*, 7(2):219–269, 1995.

[60] G. Guo and S. Z. Li. Content-based audio classification and retrieval by support vector machines. *IEEE Trans. on Neural Networks*, 14(1):209–215, 2003.

[61] B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.

[62] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[63] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Series in Statistics. Springer, 2001.

[64] J. H. Holland. Escaping brittleness: The possibilities of general–purpose learning algorithms applied to parallel rule–based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning – An Artificial Intelligence Approach*, volume 2, pages 593–624. Morgan Kaufmann, 1986.

[65] H. Homburg, I. Mierswa, B. Möller, K. Morik, and M. Wurst. A benchmark dataset for audio classification and clustering. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR 2005)*, 2005.

[66] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proc. of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 82–87. IEEE Service Center, 1994.

[67] A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. In *Proc. of the IEEE International Conference on Data Mining (ICDM 2003)*, 2003.

[68] T. Howley and M.G. Madden. The genetic kernel support vector machine: Description and evaluation. *Artificial Intelligence Review*, 2005.

[69] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 3(31):264–323, 1999.

[70] W. James and C. Stein. Estimation with quadratic loss. In *Proc. of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability*, pages 361–380, 1960.

[71] N. S. Jayant and P. Noll. *Digital coding of waveforms: principles and applications to speech and video.* Prentice Hall, 1984.

[72] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in kernel methods - support vector learning*, chapter 11. MIT Press, Cambridge, MA, 1999.

[73] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. of the 16th International Conference on Machine Learning*, 1999.

[74] T. Joachims. *Learning to classify text using support vector machines*, volume 668. Kluwer, 2002.

[75] T. Joachims. A support vector method for multivariate performance measures. In *Proc. of the International Conference on Machine Learning (ICML 2005)*, pages 377–384, 2005.

[76] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In William W. Cohen and Haym Hirsh, editors, *Proc. of the 11th International Conference on Machine Learning ICML94*, pages 121–129, San Francisco, USA, 1994. Morgan Kaufmann.

[77] T. Kahveci and A. K. Singh. An efficient index structure for string databases. In *Proc. of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 352–360. Morgan Kaufmann, 2001.

[78] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of the International Conference on Artificial Neural Networks (ICANN 1995)*, pages 1942–1948, 1995.

[79] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast classification, clustering and relevance feedback. In *Proc. of the 4th Conference on Knowledge Discovery in Databases (KDD 1998)*, pages 239–241, 1998.

[80] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Proc. of the 3rd Conference on Knowledge Discovery in Databases (KDD 1997)*, pages 24–30, 1997.

[81] Y. Kim, W. N. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD 2000)*, pages 365–369. ACM Press, 2000.

[82] Y. Kim, W. N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6:531–556, 2002.

[83] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

[84] K. Kira and L. Rendell. The feature selection problem: traditional methods and a new algorithm. In *Proc. of the Tenth National Conference on Artificial Intelligence*, pages 129–134. MIT Press, 1992.

[85] P. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium*, pages 91–102. Springer, 1998.

[86] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*, 8(3):281–300, 2004.

[87] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence Journal, Special Issue on Relevance*, 97(1–2):273–324, 1997.

[88] R. Kohavi and G. H. John. The wrapper approach. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction, and Selection: A Data Mining Perspective*, pages 33–50. Kluwer, 1998.

[89] D. Koller and M. Sahami. Toward optimal feature selection. In *Proc. of the 13th International Conference on Machine Learning (ICML 1996)*, pages 129–134, 1996.

[90] H. Köpcke and I. Mierswa. Optimizing process plant layouts. In *Proc. of the 6th International Symposium on Tools and Methods of Competitive Engineering (TMCE 2006)*, 2006.

[91] J.R. Koza. *Genetic Programming: on the programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[92] F. Kurth and M. Clausen. Full-text indexing of very large audio data bases. In *110th Convention of the Audio Engineering Society*, 2001.

[93] W. B. Langdon, T. Soule, R. Poli, and J. A Foster. The evolution of size and shape. In *Advances in Genetic Programming 3*, pages 163–190. MIT Press, 1999.

[94] P. Langley. *Elements of machine learning*. Morgan Kaufmann, Inc. San Francisco, California, 1996.

[95] N. Lavrac, D. Gamberger, and P. Turney. A relevancy filter for constructive induction. In H. Liu and H. Motoda, editors, *Feature Extraction Construction and Selection – A Data Mining Perspective*, pages 137–154. Kluwer, 1998.

[96] H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods, 2003.

[97] Z. Liu, Y. Wang, and T. Chen. Audio feature extraction and analysis for scene segmentation and classification. *Journal of VLSI Signal Processing System*, 20, 1998.

[98] G. Loy. Musicians make a standard: the MIDI phenomenon. *Computer Music Journal*, 9(4), 1989.

[99] S. W. Mahfoud. Niching methods. In *Evolutionary Computation 2: Advanced Algorithms and Operators*, pages 87–92. Institute of Physics Publishing, 2000.

[100] X. Mary. *Hilbertian subspaces, subdualities and applications*. PhD thesis, Institut National des Sciences Appliquees Rouen, 2003.

[101] J. Mehnen, H. Traumann, and A. Tiwari. Introducing user preference using desirability functions in multi-objective evolutionary optimisation of noisy processes. In *Proc. of the Congress on Evolutionary Computation (CEC 2007)*, pages 2687–2694, 2007.

[102] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Trans. of the Royal Society*, A 209:415–446, 1909.

[103] I. Mierswa. Beatles vs. Bach: Merkmalsextraktion im Phasenraum von Audiodaten. In *Proc. of the Annual National German Workshop on Machine Learning (FGML 2003)*, 2003.

[104] I. Mierswa. Automatic feature extraction from large time series. In C. Weihs and W. Gaul, editors, *Proc. of the 28. Annual Conference of the GfKl 2004*, pages 600–607. Springer, 2004.

[105] I. Mierswa. Automatic feature extraction from large time series. In A. Abecker, S. Bickel, U. Brefeld, I. Drost, N. Henze, O. Herden, M. Minor, T. Scheffer, L. Stojanovic, and S. Weibelzahl, editors, *Proc. of the Annual National German Workshop on Machine Learning (FGML 2004)*, 2004.

[106] I. Mierswa. Automatisierte Merkmalsextraktion aus Audiodaten. Master's thesis, Fachbereich Informatik, Universität Dortmund, 2004.

[107] I. Mierswa. Incorporating fuzzy knowledge into fitness: Multiobjective evolutionary 3d design of process plants. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, 2005.

[108] I. Mierswa. Evolutionary learning with kernels: A generic solution for large margin problems. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2006)*, 2006.

[109] I. Mierswa. Making indefinite kernel learning practical. Technical report, Collaborative Research Center 475, University of Dortmund, 2006.

[110] I. Mierswa. Controlling overfitting with multi-objective support vector machines. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007.

[111] I. Mierswa. Finding all local models in parallel: Multi-objective SVM, 2007. Talk at the Dagstuhl Seminar on Local Models.

[112] I. Mierswa. Regularization through multi-objective optimization. In R. Klinkenberg, I. Mierswa, A. Hinneburg, S. Posch, and S. Neumann, editors, *Proc. of LWA 2007 - Lernen - Wissensentdeckung - Adaptivität*, 2007.

[113] I. Mierswa, R. Klinkenberg, S. Fischer, and O. Ritthoff. A flexible platform for knowledge discovery experiments: YALE – Yet Another Learning Environment. In *Proc. of the Annual National German Workshop on Machine Learning (FGML 2003)*, 2003.

[114] I. Mierswa and K. Morik. Learning feature extraction for learning from audio data. Technical Report 55/04, Collaborative Research Center 475, University of Dortmund, 2004.

[115] I. Mierswa and K. Morik. Automatic feature extraction for classifying audio data. *Machine Learning Journal*, 58:127–149, 2005.

[116] I. Mierswa and K. Morik. Evolutionäre Aufzucht von Methodenbäumen zur Merkmalsextraktion aus Audiodaten. *Informatik Spektrum, Themenheft Musik*, 28(5):381–388, 2005.

[117] I. Mierswa and K. Morik. Method trees: Building blocks for self-organizable representations of value series. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2005), Workshop on Self-Organization in Representations for Evolutionary Algorithms: Building complexity from simplicity*, 2005.

[118] I. Mierswa, K. Morik, and M. Wurst. Collaborative use of features in a distributed system for the organization of music collections. In Shen, Shepherd, Cui, and Liu, editors, *Intelligent Music Information Systems: Tools and Methodologies*, pages 147–175. Information Science Reference, 2007.

[119] I. Mierswa, K. Morik, and M. Wurst. Handling local patterns in collaborative structuring. In Florent Masseglia, Pascal Poncelet, and Maguelonne Teisserie, editors, *Successes and New Directions in Data Mining*, pages 167–186. Information Science Reference, 2007.

[120] I. Mierswa and M. Wurst. Efficient case based feature construction for heterogeneous learning tasks. Technical Report CI-194/05, Collaborative Research Center 531, University of Dortmund, 2005.

[121] I. Mierswa and M. Wurst. Efficient case based feature construction for heterogeneous learning tasks. In J. Gama et al., editor, *Proc. of the European Conference on Machine Learning (ECML 2005)*, LNAI 3720, pages 641–648. Springer, 2005.

[122] I. Mierswa and M. Wurst. Efficient feature construction by meta learning – guiding the search in meta hypothesis space. In *Proc. of the Internation Conference on Machine Learning (ICML 2005), Workshop on Meta Learning*, 2005.

[123] I. Mierswa and M. Wurst. Information preserving multi-objective feature selection for unsupervised learning. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2006)*, 2006.

[124] I. Mierswa and M. Wurst. Sound multi-objective feature space transformation for clustering. In *In Proc. of the Knowledge Discovery, Data Mining, and Machine Learning (KDML 2006)*, 2006.

[125] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. YALE: Rapid prototyping for complex data mining tasks. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, 2006.

[126] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Ratsch. Kernel PCA and de–noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[127] B. L. Miller and M. J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. Technical Report IlliGAL Report No. 95010, Department of General Engineering, University of Illinois, 1995.

[128] F. Mörchen, I. Mierswa, and A. Ultsch. Understandable models of music collections based on exhaustive feature generation with temporal statistics. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, 2006.

[129] P. J. Moreno and P. P. Ho. A new SVM approach to speaker identification and verification using probabilistic distance kernels. In *Proc. of Eurospeech 2003*, 2003.

[130] K. Morik. The representation race - preprocessing for handling time phenomena. In *Proc. of the European Conference on Machine Learning (ECML 2000)*, pages 4–19. Springer, 2000.

[131] K. Morik and S. Wessel. Incremental signal to symbol processing. In K. Morik, M. Kaiser, and V. Klingspor, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning*, chapter 11, pages 185–198. Kluwer Academic Publishers, 1999.

[132] M. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In *Proc. of the 7th International Conference on Document Analysis and Recognition (ICDAR 2003)*, 2003.

[133] F. Murtagh. *Clustering in massive data sets*, pages 501–543. Kluwer Academic Publishers, 2002.

[134] R. M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, Department of Computer Science, University of Toronto, 1993.

[135] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1964.

[136] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[137] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 2006.

[138] M. Nöthe. Erweiterung evolutionärer Merkmalskonstruktionen um Case Base Retrieval. Master's thesis, Fachbereich Informatik, Universität Dortmund, 2008.

[139] A. L. Oliveira and A. Sangiovanni-Vincentelli. Constructive induction using a non-greedy strategy for feature selection. In *Proc. of the International Conference on Machine Learning*, pages 354–360. Aberdeen, 1992.

[140] C. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *Proc. of the 21st International Conference on Machine Learning (ICML 2004)*, pages 639–646, 2004.

[141] E. Osuna, R. Freund, and F. Girosi. Support vector machines: training and applications. Technical Report AIM-1602, Massachusetts Institute of Technology, 1997.

[142] A. Osyczka. *Multicriterion optimization for engineering design*, chapter 7, pages 193–227. Academic Press, 1985.

[143] J. Pickens. A survey of feature selection techniques for music information retrieval. Technical report, Center of Intelligent Information Retrieval, Department of Computer Science, University of Masschusetts, 1996.

[144] J. Platt. *Advances in large margin classifiers*, chapter Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. MIT Press, 1999.

[145] R. Poli, J. E. Rowe, and N. McPhee. Markov chain models for GP and variable-length GAs with homologous crossover. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 112–119, 2001.

[146] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[147] R. J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[148] C. E. Rasmussen and J. Quinonero-Candela. Healing the relevance vector machine through augmentation. In *Proc. of the 22nd International Conference on Machine learning (ICML 2005)*, pages 689–696. ACM Press, 2005.

[149] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Trans. on Evolutionary Computation*, 4, 2000.

[150] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[151] A. Ribbrock and F. Kurth. A full-text retrieval approach to content-based audio identification. In *International Workshop on Multimedia Signal Processing*, 2002.

[152] D. Richardson. Some unsolvable problems involving elementary functions of a real variable. *Journal of Symbolic Logic*, 33:514–520, 1968.

[153] F. Riesz and B. Nagy. *Functional analysis*. Frederick Ungar Publishing Co., 1955.

[154] O. Ritthoff and R. Klinkenberg. Evolutionary feature space transformation using type-restricted generators. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 1606–1607, 2003.

[155] O. Ritthoff, R. Klinkenberg, S. Fischer, and I. Mierswa. A hybrid approach to feature selection and generation using an evolutionary algorithm. In John A. Bullinaria, editor, *Proc. of the 2002 U.K. Workshop on Computational Intelligence (UKCI 2002)*, pages 147–154, 2002.

[156] E. Rivlin and I. Weiss. Deformation invariants in object recognition. *Computer Vision and Image Understanding*, 65:95–108, 1995.

[157] J. P. Rosca. Generality versus size in genetic programming. In *Genetic Programming 1996: Proc. of the 1st Annual Conference*, pages 381–387. Morgan Kaufman, 1996.

[158] V. Roth and T. Lange. Feature selection in clustering problems. In *Proc. of Neural Information Processing Systems (NIPS 2003)*, 2003.

[159] T. P. Runarsson and S. Sigurdsson. Asynchronous parallel evolutionary model selection for support vector machines. *Neural Information Processing*, 3(3):59–67, 2004.

[160] S. Rüping. *mySVM Manual*. Universität Dortmund, Lehrstuhl Informatik VIII, 2000.
`http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

[161] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Minining and Knowledge Discovery*, 2(2):169–194, 1998.

[162] R. Schlittgen and B. H. J. Streitberg. *Zeitreihenanalyse*. Oldenburg, 2001.

[163] B. Schölkopf. The kernel trick for distances. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000*, pages 301–307. MIT Press, 2001.

[164] B. Schölkopf and A. J. Smola. *Learning with kernels – support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[165] M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2):1–38, 2004.

[166] M. M. Silva, T. T. Maia, and A. P. Braga. An evolutionary approach to transduction in support vector machines. In *Proc. of the Fifth International Conference on Hybrid Intelligent Systems*, pages 6–11, 2005.

[167] A. Smola, B. Schölkopf, and K.-R. Müller. General cost functions for support vector regression. In *Proc. of the 8th International Conference on Artificial Neural Networks*, pages 79–83, 1998.

[168] A. J. Smola, Z. L. Ovari, and R. C. Williamson. Regularization with dot-product kernels. In *Proc. of the Neural Information Processing Systems (NIPS 2000)*, pages 308–314, 2000.

[169] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, NeuroCOLT2 Technical Report Series, 1998.

[170] T. Soule, J. A. Foster, and J. Dickinson. Code growth in genetic programming. In *Genetic Programming 1996: Proc. of the 1st Annual Conference*, pages 215–223. Morgan Kaufman, 1996.

[171] R. Srikant and R. Agrawal. Mining generalized association rules. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proc. of 21th International Conference on Very Large Data Bases (VLDB 1995)*, pages 407–419. Morgan Kaufmann, 1995.

[172] I. Stahl. Predicate invention in inductive logic programming. In Luc DeRaedt, editor, *Advances in Inductive Logic Programming*, pages 34 – 47. IOS Press, 1996.

[173] Statlib – datasets archive. http://lib.stat.cmu.edu/datasets/.

[174] D. Steuer. Multi-criteria-optimisation and desirability indices. Technical Report 20, Universität Dortmund, FB Statistik, 1999.

[175] T. Storch. On the impact of objective function transformations on evolutionary and black-box algorithms. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 833–840, 2005.

[176] F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L. S. Young, editors, *Dynamical systems and turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer, 1980.

[177] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proc. of the International Conference on Machine Learning (ICML 2005)*, 2005.

[178] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In L. Saitta, editor, *Proc. of the 13th International Conference on Machine Learning (ICML 1996)*. Morgen Kaufmann, 1996.

[179] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[180] H. Trautmann and C. Weihs. On the distribution of the desirability index using harringtons desirability function. *Metrika*, 63(2):207–213, 2005.

[181] G. Tzanetakis. *Manipulation, analysis and retrieval systems for audio signals*. PhD thesis, Computer Science Department, Princeton University, 2002.

[182] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR 2001)*, pages 205–210, 2001.

[183] H. Vafaie and K. De Jong. Evolutionary feature space transformation. In Huan Liu and Hiroshi Motoda, editors, *Feature Extraction, Construction, and Selection – A Data Mining Perpective*, pages 307–323. Kluwer, 1998.

[184] V. N. Vapnik. *The nature of statistical learning theory*. Springer, 1995.

[185] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[186] V. N. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

[187] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

[188] H. Wang, D. Bell, and F. Murtagh. Relevance approach to feature subset selection. In H. Liu and H. Motoda, editors, *Feature Extraction Construction and Selection – A Data Mining Perspective*, pages 85–99. Kluwer, 1998.

[189] C. Weihs, S. Berghoff, P. Hasse-Becker, and U. Ligges. *Mathematical Statistics and Biometrical Applications*, chapter Assessment of purity of intonation in singing presentations by discriminant analysis, pages 395–410. Josef Eul, Bergisch-Gladbach, Köln, 2001.

[190] I. Witten and E. Frank. *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[191] J. Wnek and R. S. Michalski. Hypothesis–driven constructive induction in aq17: A method and experiments. Reports of Machine Learning and Inference Laboratory P 91-9 MLI 91-4, Center for Artificial Intelligence, George Mason University, Fairfax, VA 22030, 1991.

[192] W. H. Wolberg, W. N. Street, D. M. Heisey, and O. L. Mangasarian. Computer-derived nuclear "grade" and breast cancer prognosis. *Analytical and Quantitative Cytology and Histology*, 17:257–264, 1995.

[193] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fé Institute, Santa Fé, CA., 1995.

[194] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimisation. *IEEE Trans. on Evolutionary Computation*, 1:67–82, 1997.

[195] M. Wurst, K. Morik, and I. Mierswa. Localized alternative cluster ensembles for collaborative structuring. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proc. of the European Conference on Machine Learning*, pages 485–496, Berlin, 2006. Springer.

[196] J. Yang and V. Honovar. Feature subset selection using a genetic algorithm. In *IEEE Intelligent Systems (Special Issue on Feature Transformation and Subset Selection)*, volume 13, pages 44–49. Kluwer, 1998.

[197] B. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time series under time warping. In *Proc. of the 14th Conference on Data Engineering*, pages 201–208, 1998.

[198] P. L. Yu and M. Zeleny. The set of all nondominated solutions in linear cases and a multicriteria Simplex method. *Journal of Mathematical Analysis and Applications*, 49:430–468, 1975.

[199] A. Zellner. Bayesian estimation and prediction using asymmetric loss functions. *Journal of the American Statistical Association*, 81:446–451, 1992.

[200] K. Zhang, J. T. L. Wang, and D. Shasha. On the editing distance between undirected acyclic graphs and related problems. In *Proc. of the 6th Annual Symposium on Combinatorial Pattern Matching*, pages 395–407. Springer, 1995.

[201] T. Zhang and C. Kuo. Content-based classification and retrieval of audio. In *Proc. of the Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations*, 1998.

[202] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.