Self-adaptive Structure Semi-supervised Methods for Streamed Emblematic Gestures

Von der Fakultät
Elektrotechnik und Informationstechnik
der Technischen Universität Dortmund
genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
eingereicht von

Husam Jumaah Naeemah Al-Behadili

Dortmund, 2017

Tag der mündlichen Prüfung:	25.10.2017
Hauptreferent:	Prof. Dr. rer. nat. habil. Christian Wöhler
Korreferent:	Prof. Dr.-Ing. Jürgen Götze

Arbeitsgebiet Bildsignalverarbeitung Technische Universität Dortmund

Abstract

Although many researchers try to improve the level of machine intelligence, there is still a long way to achieve an intelligence similar to what humans have. Scientists and engineers are continuously trying to increase the level of smartness of the modern technology, i.e. smartphones and robotics. Humans communicate with each other by using the voice and gestures. Hence, gestures are essential to transfer the information to the partner. To reach a higher level of intelligence, the machine should learn from and react to the human gestures, which means learning from continuously streamed gestures. This task faces serious challenges since processing streamed data suffers from different problems. Besides the stream data being unlabelled, the stream is long. Furthermore, "concept-drift" and "concept evolution" are the main problems of them. The data of the data streams have several other problems that are worth to be mentioned here, e.g. they are: dynamically changed, presented only once, arrived at high speed, and non-linearly distributed. In addition to the general problems of the data streams, gestures have additional problems. For example, different techniques are required to handle the varieties of gesture types. The available methods solve some of these problems individually, while we present a technique to solve these problems altogether.

Unlabelled data may have additional information that describes the labelled data more precisely. Hence, semi-supervised learning is used to handle the labelled and unlabelled data. However, the data size increases continuously, which makes training classifiers so hard. Hence, we integrate the incremental learning technique with semi-supervised learning, which enables the model to update itself on new data without the need for the old data. Additionally, we integrate the incremental class learning within the semi-supervised learning, since there is a high possibility of incoming new concepts in the streamed gestures. Moreover, the system should be able to distinguish among different concepts and also should be able to identify random movements. Hence, we integrate the novelty detection to distinguish between the gestures that belong to the known concepts and those that belong to unknown concepts. The extreme value theory is used for this purpose, which overrides the need for additional labelled data to set the novelty threshold and has several other supportive features. Clustering algorithms are used to distinguish among different new concepts and also to identify random movements. Furthermore, the system should be able to update itself on only the trusty assignments, since updating the classifier on wrongly assigned gesture affects the performance of the system. Hence, we propose confidence measures for the assigned labels.

We propose six types of semi-supervised algorithms that depend on different techniques to handle different types of gestures. The proposed classifiers are based on the Parzen window classifier, support vector machine classifier, neural network (extreme learning machine), Polynomial classifier, Mahalanobis classifier, and nearest class mean classifier. All of these classifiers are provided with the mentioned features. Additionally, we submit a wrapper method that uses one of the proposed classifiers or ensemble of them

to autonomously issue new labels to the new concepts and update the classifiers on the new incoming information depending on whether they belong to the known classes or new classes. It can recognise the different novel concepts and also identify random movements.

To evaluate the system, we acquired gesture data with nine different gesture classes. Each of them represents a different order to the machine e.g. come, go, etc. The data are collected using the Microsoft Kinect sensor. The acquired data contain 2878 gestures achieved by ten volunteers. Different sets of features are computed and used in the evaluation of the system. Additionally, we used real data, synthetic data and public data as support to the evaluation process.

All the features, incremental learning, incremental class learning, and novelty detection are evaluated individually. The outputs of the classifiers are compared with the original classifier or with the benchmark classifiers. The results show high performances of the proposed algorithms.

Acknowledgements

I'm genuinely grateful to my supervisor, Prof. Christian Wöhler for his unlimited support. Words cannot express my thanks for all valuable suggestions, advice, learning opportunities, and trust that he provided me during this dissertation. His reviews encouraged me to be more precise in my work, and his critiques guided me to think more prudently about the proposed approaches.

I'd also like to thank my colleague Dr. Arne Grumpe. His comprehensive interdisciplinary knowledge and his dynamic personality caused new impulses to my work. This dissertation is hard to complete without his tips and suggestions.

I further would like to express my thank to my colleague Daniela Rommel. She did not hesitate to provide me with any assistance. Similar thanks, for all other colleagues, who I enjoyed working with them and with their discussions. I'm fortunate to have such a friendly group who make my life here so excellent. I also would like to thank my friend Majed Edan, who support me by being my formal representative in my country during my research.

I also thank the "Establishment of Martyrs" for funding, along with the management of the Iraqi Ministry of Higher Education.

Finally, I would like to express my thanks and gratitude to my parents and my family, particularly my wife, for their love, understanding and strong support along the way. Great thanks for my children Jumana, Mohammed and Wafaa for being patients while I'm busy and away from them.

List of related Publications

This thesis is based on the following publications of the author. The publications are listed in descending chronological order.

Peer-reviewed journal articles and book chapters

(Al-Behadili et al., 2016c) Al-Behadili, H., Grumpe, A., and Wöhler, C. Confidence band and extreme value theory based outlier detection for semi-supervised learning of incremental polynomial classifier. *International Journal of Simulation Systems, Science & Technology*, 17(34), 2016c.

(Al-Behadili et al., 2016b) Al-Behadili, H., Grumpe, A., Migdadi, L., and Wöhler, C. Incremental parzen window classifier for a multi-class system. *International Journal of Simulation–Systems, Science & Technology*, 17(34), 2016b.

(Al-Behadili et al., 2014) Al-Behadili, H., Grumpe, A., and Wöhler, C. Semi-supervised learning of emblematic gestures. *At-Automatisierungstechnik*, 62(10):732–739, 2014.

Conference contributions

(Al-Behadili et al., 2016e) Al-Behadili, H., Grumpe, A., and Wöhler, C. Non-linear distance-based semi-supervised multi-class gesture recognition. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP,*, pages 280–286, 2016e. ISBN 978-989-758-175-5. doi: 10.5220/0005674102800286.

(Al-Behadili et al., 2016d) Al-Behadili, H., Grumpe, A., and Wöhler, C. Neural network based novelty detection for incremental semi-supervised learning in multi-class gesture recognition. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP,*, pages 287–294, 2016d. ISBN 978-989-758-175-5. doi: 10.5220/0005674202870294.

(Al-Behadili et al., 2015c) Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Non-linear distance based large scale data classifications. In *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 613–617, Dec 2015c. doi: 10.1109/PIC.2015.7489921.

(Al-Behadili et al., 2015d) Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Incremental learning and novelty detection of gestures using extreme value theory. In *IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, pages 169–174, Nov 2015d. doi: 10.1109/CGVIS.2015.7449915.

(Al-Behadili et al., 2015a) Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Extreme learning machine based novelty detection for incremental semi-supervised learning. In *Third IEEE International Conference on Image Information Processing (ICIIP)*, pages 230–235, Dec 2015a. doi: 10.1109/ICIIP.2015.7414771.

(Al-Behadili et al., 2016a) Al-Behadili, H., Grumpe, A., Migdadi, L., and Wöhler, C. Semi-supervised learning using incremental support vector machine and extreme value theory in gesture data. In *18th IEEE International Conference on Computer Modelling and Simulation (UKSim-AMSS)*, pages 184–189, 2016a. doi: 10.1109/UKSim.2016.5.

(Al-Behadili et al., 2015b) Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Incremental class learning and novel class detection of gestures using ensemble. In *Workshop New Challenges in Neural Computation 2015*, pages 122–132, 2015b.

(Al-Behadili et al., 2015e) Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Semi-supervised learning using incremental polynomial classifier and extreme value theory. In *3rd IEEE International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 332–337, 2015e. doi: 10.1109/AIMS.2015.60.

(Al-Behadili et al., 2015f) Al-Behadili, H., Grumpe, A., and Wöhler, C. Incremental learning and novelty detection of gestures in a multi-class system. In *3rd IEEE International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 304–309, 2015f. doi: 10.1109/AIMS.2015.55.

Contents

1

Introduction

Sometimes when humans speak about whatever subject, they use some gestures to give more information about the subject or make it more spectacular. For example, if someone narrates an adventure of climbing, he may use his hand to mimic the event (McNeill, 2008). Although the gestures are mostly performed by hand, other parts of the body may be utilised in the gesture's forming e.g. head, the whole body or even by the face, such as nodding the head to say yes. The gestures also may be used to give an order or say something to another human. For example moving the index finger repeatedly towards the person, who is performing the gesture, while other fingers are clenched fist is meaning "come here" in Northern Europe Fig. 1.1(a) and extending the thumb upward gives approval Fig. 1.1(b).

These motions are used to communicate with machines in Human-Machine Interaction (HMI) and they are called "gesture". The Human-Machine interface starts moving toward using gestures and voices as an input to smart devices instead of keyboards, mice, gloves or other input devices. This is because the interaction through the gesture is exactly like the conversation with other humans. Additionally, with the new technology of virtual reality displays, mice and keyboards have been very limited to use as input devices (Pavlovic et al., 1997). If a high accuracy of gesture recognition is achieved, great applications can be implemented using gestures. An example, an application that control many monitors by using gestures, e.g. if you point towards a monitor it will turn on, if you show your palm toward the another it will stop or may move your hand from right to left to go to the next page, etc., exactly as in the science fiction films. If the social robotics can understand the gestures, its reactions will be like humans. Gesture recognition is included in a wide range of computer games e.g. (Lee et al., 2005; Lee and Hong, 2010). Furthermore, Gestures are considered an intuitive, fast and safe way in HMI (Wachs et al., 2006; Soutschek et al., 2008; Yusoff et al., 2013). Hence, an important application for gestures are the medical devices, for example, during the surgery touching the keyboards, mice or any other non-medical tools may cause a spread of infection hence the gesture will be the ideal solution

(a) Come here (b) Approval

Figure 1.1: Some of Well-Known Gestures[1].

for this case (Wachs et al., 2006).

Unfortunately, gesture recognition is not as simple as recognising the words written by the keyboard, since gestures are done freely mid-air. The same type of gesture cannot be implemented in the same manner by different persons. Additionally, the same individual might change his/her performing of the same gesture from time to time. Moreover, the gestures are used to simulate the real communications, this leads to the possibility of receiving data continuously from the data stream. The significant problems of data streaming are the "infinite length" , "concept-drift" and "concept-evolution". They were addressed by most of the existing online algorithms (Masud et al., 2012). Due to the infinite length, the size of the data collected from the stream will be increased, and then both unlimited hardware resources and an indefinitely long time will be required to train the system with these data. The concept-drift, in contrast, requires the system to be retrained with the data continuously to overcome the effects of concept-drift, e.g. changing statistical properties of the classes or the appearance of novel classes. It will be very complicated to retrain the system continuously if the data originate from streams of infinite length. The immersion of new classes within time, the vanishing of some of the existing classes, non-linearly separable classes, and the scarcity of labelled data are additional serious problems of data streams. Most of the studies of data streams offer a solution to each problem separately, but they happen together in real-life. Hence, we need to

[1]The photos are downloaded from https://morguefile.com/

resolve all these problems ("infinite length", "concept-drift", "novel classes", "unlabelled training data", "unlimited size of training data", "continuous incoming training data", "non-linearly separable classes" and "multi-class system") in one algorithm.

In principle, the classifier should be trained with all possible gestures performed in all possible manners to get an acceptable recognition rate. As mentioned the manually labelled data in the data stream are costly and it is impossible to get labels for all possible gestures. In contrast, unlabelled data may be streamed continuously. Hence, a possible solution is to use semi-supervised learning instead of fully supervised learning. Semi-supervised learning corresponds to first training a classifier on a small number of manually labelled training samples in a fully supervised manner and updating the training set using the labels assigned by the classifier itself (Zhu and Goldberg, 2009). Semi-supervised approaches are also called "self-training" (Rosenberg et al., 2005; Zhu and Goldberg, 2009). Thus, by using the semi-supervised learning scenario the model should update itself autonomously to new partner's gestures instead of enforcing the persons to limited ways of performing the gesture.

Recently, the most interesting field to companies is to submit smart devices that can react like humans and may learn independently, i.e. simulate the intelligence of humans. To achieve these features, the system should react to the actions around it in real-time and also be able to learn new information and new concepts from the data stream. Therefore, the system should recognise if the samples belong to the known concepts or not. If a sample belongs to the known concept, the system should measure if the system believes its assignment to update its information if this assignment is highly believable. While, if the sample does not belong to the known concepts, the system should be able to recognise if it is a random movement or it belongs to a novel concept. Thus, if more than one group of data belong to novel concepts, the system should be able to recognise the various concepts and update its structure to these new classes. All of these processes should be implemented independently by the system without the intervention of the human. We expect implementing these features increases the smartness of the machine, which may help to improve the technology that used to implement the virtual reality.

We submit several semi-supervised methods to fit the various types of gesture data. The system that we proposed has the ability to label the new gesture with three outputs. The first output is the label of the most probable class. The second output represents the believability flag, which will be set only if the first output is trusted. This flag is used in the semi-supervised learning to select only the samples that the system is sure of their labels (in the standard incremental learning). The last output is the novelty flag, which is set on only if the sample does not belong to the concept the classifier was trained on. This flag helps to distinguish between the gesture and random movements and those gestures belonging to novel/unseen class. The novelty flag is used in the process of updating the classifier on new classes by the incremental class learning. A sample of the classifier output is shown in Table 1.1.

Table 1.1: A sample of the proposed algorithms' outputs, where each row represents different test sample. The first column corresponds to the most similar class label; the second column is the believability flag which indicates if the label in the first column is believable, and the last column contains the novelty flag.

Sample No.	1	2	3	4	5	6	7	8	9	10	11	12	13
Label	3	7	9	6	2	4	5	7	4	5	9	1	3
Believability	1	0	0	1	1	1	0	0	1	0	1	1	0
Novelty	0	0	1	0	0	0	1	1	0	0	0	0	1

1.1 Thesis Outline

The reminding parts of the thesis are arranged as follows. Reviews of the related work for the gesture recognition, and semi-supervised learning are provided in Chapter 2 and Chapter 3, respectively. The acquired and used datasets are presented in Chapter 4. Chapter 5 – Chapter 7 present the details of the proposed semi-supervised methods of non-parametric, non-linear classifiers and Metric Learning Based Semi-supervised, respectively. A novel algorithm for the self-adaptive structure semi-supervised learning is proposed in Chapter 8. The last chapter concludes the thesis and puts the suggestion of the future extension.

1.2 Contribution

Due to the varieties of the gesture data, we present several accurate semi-supervised methods that can update themselves autonomously to the concept-drift. The semi-supervised methods are incorporated with the ability to indicate the outliers and the novel gestures. As the data are streamed continuously, an incremental learning is proposed for each algorithm to reduce the time of the retraining in the online system and the ability to update the class structure of the classifier is also combined. The summary of the contribution of the thesis include:

- Proposal of a professional gesture data set that can be integrated into robotic systems or smart devices to control them.
- A simple and effective method is presented to spot the gestures. Where the segmentation of the gesture is the most difficult challenge that faces the gesture recognition systems. This is done by presuming the pre- and post-stroke. We assume the gesture is occurring between two non-moving hand points. Additionally, the velocity and the direction of the movement are taken into account.

- Proposal of efficient and effective features for the gesture recognition. This help to represent the gesture with compact features, which leads to fast computations. As an example of these features, we used the PCA to detect the most effective dimensions, where we encode the movement direction in the new space by one feature called "direction code".
- An incremental computational of the covariance is proposed and used in the Mahalanobis distance to produce incremental metric learning. Then manipulate the output of the metric learning to fit the extreme value theory (EVT) for novelty detection.
- An incremental approximation method for Parzen window is proposed. The output is adapted to fit the EVT.
- The output of incremental SVM classifier is adapted to fit the EVT.
- A confidence band calculation method is proposed on the output of the extreme machine learning (ELM) to use it in the novelty detection in combination with the output of the EVT. The outputs of the ELM are also manipulated to fit the EVT. Additionally, we proposed a method to update both the confidences band and the residual of the outputs incrementally.
- Proposal of a method of using the ELM as auto-encoder for novelty detection.
- Computation of a confidence band on the output of the polynomial classifier and making use of it in the novelty detection in combination with the output of the EVT. The outputs of the polynomial are also adapted to fit the EVT. Both the residuals and the confidences band are updated incrementally.
- Proposal of a method to use the kernel trick in metric learning with decreasing the dimension instead of increasing it.
- Implementation of the incremental learning and incremental class learning for all proposed classifiers.
- Proposal of a general method to update the structure of the classifier (and also ensemble classifier) independently based on incremental learning and incremental class learning method.
- Proposal of a method to increment the number of classes autonomously using the metric learning and the polynomial classifier.

2

Related work: Gesture Recognition Systems

The main aim of gesture recognition approaches is to implement algorithms that can recognise particular human gestures and response to these gestures in a way that fit the system requirements, which is served by the recognition application. Previously, the analyses of gestures in the computer science were limited. They have been increased with the growth of the science of equipment and the development of sensors devices. With the new high power of processing and large capacity of memories, the straightforward mechanical input devices in the Human-Computer Interaction (HCI) such as keyboards or mice have become insufficient to interact with these processors. This problem becomes clearer with new technologies e.g. virtual reality displays or smart home environment (Pavlovic et al., 1997; Madeo et al., 2016). Additionally, social robots need to recognise what humans are doing and may able to guess why are they doing it to respond correctly (Nehaniv, 2005). Although the long-term attempts in HCI to integrate the automatic speech recognition and their successful implementation (e.g. Rabiner and Juang, 1993), recently the gesture recognition becomes the most interest field in both academic and industrial research (Madeo et al., 2013a) and it turned into a relentlessly vital part of our daily lives. This technique introduces a new model of human-human communication into HCI, which follows the pose/movement of human's hand, arm, or any other part of the human's body depending on the application. Gestures are a non-vocal method used to convey information among people (Pavlovic et al., 1997).

There are a variety of gestures types, and different parts of body implement these gestures; consequently, various gesture recognition techniques are proposed to deal with these categories. Turk (2014) mentions some ways of interacting, while Tran et al. (2012) investigated the gestures that executed with different body parts. Interactions with robots are studied by Salem et al. (2012) and with avatars by Kopp et al. (2003). Baraldi et al. (2008) explored a method to recognise several gestures at the same time, where these gestures may be implemented by a different individual on one or more devices (Madeo

et al., 2016). Before going further into the details of the gesture recognition, we will explain some important terms in details to understand the recognition system efficiently.

2.1 Gesture Definition and Categories

A basic background and definition of the gesture are presented in this section. The concepts in this section are discussed in the view of Human and Computer Interface (HCI), psycholinguistic, and Linguistics; simultaneously to comprehensive, the readers view.

2.1.1 Gesture definition

The gesture is non-vocal information conveyance, commonly used instead of or together with the verbal communication (Kaâniche, 2009). There is a comprehensive range of gesture types and it has been studied in different fields including linguistics, psychology, and HCI. This variety leads to expand the concept of the gesture; consequently, there is no exact definition of the gesture. In the view of the linguistics, McNeill (1992) defines the gesture in his book (Hand and Mind: What Gestures Reveal about Though, page 1) as: "the movements of the hands and arms that we see when people talk". He also wrote (page 2) "gestures are an integral part of the language as much as are words, phrases, and sentences gesture and language are one system". This concept is settled by Xu et al. (2009) when they prove that the same neural system processes the symbolic gestures and the spoken language. Kendon (2004), another pioneer in the linguistic field, defines the gesture in his book (Gesture: Visible action as utterance) as: "a label for acts that have the features of manifest deliberate expressiveness". Thus, not all the gestures are used for interaction. However, the definition of the gesture in the HCI fields considerably different. Pavlovic et al. (1997) described the gesture as: "In a computer controlled environment one wants to use the human hand to perform tasks that mimic both the natural use of the hand as a manipulator and its use in human-machine communication (control of computer/machine functions through gestures)". However, this definition is limited since the gesture approaches not restricted to the hand gestures e.g. in the smartphones, looking down while reading a website or document leads to automatic scrolling upwards. The gesture can be implemented by using particular shape or by using precise movement as will be shown in Section 2.1.2.

2.1.2 Gesture categories

Similar to the definition of the gesture the categories are highly related to the field of the study. From the linguistic point of view, Kendon (2004) identify different types of gestures including:

- **Gesticulation:** is the most daily used gesture along to humanism's conversations;
- **Speech-linked gestures:** replace some words in the conversations i.e. it has own slot within the sentences;
- **Emblems:** are familiar conventionalized signs used instead of the vocal interaction and they are culture-based gestures;
- **Pantomime:** is a series of gestures as dumb show; and
- **Signs:** are lexical words in sign language such as American sign language (ASL).

This taxonomy is known as "Kendon continuum" (McNeill, 1992). While Ottenheimer (2008); Kaâniche (2009) widen the categorization of the gestures into:

1. **Emblems:** for each gesture there is a direct interpretation of spoken words. It replaces the verbal communications and culture-related exactly same as the emblems in (Kendon, 2004).
2. **Illustrators:** correspond to the Gesticulation in the Kendon's taxonomy which is implemented in combination with the speech. A further division for this type into four sub-categories is proposed by McNeill (1992) includes: (a) "iconic" have the same scene of the speech; (b) "metaphoric" present an image of an abstract concept; (c) "deictic" performed by pointing figure toward an object; and (d) "beats" such as quick flick on the hand.
3. **Affect displays:** reflect the emotion or the intention of the communicator e.g. the voice of the communicator may change if he is embarrassed.
4. **Regulators:** is a type of gesture used to control the interaction e.g. turn-taking in the conversation.
5. **Adaptors:** is employed for a personal convenience, which may release the body tension e.g. fast head shaking. This type of gesture is not used for communication and might turn into a habit.

On the other side, Pavlovic et al. (1997) present a taxonomy of the gesture based on (Quek, 1994, 1995) that fits the HCI. First, the movement of the hand may belong to one of two major classes, gesture or unintentional movements. Secondly, the gesture is subdivided to manipulative gesture, act on object e.g. object movement, or communicative gesture, which is the most significant gesture. Thirdly, the communicative gesture further subdivided into acts, related to the motion itself, and symbols gestures, which have some linguistic roles. Finally, each of last two groups is divided into two subgroups. The symbolic gestures are classified as either referential, e.g. move the index finger around a

Table 2.1: The gesture taxonomy concerning the linguistics and HCI, the suggested unified taxonomy at the last column.

Nehaniv	**McNeil**	**Ottenheimer**	**Pavlovic**	**Unified**	
Symbolic Gestures	Iconic	Illustrators	Modalizing	Emblems	**Modalizing**
	Methaphoric	Emblems	Referential		**Referential**
Deixis	Deictic		Deictic	Acts	**Deictic**
			Mimetic		**Mimetic**
Side Effect	Beats	Affect Displays		Beats	
Manipulative		Adaptors	Manipulative	Manipulative	
Interactional		Regulators		Regulators	
			Unintentional Movements	Unintentional Movements	

mean circle wheel or modelling gestures, which is the most used in HCI. It uses some particular pose of the hand to represent specific means. The acts gestures are further divided into mimetic, i.e. imitate action, and deictic i.e. pointing acts.

A new gesture categorization in terms of HCI is proposed by Nehaniv (2005) including:

- **Irrelevant'/Manipulative Gestures** are spontaneous movements along with the motion of the human body but not reflect any meaning e.g. playing with a paper clip,
- **Side Effect of Expressive Behaviour** are casual movements during the communication with others e.g. moving hands with emphasis speech,
- **Symbolic Gestures** are related to emblems gesture and has specifically prescribed interpretation, like nodding,
- **Interaction Gestures** used to control and synchronise the interaction with the partner such as the initialization and the termination of a particular behaviour, and
- **Referential/Pointing Gestures** are related to the deictic gestures in (McNeill, 1992) and also called Deixis, which used to indicate an object while communicating.

It is clear that there is a considerable overlap among all these gesture taxonomies. The most common taxonomies are consolidated in Table 2.1 to simplify the disarrayed classification, and we have proposed a unified taxonomy. The most important types concerning HCI are written in a Bold font in the unified taxonomy Table 2.1.

Among the variety of gestures types, the symbolic/emblems gesture has been mostly used in the HCI especially of the gestures that implemented by hands or arms. These gestures are represented by either single hand or by both hands, and they are performed either by hand pose "***static gesture***", hand movement "***dynamic gesture***" or by both. In this work, we track the emblematic dynamic gestures implemented by a single hand through tracking the hand path for gestures acquired by Microsoft Kinect Sensor.

2.2 Sensors

In order to use the power of the HCI in term of the gesture, the movement and configuration of the hand or any other part in the body that incorporate with the gesture should be provided to the computers by using some technique. Acquiring the gesture has been a target of different types of technology. This technology can be divided into two main types: Contact-based Technology, and Vision-based Technology.

2.2.1 Contact based technology

Contact-based technology may be subdivided into five categories: Mechanical, Inertial, Haptics, Magnetic and Ultrasonic (Kaâniche, 2009). Earliest devices have been a wearable mechanical device like the glove-based devices CyberGloves (Fels et al., 2009) and DataGloves (Quam, 1990). This type of devices was heavy due to the cables connecting the device to the computer and hence it restricted the naturalness of the interacting with the machine (Pavlovic et al., 1997). A wireless connection between this device and the computer is proposed to reduce the weights in CyberGrove II glove-based devices (Fig. 2.1). These devices are used in gesture recognition e.g. (Kevin et al., 2004). A full body suit "IGS-Cobra" is proposed by Synertial company to acquire the whole body gesture (Fig. 2.2).

Inertial devices detect the movements with respect to the ground magnetic fields change. They are available in two types: accelerometers and gyroscopes. The Wii-mote sensors, which belong to the accelerometers type, are used in gesture recognition proposed by Schlömer et al. (2008). Additionally, the most recent smart-watches are equipped with an inertial measurement unit (IMU); whose main components are accelerometers, gyroscopes and magnetometers. These sensors are accurate and position independent. The output of the IMU is a measure of velocity, acceleration or orientation. These measures

Figure 2.1: CyberGlove sensor (CyberGlove Systems LLC, 2017) as an example of contact based technology. ©2017 CyberGlove Systems LLC.

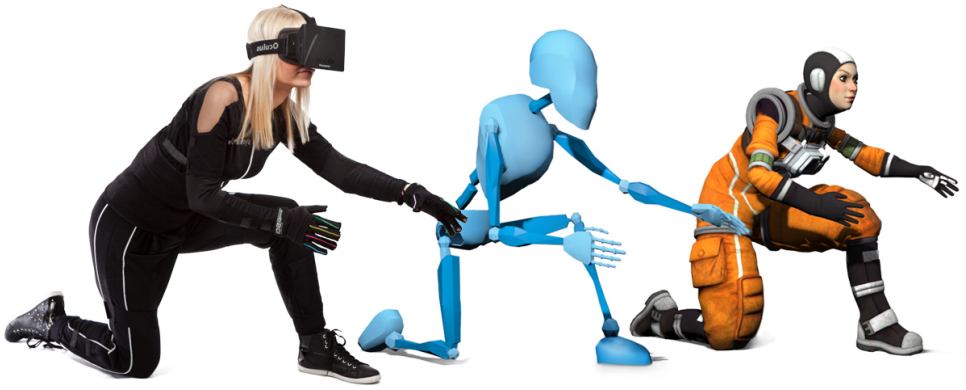


Figure 2.2: Synertial IGS-Cobra (Synertial Labs LTD, 2017), the body suit that contains up to 47 sensors. ©2017 Synertial Labs LTD

can be used directly as features without further processing. Unfortunately, the IMU cannot capture the pose information (Yin et al., 2014b). The public data ChAirGest (Ruffieux et al., 2013) uses the IMU in combination with other sensors.

Recently, the Haptics including the multi-touch screen become essential in our daily life through smartphones, tablets, laptops, etc. Although it becomes sophisticated, it is still limited to one or more fingers in 2D space. The Magnetic devices estimate the motion by measuring the variation of the artificial magnetic field. Finally, there are three types of Ultrasonic sensors: an emitter, discs and multiple sensors. The position is estimated through the time delay of the sonic propagation.

2.2.2 Vision based technology

Initially, some gesture recognition approaches used markers on gloves (without electronics) to interpret the gesture using computer vision on the output of an RGB cameras (Mistry and Maes, 2009; Schröder et al., 2012). Wearing gloves in each gesture acquiring increases the affectation of the interaction in HCI therefore, Shin et al. (2004) and Fothergill et al. (2012) used stereo RGB cameras with bare-hands considering the skin colour. The gesture recognition approaches were augmented to include an IR-sensing (for both far- and near-IR) to overcome the light condition problems in the RGB cameras (e.g. Starner et al., 2000; Toriyama et al., 2016). Different types of cameras are used in the context of the gesture recognition e.g. traditional cameras for a cheap cost, stereo cameras to deduce 3D information (Elmezain et al., 2008), and Pan-Tilt-Zoom (PTZ) cameras to focus on particular details for more accuracy (Bodor et al., 2004). A depth sensing is added to the vision-based approaches to acquire a gesture in 3D space e.g Microsoft Kinect Sensor and Leap Motion Controller. The Microsoft Kinect sensor is used in this work as (Section 4.1),

Figure 2.3: Microsoft's Kinect Sensor for the Xbox gaming system. The image is downloaded from Amos (2017a).

therefore the remind of this section discuss the Microsoft Kinect sensor. Recently, new research object to use the Radar in gesture acquisition e.g. Soli project, which is adopted by Google I/O 2015[1].

The well-known Kinect sensor (Fig. 2.3) is produced by PrimeSence. The first version of it was announced in 2011 by Microsoft for Xbox video gaming while the newer one was released in 2014. It contains an RGB camera, a depth sensor (Infrared sensor and infrared light source), and multi-microphone arrays (Webb and Ashley, 2012). Thus, the information received from the Kinect sensor is an RGB video stream, a depth data stream, a stereo sound and a skeleton data stream. The video resolution in the first version was (640×480 px at 30 Hz) or (1280×960 px at 12 Hz), while the resolution in the new version is (1920×1080 px at 30 Hz). The depth data stream has a resolution of (640×480 px). The sensor range limit is $1.2 - 3.5$ m (extended range $0.7 - 6$ m), and the required area is about $6\,m^2$ (Yin et al., 2014b). Additionally, it streams a full-body skeleton of 20 joints in the first version or 25 joints (including the thumbs and hands tips) in the newer version as seen in Fig. 2.4, which is presented by Microsoft. The error in the depth measurements is increased proportionally with increasing the distance from the Kinect sensor, start with few millimetres till 4 cm in the distance range of $0.5 - 5$ m (Khoshelham, 2011; Yin et al., 2014b).

In contrast to the Kinect sensor, the Leap Motion Controller pay attention to the accuracy within a limited area, particularly the hand region. It can track all ten fingers with an accuracy of $10\,\mu$m (Leap Motion Inc., 2017) through using two IR cameras and three IR LEDs (Fig. 2.5). It can observe the hand motion up to 1 m distance. It optimised to track the fingers and hence it may be considered as a challenge if trying to follow the hand with "fist" pose (Yin et al., 2014b; Wikipedia, 2016).

[1] https://en.wikipedia.org/wiki/Gesture_recognition

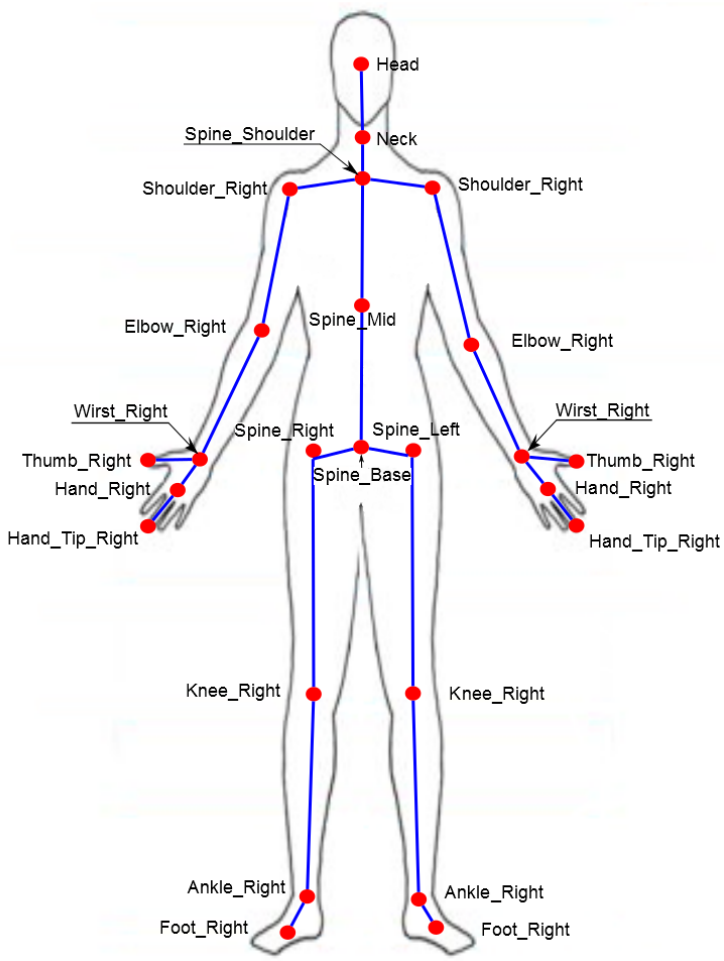


Figure 2.4: Skeleton positions of the Microsoft Kinect relative to the human body. The figure is adapted from Microsoft (2013a).

2.3 Gesture Segments

Speaking differs from the gesticulation by the starting and ending the discourse. There are periods of silence between the words in the speech; this disappearing of the voice may give an indication of the starting and the end of the word. In contrast, when a human wants to start a gesture, for example using hand motion, the hand is available before the gesture and still available after ends it i.e. the system receives a continuous movement. Thus, detecting the start and the end of the discourse is more complicated than in speech.

The manual segmentation leads to several problems including high cost, unnatural interaction with the system, and time-consuming e.g. 10 days were required to segment one video (Quek et al., 2002). In order to solve this problem, a ***rest position*** is proposed

(a) The Construction of Leap Motion Sensor ©2017 Leap Motion Inc.

(b) Using the Leap Motion Sensor to simulate the Hand ©2017 Leap Motion Inc.

Figure 2.5: Leap Motion Sensor (Leap Motion Inc., 2017)

(Kendon, 1980; McNeill, 1992). The discourse should start from the rest position and end with it. Although this assumption simplifies the segmentation process, it limits the naturalisation of the interaction with the machine. Therefore, some research is applied to detect the start and the end of the gesture without the need to use the rest position (Yin et al., 2014b). However, the more difficult challenge in the gesture recognition scope is the ***gesture segmentation***. Following Madeo et al. (2016); Kita et al. (1997), which are based on McNeill (1992); Kendon (1980), the discourse of the gesture is divided into gesture units (G-units). The G-unit represents the movement of the hand or any part of the body that is doing the gesture, between two consequent rest positions. The G-unit is decomposing to gesture phases, which are described by Madeo et al. (2016); Kendon (1980); McNeill (1992) as:

- **Preparation:** moving the hand from the rest position to the position, where the gesture's starting point.
- **Pre-stroke:** brief pause before starting the gesture to find and configure the hand.
- **Stroke** (or nucleus as in (Pavlovic et al., 1997)) the meaningful movement that represents the gesture.
- **Post-stroke:** short pause after ending the gesture with maintaining the hand configuration and position.
- **Retraction:** returning the hand to the rest position.

These phases are illustrated in Fig. 2.6. The G-unit, as proposed by Kita et al. (1997) may have more than one descriptive phrases and some phrases may (or may not) have preparation or retraction phases. Only the stroke phase is compulsory, and all the other aspects are optional i.e. it may be included or not. In addition to the dependent hold (pre-/post- stroke), Kita et al. (1997) mentioned that some independent holds may occur during the stroke phase (Madeo et al., 2016). These variations besides to the similarity between some parts of the phases e.g the rest and holding make the segmentation more complicated. Additionally, the exact instance of the transition from one phase to another depends on the opinion of the specialist who is segmenting the gesture (Madeo et al., 2016). These lead to different decisions of experts about the beginning and end of the stroke as well as the same stroke may be extracted differently by the same specialist if he segments the gesture more than once. Hence, it has become necessary for the researchers to propose a solution for these challenges. In this thesis, we proposed an automatic segmentation, which is explained in details in Section 4.1.

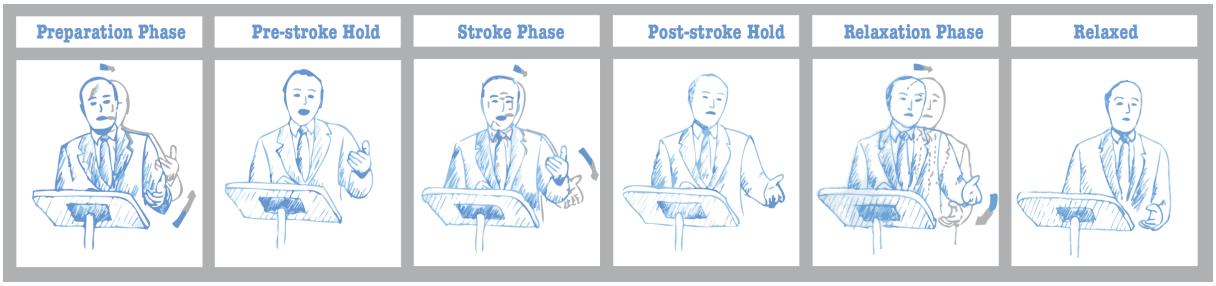


Figure 2.6: Illustration of the gesture phases (Ren, 2017). ©2017 Ada Ren

2.4 Gesture Spotting

Several kinds of literature discussed the gesture spotting e.g. (Alon et al., 2009; Yin et al., 2014b; Bhuyan et al., 2014; Elmezain, 2010). Since Alon et al. (2009) argued the spotting in details, several of their discussed items are covered here. As we mentioned early, the movements of the body's part (e.g. hands) may contain a meaningful gesture, unintentional movements or manipulative gesture. Thus, the most challenging assignment in the gesture recognition approaches is finding the start and the end of the meaningful gesture from the continuous data stream i.e. temporal segmentation. Additionally, the location of the gesturing body's part (e.g. hands) should be determined in all data framing i.e. spatial segmentation. Extracting only the meaningful gesture from the continuous stream is called *spotting*, which requires spatiotemporal segmentation. Some research is presented to solve gesture spotting, but most of them suppose that the gesturing body's part has been precisely localised in each data frame as in (Alon et al., 2005; Lee and Kim, 1999; Morguet and Lang, 1998; Oka, 1998; Yoon et al., 2001; Zhu et al., 2002). Unfortunately, this assumption is subject to several problems since in reality, locating the gesturing body's part is ambiguous in the existence of clutter, other people, skin-colour objects, or background motion. According to Alon et al. (2009), gesture spotting, existing algorithms can generally be categorised into two: direct and indirect approaches. Algorithms that belong to the direct strategies implement the temporal segmentation before starting the recognition while other type intertwined the temporal segmentation with the recognition. If we assume that each gesture recognition algorithm contains two modules, which are the low-level module, where the spatial segmentation is applied, and the high-level module, where the final trajectory should be recognised. Thus, the temporal segmentation is implemented within the low-level module indirect methods and within the high-level module in the indirect methods. The low- and mid-level motion parameters e.g. velocity, acceleration, and orientation are used in the direct approaches like (Kang et al., 2004; Kahol et al., 2004). The foremost constraint of the direct methods is the assumption that the gesture should arise between two non-gesturing intervals. The boundaries of the gesture in the indirect approaches are found by matching the input sequence with the

known gesture classes and select the input interval that gives a good recognition score as in Alon et al. (2005); Lee and Kim (1999); Oka (1998). Commonly, the indirect methods of extracting the gesture are based on one of the Dynamic Programming (DP) extensions such: Hidden Markov Models (HMMs) (Lee and Kim, 1999; Wilson and Bobick, 1999; Vogler and Metaxas, 1999; Brand et al., 1997; Starner et al., 1998; Stefanov et al., 2005; Chen et al., 2003), Dynamic Time Warping (DTW) (Darrell et al., 1996; Kruskall and Liberman, 1983), Continuous Time Programming (CTP) (Oka, 1998), and Conditional Random Fields (CRF) (Quattoni et al., 2007; Lafferty et al., 2001). Those methods find the end point of the gesture by comparing the recognition likelihood to a fixed or an adaptive threshold. Using a fixed threshold is non-practical assumption to deal with the inconstant data stream's likelihoods. Hence, several kinds of research are proposed to estimate an adaptive threshold by using a non-gesture model, for example, Lee and Kim (1999); Yang et al. (2006) used HMMs model, and Yang et al. (2009a) used CRFs. The primary constraint of the indirect methods is that first detect the end point of the gesture then they use the backwards spotting to track the starting point of the gesture. This process leads to a time delay between the segmentation and the recognition modules. Additionally, they need more time of processing and more complicated techniques e.g. an optimisation process. Regularly, the highly accurate spotting often leads to more complexity, thus less reality.

Some other types of research mixed the direct and indirect methods by candidate more than one frontier in the low-level module, and the most accurate one is selected in the high-level module e.g. Alon et al. (2009).

2.5 Hand Tracking and Feature Extraction

The first module in the gesture spotting is the spatial segmentation which is the locating of the body's part of the actor in the streamed images. In the case of hand gesture, this process is called hand tracking, and it is needed particularly in vision-based sensors. The hand tracking differs from application to another. For example, the hand shape is important in the applications that are used to recognise static (hand pose) gestures, therefore, the whole hand is extracted from the rest of the image by using one of some methods. The well-known method uses the skin colour to segment and tracks the hand (Yin et al., 2014b; Dadgostar et al., 2005). The applications that employ the dynamic gesture effort to estimate the trajectory of the hand rather than the shape of the hand (Cutler and Turk, 1998). These applications may look to the hand as a blob, and they estimate the 3D axis of the hand's centre only in each of the successive streamed image. Consequently, the trajectory of the hand is tracked only. In our work, the 3D axis of the hand is obtained directly by the Kinect sensor.

In the case of pose gestures, the main two types of features are hand contour and hand texture features. Frequently, the Fourier transform is used with the contour features

(Yao, 2014) while the texture features have different representations where the Histogram of Gradients (HoG) (Dalal and Triggs, 2005) and the Scale Invariant Feature Transform (SIFT) (Lowe, 2004) are the best-known methods. In contrast to the hand pose recognition approach, the approaches that track the dynamic hand trajectories have a limited number of features, which can be divided into two types: local and global features (Yao, 2014). Hand orientation, speed, movement direction, acceleration, and location of the Cartesian and Polar axis, are an example of local features (Yoon et al., 2001; Elmezain et al., 2009). While the pattern shape features extracted from the complete hand trajectory, represent the global features.

2.6 Gesture Recognition Approaches

As mentioned earlier, there are several types of acquiring the gesture. Consequently, the gesture recognition algorithms are diverse according to the acquisition methods, the body part that makes the gesture, and the type of the tracked gesture. We will cover the hand gesture approaches in this section since the most dominant gestures in the HCI topic are implemented by hand(s). Additionally, most of contact-based sensor's outputs are used directly as features without further processing; hence we will focus on the vision-based methods. An enormous amount of studies target vision-based methods because they help to interact naturally with the machines with bare hands or without any additional wearable device. Furthermore, the dynamic hand gesture is the main goal of our study. Since static gestures have been covered in a large number of studies, some topics in this section will deal with it. Previously, the hand gesture algorithms were divided into two main types: model-based algorithms and appearance-based algorithms(Pavlovic et al., 1997). Recently, new techniques are proposed to solve the hand gesture problems. Hence, we follow the categories proposed by Abid (2015) including:

- 3D Model-based Approaches,
- Appearance-based Approaches,
- Machine Learning Algorithm Based Approaches,
- Rule-based Approaches,
- Syntactic Approaches, and
- Local Feature Approaches.

These categories can be summarised in the Fig. 2.7.

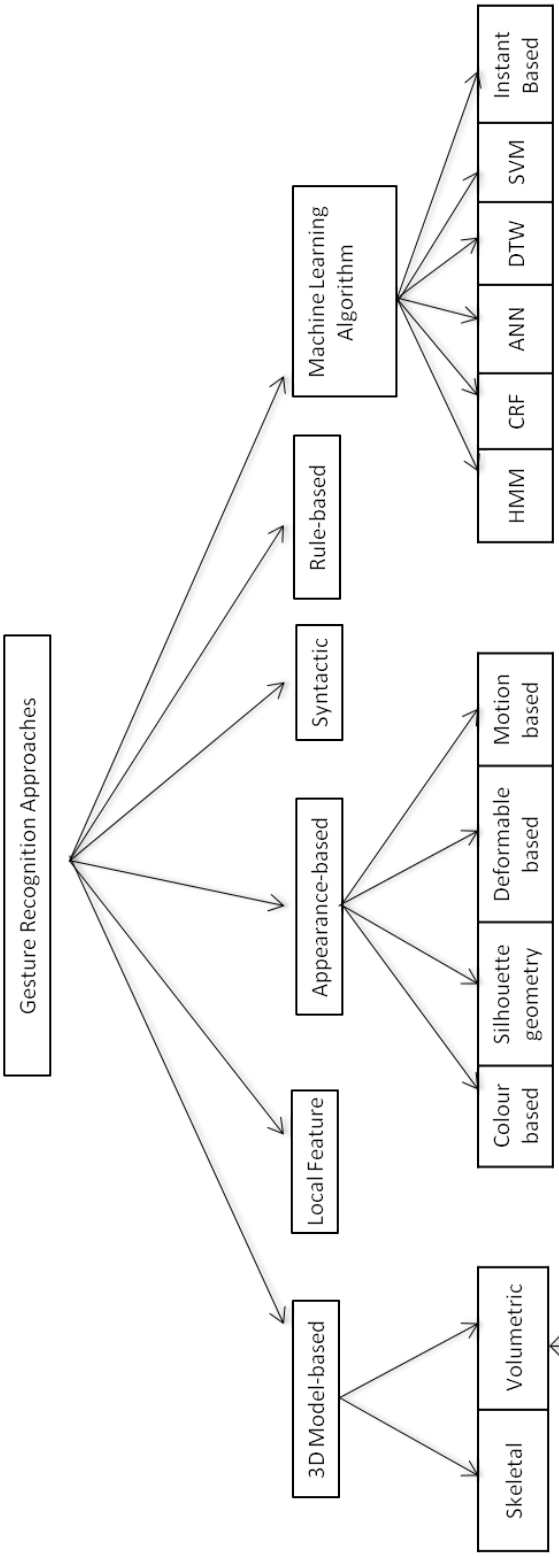


Figure 2.7: Schematic diagram represent the well-known methods of the gesture recognition approaches

2.6.1 3D model-based approaches

The 3D hand/arm models can be further subdivided to *volumetric* and *skeletal* models. Volumetric models have been mainly used in computer animation industry (Thalmann and Thalmann, 1990) and then used in computer vision applications (e.g. Koch, 1993). They are used to represent the 3D visual appearance of the human hand/arm. The 3D hand model's parameters are estimated by minimising the error between the 2D input images of the hand with the 2D projection of the proposed 3D model. Complex 3D surfaces such as NURBS (non-uniform rational B-splines), generalized cylinder, and Super-quadric (Pavlovic et al., 1997) are mostly used by the volumetric models in computer animation (Fig. 2.8). NURBS models provide a rich description to the hand, which allowed recognising a wide range of static hand gestures. Although they become realistic, they are computationally intensive, and they are too complicated to use in the online recognition. Additionally, they require a large database to train on the entire characteristic shapes for different views (Abid, 2015). The other types i.e. generalized cylinder and super-quadric, are used to simplify the body part rendering (e.g. Murugappan et al., 2013). Even though this simplification, the estimation of the model's parameters is still very complex and the dimension of the parameters is very high (Pavlovic et al., 1997). In order to overcome these problems, the skeletal-based models are used instead of the volumetric models. In these models, only the joint angles and the segments lengths are used to represent the hand or the arm. These models are discussed in details in Thompson (1981). In the recent researches, the skeletal details of the whole body can be directly gotten from the Microsoft Kinect sensor. In this work, we have used the Kinect sensors to acquire the skeletal of the human. Unfortunately, the most recent version of the Kinect sensor represents the hand by only three joints including the centre of the hand, the tip of the index and thumb fingers. In the proposed work the centre of the hand has been quite sufficient.

2.6.2 Appearance based approaches

This type of approaches derives the parameters directly from the images, without needs to construct a spatial model to the body. Pre-define templates of the hand or any other body's part are used for this purpose. Following Kaâniche (2009) and Pavlovic et al. (1997), different models belonging to this type of approaches include:

- **Colour based models**: body markers are used in this model to track the motion of the body. A multi-scale colour feature together with particle filtering may be used in this methods in a hierarchal way (e.g. Bretzner et al., 2002).
- **Silhouette geometry based models**: the silhouette geometric properties, e.g. perimeter, surface, convexity, rectangularity, bounding box/ellipse, orientation, elongation and centroid are used in this methods (e.g. Birdal and Hassanpour, 2008).

- **Deformable based models**: deformable 2D templates are represented by a set of points located on the outline of the hand, or any body's part. They are used to approximate the object outline by considering the points as interpolation nodes (Pavlovic et al., 1997). These templates consist of two types of parameters, which are the internal parameters and external deformation. The internal parameters are represented by the average point sets, which describe the average shape of a specific group of shapes, and the point variability parameters, which describe the possible variation within a certain group of shapes. For example, all the open hand posture belongs to the same group specified by the same shape on average. Any possible posture of an open hand may slightly vary from the average. The external deformations describe the overall motion of the template e.g. rotation and translation. This method is used in a wide range of applications, for example, (Cipolla and Hollinghurst, 1996; Cootes et al., 1995; Ju et al., 1996; Ramani et al., 2016).

- **Motion based models**: 2D image sequences are used as gesture templates. Each of these image sequences models one type of the used gesture types. The parameters that used in these methods are the images themselves or features derived from them. Either the complete hand image sequence (e.g. Darrell et al., 1996) or the fingers image sequence (e.g. Crowley et al., 1995) can be used as templates depending on the application. Another type of approaches called *motion history images* (MHIs) (Bobick and Davis, 1996), which are 2D images represent the visual image over a temporal window. The intensity of each pixel in this image is related to the period of the motion sustained at this pixel (Pavlovic et al., 1997). Some other applications, which concern about the deictic gesture, used a single fingertip and a reference point only. These approaches assume the location of the fingertips relative to the palm is different in most of gestures (e.g. Fukumoto et al., 1994; Vogel and Balakrishnan, 2005).

2.6.3 Machine learning algorithm based approaches

Learning the machine means giving it the ability to discriminate among classes that it had been previously trained on. Commonly, the accuracy of the learning algorithms is increased with increasing the training data. This section gives a short overview of machine learning algorithms for gesture recognition. Using the machine learning in the pattern recognition is described in details in Chapter 3. Some of the massive machine learning algorithms are used in the gesture recognition field. Among these algorithms, the Hidden Markov Model (HMM) and also the Conditional Random Fields (CRF), the Artificial Neural Network (ANN), Dynamic Time Warping (DTW), Support Vector Machine (SVM) and instant based approaches are dominant in the gesture recognition. A brief description

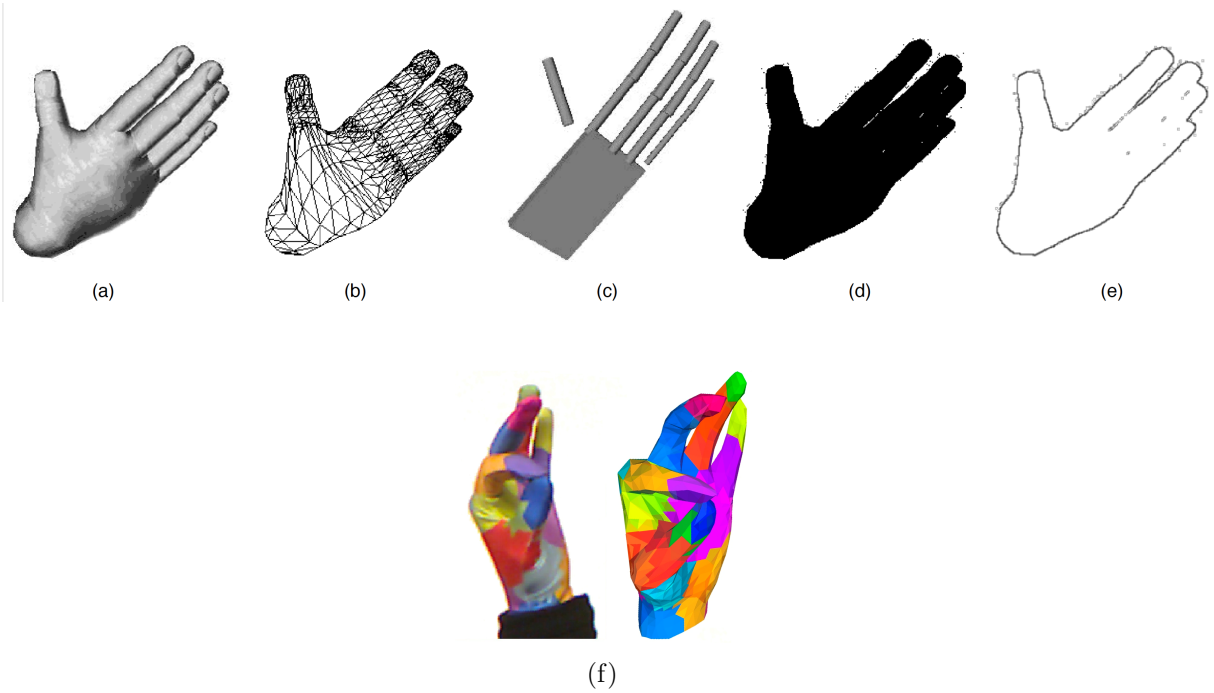


Figure 2.8: Hand models. Different hand models can be used to represent the same hand posture. (a) 3D Textured volumetric model. (b) 3D wireframe volumetric model. (c) 3D skeletal model. (d) Binary silhouette. (e) Contour (f) Colour. The sub-figures (a)–(e) are adopted from Pavlovic et al. (1997) (©1997 IEEE) and the sub-figure (f) is adopted from Schröder et al. (2012) (©2012 IEEE).

of these approaches is presented in this section in separate paragraphs. Despite the advantage of the machine learning in the gesture recognition e.g. high accuracy, flexible, fast, etc., they are suffering from two major problems: feature selection and training dataset (Elmezain, 2010). Using a large number of features raises the dimensionality, which may affect the speed of the recognition and in some cases, affects the accuracy. In contrast, a significant reduction in the number of the features leads to poor recognition accuracy. Hence, finding only the useful features is crucial. On the other side, training the algorithm with a sufficient number of the training samples is vital to work properly. Unfortunately, in this thesis, we work on data streaming, which means scarcity in the training data with abundant unlabelled data. This is the main problem that we try to solve in this thesis.

Hidden Markov model

Hidden Markov Model (HMM) is widely used to solve the Spatio-temporal problems (Elmezain, 2010). Initially, the HMM was used in terms of speech recognition (e.g. Rabiner, 1989) then it extended to the gesture recognition (e.g. Starner et al., 1998; Gao et al., 2004; Elmezain, 2010). It has been used in about 29% of all gesture recognition studies (Madeo et al., 2013b). The segmentation phase and the recognition phase can be optimised simultaneously (e.g. Yin et al., 2014b), which is the main feature of the HMM. Commonly, HMM used to recognize pre-defined gestures (e.g. Chen et al., 2011). An extension of the HMM has been used to discriminate more gesture classes (e.g. Kettebekov et al., 2005). Harper and Shriberg (2004) proposed three different models and combined their outputs in one HMM model in order to achieve a final decision. Wilson and Bobick (2000) used the HMM to estimate the temporal model structure only, while training the HMM model is implemented at the beginning of online phase rather than offline phase. Corradini (2001) used the Neural Networks to compute the transition probabilities for the HMM in a hybrid model.

Conditional random fields

Conditional Random Fields (CRFs) are a undirected graphical model, which unlike HMMs, that model the joint probability $p(x, y)$, CRFs model the probability $p(x|y)$ of a label sequence $\mathcal{Y}$ given the input sequence $\mathcal{X}$ (Vail et al., 2007). This gives CRFs the ability to consolidate complex input features and maintaining the independence assumption. The linear-chain CRF is well-known in the language processing (e.g. Roark et al., 2004). Since, the gesture is similar to the language, CRFs presented in term of gesture recognition (e.g. Yang et al., 2009a; Vail et al., 2007).

The CRFs are augmented by adding hidden states to present the Hidden Conditional Random Fields (HCRFs). The HCRFs can represent the complex dependencies of the training data or their implicit structure (Chang et al., 2009b). It is used in the gesture recognition in several approaches, for example, (Wang et al., 2006; Quattoni et al., 2007).

Artificial neural networks

The Artificial Neural Networks (ANN) simulate the biological neural network in the brain of humans or animals. Normally, ANNs have been used in the machine learning to estimate non-linear functions, which may depend on multiple inputs. The nodes in the ANN are the fundamental units, which may act similar to the neurons in the brain. The neural networks consist at least of 3 layers; the input layer, where the input should apply to the ANN; the hidden layer(s), which contain the nodes (neurons); and the output layer. By increasing the number of hidden layers (as in deep learning) it becomes able to approximate very complex functions but at the expense of the complex training. The nodes are fully connected via weighted links. These weights are estimated during

the training phase. Additionally, each node has an activation function e.g. radial basis, sigmoid, etc. The activation function's parameters are also estimated during the training phase. Mainly, there are two structures of ANN, which are the recurrent neural network and a multi-layer feed forward neural network. Although the recurrent structure may model more complex systems than the feed-forward structure, its training is very complex and it may behave chaotically. An example of using the neural network in the gesture recognition is the Gesture Driven Interface in Virtual Environments (GIVEN) proposed by Vaananen and Bohm (1993). They used two models of ANN; the first one is for static gesture recognition and the second model is for dynamic gesture recognition (Elmezain, 2010). A real-time gesture to speech approach is implemented by Fels (1994) based on a neural network. Another application of using neural networks in the gesture recognition is proposed by Kjeldsen and Kender (1995) to control a window-based interface system. The system is based on two layers; in the first layer, the hand is spotted in real time and the second layer (or the action layer) uses a grammar to map the detected images by the first layer with the required action. Regarding stereo cameras, Stiefelhagen et al. (2004) used two neural networks to process the head's disparity and intensity in a combined grey and depth data. The first network is used for the tilt in the head's orientation and the second for panning.

Template matching

Before comparing a gesture with the template, it should be normalised. The main technique of the normalisation is the dynamic programming. The dynamic programming includes Dynamic Time Warping (DTW), Continuous Dynamic Programming (CDP) and Dynamic Time Alignment (DTA). DTW algorithm is used for the normalisation in many applications, particularly the speech recognition (Rabiner and Juang, 1993; Bhuyan et al., 2008). It can measure the similarity between two sequences even they differ in time, speed, acceleration and length. We may get an inaccurate result if we used the Euclidean distance to compute the similarity between two out of scale gestures. Normalising the gestures in the Spatio-temporal space is very important to get a realistic result when comparing two gestures because it is very difficult to perform two identical gestures in length and dimensions even if the person insist. Even if we assume the gestures are identically performed, the result of comparing them using direct Euclidean distance will be not accurate if there is a time shift between the two gestures. Initially, the DTW was used with 1D sequences especially, audio signals but then it was extended to work with multi-dimensional data (Gillian, 2011). In this work, it is modified to fit 2D data, which represent the location of the hand in 2D space (cf. Section 4.1.1). The CDP is also used in several approaches to segment and classify the gestures. It is successfully applied in different approaches (e.g. Takahashi et al., 1992; Li and Greenspan, 2007; Seki et al., 1993).

Support vector machine (SVM)

Support Vector Machine (SVM) is a technique that maps two distributions, non-linearly separated, from the original features space to a high-dimensional space to find the optimum hyperplane that separates these two distributions. The optimised hyper-plane is then used to perform a classification or regression for the new samples (Vapnik, 1998; Madeo et al., 2013a). Originally, SVM was announced by Vapnik (1963) and then developed by Boser et al. (1992). It can be simplified by the following example (Madeo et al., 2013a), suppose N is the number of samples in the training set. The samples are in the form $\{\vec{x}_i, y_i\}_{i=1}^{N}$, where the $\vec{x}_i$ is the feature vector and the $y_i \in \{-1, +1\}$ is the label of the $\vec{x}_i$. Then the goal of the SVM is finding a hyperplane that separate the two classes optimally using $f_{svm}(\vec{x}_i) = \vec{w}_{\text{svm}}{}^T \vec{\varphi}(\vec{x}_i) + b'$. The $\vec{w}_{\text{svm}}$ here represents the optimal weights, b' is the optimal bias. The input feature vectors are mapped to the high-dimensional space using the non-linear mapping $\vec{\varphi}$. The hyperplane is optimised by maximising the distance between hyperplane and the nearest data from each class in the training phase, which leads to computing the optimal weights w_i and optimal bias b'.

Initially, the SVM was a binary classifier i.e if the output $\hat{y} > 0$ then the sample belongs to first class else it belongs to the second class. Solving the optimisation of the hyperplanes for multi-classes together is more complicated in comparison to solving it into several binary classes. This problem is solved by looking at the multi-classes problem as several binary problems, Zhang et al. (e.g. 2006). The multi-classes problem decomposed to either one-vs-all or one-vs-one binary classifiers. In one-vs-all taking each class and consider all other classes as one class and find an SVM model for this class. This process is repeated for all classes and the output of each new sample is computed for each model. Hence, if the number of the classes is N_{class} then the number of models in this approaches will equal to N_{class}. While one-vs-one is taken each possible combination of two classes and compute the hyperplane. This method gives a number of $N_{\text{class}}(N_{\text{class}} - 1)/2$ models. The final output is obtained by different methods e.g. voting by majority. The SVM classifier is used in the gesture segmentation and recognition in some approaches (e.g. Madeo et al., 2016; Ramakrishnan, 2011; Ramakrishnan and Neff, 2013).

In the systems that need to increment the number of classes without train new classifiers from the start as we supposed in this thesis, the one-vs-one configuration needs to add several new SVM models for each new class. Hence, we adopt the one-vs-all configuration in Section 5.4.

Instance/Distance based approaches

In this type of approaches, the training data samples are explicitly used to classify the new samples. Literally, there is no real training process and no any prior pieces of information otherwise the training data. It assumes that the samples which have the same label should lie close each other in the Euclidean space (Dasarathy, 1991). The clearest example of

this approaches the metric learning and particularly, the K-nearest neighbour algorithm (KNN) (Altman, 1992). The KNN algorithm classifies the new sample by finding the labels of the k-nearest training samples and make a majority voting. Although these approaches are simple and have acceptable accuracy in the gesture recognition topic (Ali and Shah, 2010), they need to store all the old samples, which it is difficult especially when training data are streamed continuously as in our case.

Other approaches

There are extensive studies in the field of machine learning related to gesture recognition. We explain the most used in the field of the gesture recognition, but the studies are not limited to these methods. Spano et al. (2012) used a Petri Nets based meta-model to implement a compositional model gesture definition. While, Choi et al. (2008) apply the k-means. Some other techniques used the probabilistic methods (e.g. Eisenstein et al., 2008) used Bayesian analysis and (Wong and Cipolla, 2006) used Relevance Vector Machine .

2.6.4 Rule-based approaches

Manual rules are encoded in this type of approaches, and the gesture's label will provide the output if the input features justify the corresponding rule (Abid, 2015). A set of rules is proposed by Cutler and Turk (1998) to classify human actions. Su (2000) predicts the hand's motion that is related to low-level features. Trajectories of the motions are analysed and predicted in Hassan et al. (2010) using the rule-based technique. Whereas Chang et al. (2009a) suggest a method which accepts the variation in the input gesture if it is performed by different people. They based on fuzzy rules to track and recognise the posture and action.

2.6.5 Syntactic approaches

Syntactic pattern recognition is also called structural pattern recognition. Sometimes the simple features are not enough to represent complex structural objects in the recognition system. Instead, the Syntactic approaches represent them by a set of elementary parts called Primitives (Abid, 2015). The primitives are a variable-cardinality set of symbolic and nominal features[2]. The elementary part should satisfy some rules to be identified as Primitive (see Sonka et al., 2014). Thus, Syntactic approaches consider the interrelationships among the attributes, which leads to the ability to represent more complex structures than normal pattern recognition. Additionally, a set of grammar should be identified to construct the different patterns or actions from the described primitives. This type of pattern recognition is used to recognise the hand gesture or the human activity (e.g. Hand et al., 1994; Derpanis et al., 2004; Ryoo and Aggarwal, 2006).

[2]https://en.wikipedia.org/wiki/Syntactic_pattern_recognition

2.6.6 Local feature approaches

Local image features have been used to provide a compact representation of the pattern in image (Laptev, 2005). These features are also called "interest points" (Laptev, 2005), which are a set of independent local regions within the image. The spatial interest points are extended to Spatio-temporal features for a compact representation of video or space-time events (Abid, 2015). The shape and motion characteristics are captured by Local space-time features and then provide a representation of the events. The events representation is independent of their spatial-temporal shifts and scales. Additionally, it is independent of the background or to the multiple motions in the scene. motion segmentation and tracking are not required to capture these features since they are extracted directly from the video (Wang et al., 2009). Several approaches used different tools to implement the local feature on the gesture or action recognition (e.g. Laptev, 2005; Abid, 2015; Sanin et al., 2013).

2.7 On-line Recognition Algorithms

In some applications, the recognition and sometimes the training should be implemented in real-time e.g. robotics. Since the gestures streamed continuously in the real-time mostly by humans, the receiver e.g. Robot should reply in the same speed, and any delay may cause to miss some information, which may incur a high-risk (Feng et al., 2011). For example, a double tap gesture used in some application to add a musical note, if the application is not fast enough to acquire both taps then the gesture will be translated into another meaning. Some other applications need to update/retrain the classifier by new training samples in real-time. The training or updating the classifier in real-time is called real-time learning, or online learning. Updating the classifier by using only the new data and some parameters representing the old data instead of saving all the old samples is called incremental learning. Song et al. (2012) used the Latent-Dynamic Conditional Random Field (LDCRF) with temporal window to achieve online segmentation and recognition simultaneously. The hierarchical HMMs are used for online musical gesture recognition in (Gillian, 2011). Yin et al. (2014b) also used HMMs to implement online recognition for the static and dynamic gesture in a single framework.

2.8 Gesture Application

Gesture recognition has a wide range of applications and it has been employed in several domains of our life. Some of the recent and important applications, particularly vision-based gesture, are discussed here.

2.8.1 Virtual reality

Simulated environments are produced by a computer technology to enable the humans to manipulate objects and interact virtually. The Virtual Reality (VR) uses gestures in the interactions to obtain more realistic environment. VR is widely used in the Gaming applications through 3D monitors or head-mounted display (Fig. 2.9).It is also used in the training systems, for example, the pilot simulation. Berry (1998) used the gesture to navigate, select, and move the objects in a virtual battlefield environment (Abid, 2015). Roudavski et al. (2010) used the virtual reality for transport and urban design projects[3].

2.8.2 Sign Language Recognition

The sign language recognition aims to interpret the hand and head gesture into text or speech. Developing this application is very important for deaf people to enable them to communicate with other people or machines. Both static and dynamic gestures may be used in the sign language, which is mainly implemented by hand or head. A sentence-level continuous American Sign Language (ASL) is recognised in real-time based on HMM system by Starner et al. (1998). The neural network also used in the recognising the Persian and Japanese languages in (Karami et al., 2011; Murakami and Taguchi, 1991), respectively.

2.8.3 Human Machine/Robot Interaction

Several applications provide interactions between human and machine without using the traditional input devices such as mice and keyboards. An example of devices that use gesture in the interaction with the human is smartphones, tablet, laptop, or any other touchscreen devices, which may be available at shops or public services. The smart environments or smart houses are another application of the gesture recognition regarding the human-machine interaction e.g. control the light intensity or navigate the channels on the TV by using the gesture.

Many approaches have been implemented to provide communication between the human and the robots through using hand and arm gestures (Malima et al., 2006; Ghobadi et al., 2008). The social robots, which socially interact with a human, are currently growing. This type of robots can provide a kind of care to the children, elderly, or atypical human (Dautenhahn, 2007). Thus, they are necessary to understand the human gesture and interact according to it. Additionally, they designed to communicate with each other, communicate with the environment and behave according to traditional social and cultural norms (Li et al., 2011). These robots have been expected to learn and reproduce the human body movements. The movements or gestures that are companying the speech have been shown they induce the meaning of the social interaction. Consequently,

[3]https://en.wikipedia.org/wiki/Virtual_reality#cite_note58

Figure 2.9: Head Mounted Display. The image is downloaded from Amos (2017b).

they have become the main feature in the human-robot interaction design (Fong et al., 2003; Cabibihan et al., 2012). "Hawk" (Fig. 2.10) is the first commercial generation of the humanoid robots introduced by Dr Robot Inc.(Abid, 2015). Some other robots can follow the human instructions or can classify their emotions (Sato et al., 2007). Another generation of the robots can learn by human demonstration through different types of machine learning (Yorita and Kubota, 2011). Recently, the robot "Pepper" produced in Japan by "Aldebaran Robotics" and "SoftBank Mobile" is well-known as the newest emotional robot (Fig. 2.11). It is designed to be similar to persons with 4 feet tall and has pretty good ability to recognise the facial expressions such as sadness, hostility, and also recognise the facial expression within the voice (Pepper, 2017).

In this thesis, Some specific emblematic arm gestures are adopted, which are normally used as instructions to robots Section 4.1.1.

2.8.4 Medical Systems and Healthcare Technology

The computerised devices have gradually become dominant in the hospitals. Using these devices in the hospitals has encouraged the use of new methods to interact easily with the devices and prevent spreading infection caused by the conventional methods like keyboards and mice. This interaction can be implemented by using the natural interaction with the computer including the gesture and speech recognition (Wachs et al., 2006). Some research is proposed in this context Graetzel et al. (e.g. 2004) proposed several methods

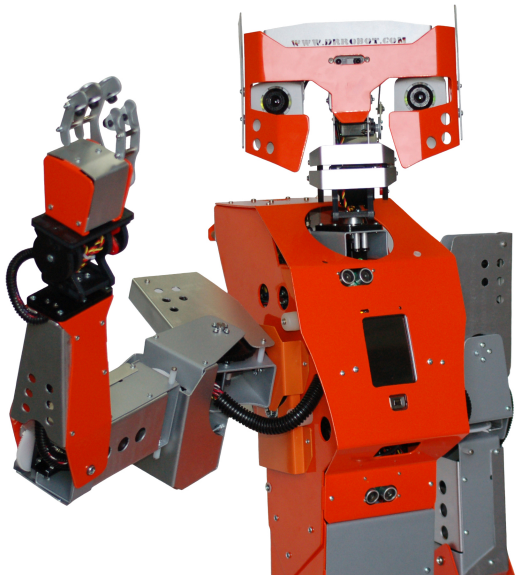


Figure 2.10: Hawk, the earliest social robot produced by "Dr. Robot Inc" (©2001-2017 Dr. Robot Inc.).

Figure 2.11: Pepper, the newest emotional robot produced by "Aldebaran Robotics" and "SoftBank Mobile". This image is adopted from Tanaka et al. (2015).

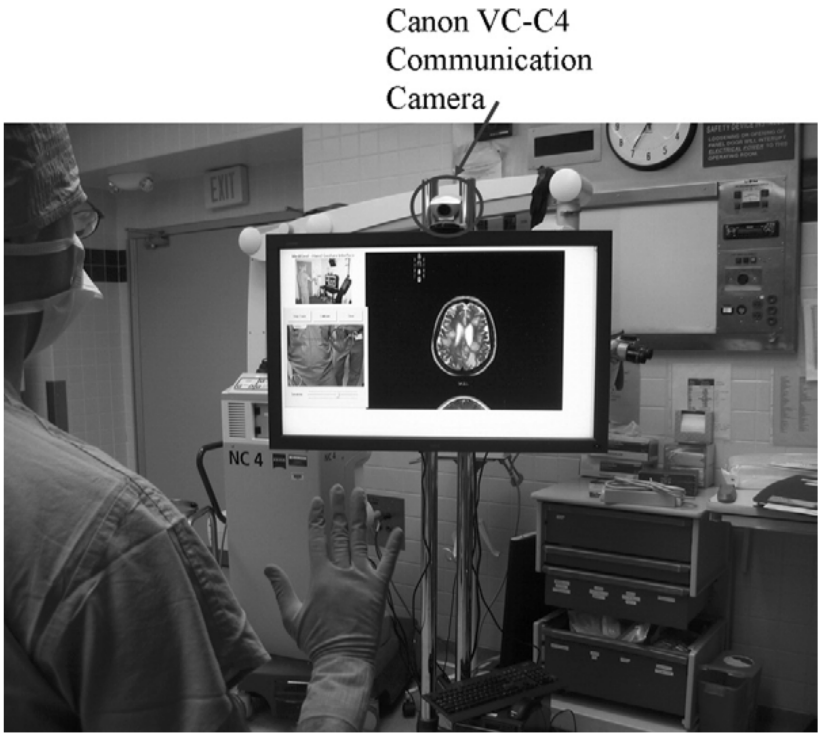


Figure 2.12: Hand gesture interface for medical visualization applications (Gestix). The image is adopted from Wachs et al. (2008). ©2008 IEEE

to integrate the gesture with the medical devices and the virtual mouse is one of these methods. While Wachs et al. (2011, 2008) present some methods to navigate and explore the Magnetic Resonance Imaging (MRI) images via gestures (Fig. 2.12).

Although the medical instruments normally are used by the doctors and nurses, they help the disabled patients to feedback and user adaptability as a part of their rehabilitation therapy (Abid, 2015). For example, the wheelchairs in (Kuno et al., 2000) are provided by gesture command recognition. Chen et al. (2009) present a smart environment based on gesture and facial expression as a healthcare environment.

2.8.5 Presentations/Gesture-to-Speech

The gesture may be used in the presentation of the weather narration or technical reports to pinpoint what the presenter is talking about (Ju et al., 1997). The pointing gesture is mostly used in this type of applications. Additionally, the interpretation would be very helpful if implemented simultaneously with the presenter speech (Kaâniche, 2009). Kettebekov et al. (2005) proposed an algorithm to analyse the synchronisation of the

gesture and the speech during the presentation. While Yin et al. (2014b) submitted HMM models of hand gesture to control and navigate the slides during the presentation.

2.8.6 Video Surveillance

Surveillance cameras and Closed-Circuit Television (CCTV) have been increasingly grown and used for surveillance the public places since the September 11'th attack (Niu et al., 2003). The safety and the security are the main targets for the video surveillance (Kaâniche, 2009). While the number of the videos have been increased, an automatic method of detecting the abnormal actions e.g. violent and furtive actions are necessary to achieve. Hiroshi et al. (2006) submitted an approach to observe elevators for any violent actions e.g. bag-snatching. The video surveillance is used for safety purpose in the hazard areas in (Chang and Huang, 2004). A large study is proposed by Carnegie Mellon University on the video surveillance in (Collins et al., 2000).

3

Related work: Semi-Supervised Learning and Data Stream

The gestures of the same class cannot be performed in the same manner by different persons. Also, the same individual may implement the same gesture class in a different manner if he implements it more than one time. Therefore, in principle, the classifier should be trained with all possible gestures performed in all possible manners to get an acceptable recognition rate. Unfortunately, the manually labelled data are costly, and it is impossible to get labels for all possible gestures. Hence, we cannot train the classifier in fully-supervised learning. In contrast, the unlabelled data are streamed continuously, which has made us able to use the semi-supervised learning to solve this problem. That means the classifier is initially trained on a labelled data set in a supervised manner; then the training set is updated using the labels assigned by the classifier (Zhu and Goldberg, 2009). However, several problems arose out of using streamed data, e.g. the "infinite length" and the "concept-drift". Many online algorithms (Masud et al., 2011) addresses these issues. Besides, there are some possible problems such non-linearly separable distributions of the data, the emergence of new classes or outliers, and the computational complexity.

These problems can be overcome by using incremental learning and an outlier detection. Incremental learning is used to discard all old data and update the classifier parameters with the new data only. So, the continuous update of the classifier ensures that it follows the concept drift without needing to use the whole data of the training set. An outlier detection is adopted to reject all outliers, since the false labels assigned by the classifier potentially affect the performance of the classifier after the next training cycle.

This chapter has been adapted and/or adopted from: (Al-Behadili et al., 2015a,b,c,d,e,f, 2016a,b,c,d,e)

3.1 Machine Learning

The term "Machine Learning" was initially used by Arthur Samuel in 1959. He defined it as "*the field of study that gives computers the ability to learn without being explicitly programmed*". Mitchell (1997) broadened the definition of the machine learning to cover most types of learning algorithms. He stated that:

"*a computer program is said to learn from experience E, with respect to some task T, and some performance measure P, if its performance on T as measured by P improves with experience E.*"

To better understand this definition, we consider an email spam filter example. When the user marks some of the emails as spam, the email program learns which type of emails is mostly undesired (corresponds to experience E) and uses this information to improve the spam filter automatically (corresponds to the task T). The performance measure P is the accuracy of the filter (Gutierrez, 2015).

The term "Machine Learning" in the computer science community is highly related to the term "Pattern Recognition" in the engineering community. They are two faces of the same field of studies, which are related to automated data analysis and trying to classify them into classes (Theodoridis and Koutroumbas, 2009; Bishop et al., 2006; Barber, 2013). The machine learning has made a revolution in the artificial intelligence and currently, it is considered the most interesting field in the computer science and the software engineering. It has been integrated into a wide range of applications including computer vision, speech recognition, dimensionality reduction, biomedical applications, smart devices, etc.

Barber (2013) broadly divided machine learning into *supervised learning* and *unsupervised learning*. In supervised learning, all the labels of the *training data*, which form the set of the data that is used to train the algorithm, and the training data itself are assumed to be available at the beginning of the learning process. The aim of the supervised learning is to get a highly accurate classification. In contrast, the training data are assumed to be available without labels in the unsupervised learning. The objective of unsupervised learning is to group or cluster the data due to its similarities. Unsupervised learning methods are also called *clustering algorithms* (Theodoridis and Koutroumbas, 2009).

Many traditional machines learning algorithms are trained only on manually assigned data, where all data should be available at the beginning of the training. However, these two conditions of such algorithms are hard to fulfil, since most real-life data such as gesture, are streamed continuously as unlabelled data. The manually labelled data are costly, time-consuming, and hard to get since they require experienced annotators. Additionally, streamed data are often affected by drift, outliers, the emergence of new classes and disappearance of some other classes. Similarly, it is tough to use unsupervised learning, since the data is streamed continuously and not entirely available at the beginning of the learning process. A third technique of machine learning called ***semi-supervised***

learning is located in between the supervised and the unsupervised learning. It uses labelled and unlabelled data to extract the pattern formula. There is a broad range of semi-supervised techniques, and some of them can be utilised in the online classification, which fit the data stream. For a better understanding, the semi-supervised techniques are presented in more details in 3.4. Additionally, a brief explanations of the supervised and unsupervised learning are discussed in Section 3.2 and Section 3.3, respectively.

3.2 Supervised Learning

Each pattern should be represented by feature vectors $\vec{x}_i$, for example the hand gesture may be represented by the hand position, velocity, orientation, etc. In supervised learning, along with each pattern $\vec{x}_i$, e.g. gesture, there is an output label y_i that state its means or class, e.g. $y_i \in \{Come, Bye\}$ if the gesture corresponds to come or bye. The aim of the supervised learning is to estimate the mapping from $\vec{x}_i$ to y_i, given a training set $\mathcal{D}$ of N independently identically distributed (i.i.d) examples $\vec{x}_i$ with their corresponding labels $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\} \in (\mathcal{X}, \mathcal{Y})$, where $\mathcal{X}$ and $\mathcal{Y}$ are the a set of feature vectors and their label set, correspondingly. Hence, for any new sample $\vec{x}'$, the algorithm should accurately predicts the label $y(\vec{x}')$. If the label set of the training data is continuous i.e. $y \in \mathbb{R}$ then the learning task is called regression while it is called classification if the label set belongs to discrete labels. The supervised algorithms belong to two families, which are either generative or discriminative algorithms. The generative algorithms try to estimate the predictive density $p(y|x)$ by estimating the class-conditional density $p(x|y)$, which is related to the joint density $p(x, y)$ from which the pair $(\vec{x}_i, y_i)$ may be drawn. The discriminative algorithms try to estimate $p(y|x)$ without caring about how the data have been generated, e.g. SVM (Chapelle et al., 2006b; Barber, 2013).

3.3 Unsupervised Learning

In supervised learning, we assumed that the training data $\mathcal{D}$ were available as a set of feature vectors $\mathcal{X}$ and their labels $\mathcal{Y}$. However, the labels are not always available and only feature vectors set $\mathcal{X}$ is attainable. Hence, the aim of unsupervised learning is to detect the underlying similarities and group (cluster) similar features vectors together (Theodoridis and Koutroumbas, 2009). In other words, the objective of unsupervised learning is to model the distribution of the data $p(x)$ (Chapelle et al., 2006b). Unsupervised learning has been used in several applications in engineering and social science, including outlier detection, data mining, dimensionality reduction, pattern recognition, quantile estimation, remote sensing, image segmentation, and image and speech coding.

The task of the unsupervised learning faces many challenges, e.g. the effect of the noise and data sparseness. Additionally, there is no prior information about the number,

the size and the shape of the clusters. Thus, many assumptions are used to simplify the unsupervised learning task. For example, some techniques assume the number of the clusters to be known, and the task is only to assemble the data into these clusters, e.g. k-means (MacQueen et al., 1967). Other methods presume the number of the clusters to be unknown but use a particular window size to estimate the kernel density, where the window size is significant to the results, i.e. different window size give different clusters, e.g. mean-shift algorithm (Comaniciu and Meer, 2002; Cheng, 1995; Fukunaga and Hostetler, 1975). Therefore, different algorithms are proposed to solve the various problems. Some of these techniques are OPTIC, DBSCAN, principal component analysis (PCA), minimum spanning trees, hierarchical clustering, Gaussian mixture models, self-organizing maps and hidden Markov models.

In this section, we will discuss some of these techniques since they used in this thesis, which are the k-means algorithm and the mean-shift algorithm.

3.3.1 k-means

Among unsupervised learning techniques that are based on minimising a formal objective function, the simplest and the most widely used to solve the clustering problem is k-means clustering (MacQueen et al., 1967). k-means is a process that objects to partition N individual of a d-dimensional population into k sets. It should minimise the mean squared distances (squared error distortion) between the samples and the centres (centroids) of their k sets. Initially, k samples are randomly selected as centroids of the required k clusters, one for each cluster (Larose, 2014). The location of the initial centroids is crucial for the optimisation process. Thus, choosing different initial locations may give different clusters. So, the preferred choice is to distribute them as far as possible from each other. Next, it associates each sample in the dataset to the nearest centroid. Then, the centroids are updated to the point of the gravity of the estimated sets from the previous step. This process is iterated till the locations of the centroids are approximately settled. Several algorithms are proposed to find the minimal local solution, e.g. the generalised Lloyd's algorithm (Kanungo et al., 2002). k-means is used in Section 5.3 and Section 7.2.

3.3.2 Mean-shift

Unlike the k-means, where the number of the clusters is required in prior, the mean shift estimates it automatically. Additionally, the mean shift is unrestricted to the shape of the clusters. The mean shift algorithm, which is a non-parametric technique, is proposed by Fukunaga and Hostetler (1975). All the analyses are performed in the feature space. The idea behind the mean shift clustering algorithm is that the feature space can be considered as the empirical probability density function (pdf) of the represented parameter. Hence, it detects the mode of the unknown density (that's why it is called mode-seeking algorithm) by locating the local maxima of the pdf, which corresponds to the dense region in feature

space. The cluster that is associated with the determined mode then is described based on the feature space (Comaniciu and Meer, 2002). The procedure is started by locating a kernel at each of the given N samples $\vec{x}_i, i = \{1, 2, \cdots, N\}$ that are drawn from a density function $f(\vec{x})$ in a d-dimensional space $\mathbb{R}^d$. So, the average of the samples within a particular window size h_w is determined by using the kernel density estimate. Consequently, the centre of each window is shifted to the mean of the samples in that window. This procedure is iteratively applied until the windows have no more movement. According to Comaniciu and Meer (2002), the multivariate kernel density estimate determined by kernel $K(\vec{x})$ is given by

$$f(\vec{x}) = \frac{1}{N{h_\mathrm{w}}^d} \sum_{i=1}^{N} K\left(-\frac{\|\vec{x} - \vec{x}_i\|^2}{{h_\mathrm{w}}^2}\right), \tag{3.1}$$

supposed that the profile of the kernel $K(\vec{x})$ is a normal multivariate profile

$$k(x) = \mathrm{e}^{-\frac{1}{2}x}, \quad x \geq 0. \tag{3.2}$$

The stationary locations of the density function (modes) can be found by equalising the gradient of Eq. (3.1) to zero

$$\frac{-2}{N{h_\mathrm{w}}^{(d+2)}} \sum_{i=1}^{N} (\vec{x}_i - \vec{x}) K'\left(-\frac{\|\vec{x} - \vec{x}_i\|^2}{{h_\mathrm{w}}^2}\right) = 0 \tag{3.3}$$

where $K'(\vec{x})$ represents the derivative of the kernel $K(\vec{x})$. The mean shift uses an iterative method to find the solution of this gradient, which is the local maximum. The increment is given by

$$\delta\vec{x} = \frac{\sum_{i=1}^{N} \vec{x}_i K'\left(-\frac{\|\vec{x}-\vec{x}_i\|^2}{{h_\mathrm{w}}^2}\right)}{\sum_{i=1}^{N} K'\left(-\frac{\|\vec{x}-\vec{x}_i\|^2}{{h_\mathrm{w}}^2}\right)} - \vec{x}, \tag{3.4}$$

where $\vec{x}$ is the current mean. The mean shift vector $\delta\vec{x}$ in the iterative computations is guaranteed to tend towards the direction of increasing density and converge to a local mode of the distribution (Comaniciu and Meer, 2002). A small perturbation may be added to the current mean vector to overcome saddle locations (Anand et al., 2014).

Due to the powerful features of the mean shift, it has been used in several applications, e.g. image smoothing and segmentation (Wang et al., 2004). It was also extended to work on nonlinear data e.g. kernel-induced feature space (Vedaldi and Soatto, 2008). Several other fields are mentioned in (Anand et al., 2014). The mean shift clustering algorithm is used in Section 8.2.

3.4 Semi-Supervised Learning

Commonly, the labelled data are available in small amount $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_l, y_l)\} \in (\mathcal{X}, \mathcal{Y})$, where l is the number of labelled data in the training set, while the unlabelled data is available in large amount $\{\vec{x}_{l+1}, \ldots, \vec{x}_{l+u} \in \mathcal{X}\}$, u is the number of the unlabelled data in the training set and $u > l$. For example, the gesture data are easily acquired from different sources but it is difficult to get labels to all the available gestures since it assigned by experts. The semi-supervised learning (SSL) uses both type of data, labelled and unlabelled, to enhance the accuracy of the classifier more than if the labelled data are used only (Barber, 2013).

In most cases, the researchers look to the semi-supervised learning as a supervised learning with some additional information about the distribution $\mathcal{X}$. This assumption is appropriate for the application trying to predict the label of the given $\vec{x}$ as in the supervised learning. However, it is not suitable if the number and the structure of the classes have to be concluded. In the latter case, the semi-supervised learning is seen as unsupervised learning with some constraints (Chapelle et al., 2006b). Due to the use of two types of data, the labelled and the unlabelled, there are two different settings of the semi-supervised learning, which are inductive and transductive semi-supervised learning. The inductive semi-supervised learning tries to infer the mapping $f_{\text{ssl}} : \mathcal{X} \mapsto \mathcal{Y}$ so that f_{ssl} can predict all future samples $\vec{x}$, which is similar to the supervised learning. The transductive semi-supervised learning is trying to estimate the labels of the unlabelled training set only i.e. trying to estimate the mapping $f_{\text{ssl}} : \mathcal{X}^{l+u} \mapsto \mathcal{Y}^{l+u}$ (Zhu and Goldberg, 2009).

Although, the unlabelled data do not have information about the mapping $f_{\text{ssl}} : \mathcal{X} \mapsto \mathcal{Y}$, the data give better imagination about the marginal distribution $p(x)$. The semi-supervised learning link this distribution with the conditional distribution $p(y|x)$ by using some assumptions. Following Subramanya and Talukdar (2014), the assumptions are:

- *Smoothness Assumption:* The neighbour samples in the high dense regions should have close output (same class) or the function $f_{\text{ssl}} : \mathcal{X} \mapsto \mathcal{Y}$ is continuous in regression problems. The self-training learning uses this assumption.

- *Cluster Assumption:* The decision boundary between the classes should fall into the low-dense region. The semi-supervised SVM (S3VM) and some of the graph-based methods use this assumption.

- *Manifold Assumption:* The high dimensional data fall into a low dimensional manifold. This hypothesis helps to handle high dimensional data in the machine learning on a comparatively low dimension. Some implementation of the manifold assumption can be found in the work of Goldberg (2010).

Different assumptions led to different semi-supervised learning methods (Zhu and Goldberg, 2009); the standard methods are explained in this section.

3.4.1 Self-training methods

The classifiers that based on the self-training methods are updated on their prediction. The self-learning method is implemented as follow: first training the classifier on the labelled samples $\mathcal{D}_{\mathrm{l}} = \{(\vec{x}_i, y_i)\}_{i=1}^{l}$ as supervised learning. Secondly, select a subset of samples $\mathcal{S}$ from the unlabelled data $\mathcal{D}_{\mathrm{u}} = \{\vec{x}_i\}_{i=l+1}^{l+u}$, i.e $\mathcal{S} \subset \mathcal{D}_{\mathrm{u}}$, and estimating the label $\hat{y}_i$ for each sample $\vec{x}_i \in \mathcal{S}$. Thirdly, if $\mathcal{S}$ has N' samples, then this samples and its predicted labels $\{(\vec{x}_i, \hat{y}_i)\}_{i=l+1}^{l+N'}$ are combined with $\mathcal{D}_{\mathrm{l}}$ to retrain the classifier. Finally, the number of labelled sample l updated to $l = l + N'$ and the subset $\mathcal{S}$ is excluded from the unlabelled data $\mathcal{D}_{\mathrm{u}}$. However, all the samples in the $\mathcal{S}$ are gotten labels, only the samples of highly confidence label are used to retrain the classifier. The same process is repeated till the last subset in the unlabelled data. Notably, the number of samples N' in the subset $\mathcal{S}$ may vary from iteration to another. Several advantages recognise the self-training methods. The major feature is the simplicity and it used as a wrapper method. Wrapper means the selection of the classifier's type is totally free i.e. it can be varied from simple like kNN to complex like SVM. Additionally, it doesn't effect on the inner structure of the classifier and hence it appropriates for the data stream. This method can be generative or transductive depending on the used classifier. However, the false labels assigned by the classifier potentially will affect the performance of the classifier after the next retraining cycle and hence this method is very sensitive to the outliers (Zhu and Goldberg, 2009). This method of semi-supervised learning is used in all of the proposed classifiers in this thesis (Chapter 5, Chapter 6 and Chapter 7).

The early research discussed the self-training is (Yarowsky, 1995). Several other literatures are proposed for the self-learning includes (Scudder, 1965; Riloff et al., 2003; Rosenberg et al., 2005). For a further understanding of the self-learning, the analysis is discussed in (Haffari and Sarkar, 2007; Culp and Michailidis, 2008).

3.4.2 Co-training

According to Blum and Mitchell (1998), the idea of the co-training is to look at the labelled data from two different views and use a different classifier for each view. Given a labelled data $\mathcal{D}_{\mathrm{l}} = \{\vec{x}_i, y_i\}_{i=1}^{l}$ and an unlabelled data $\mathcal{D}_{\mathrm{u}} = \{\vec{x}_i\}_{i=l+1}^{l+u}$, two views for each sample in the labelled data $\mathcal{X}_{\mathrm{l}} = \left[\mathcal{X}_{\mathrm{l}}^{(1)}\mathcal{X}_{\mathrm{l}}^{(2)}\right]$ are found. Several methods are available to implement this process such as: dividing the features of the sample to two groups; dividing the samples themselves to two groups or by compute some transformations of some information in the original data. The two views should be conditionally independent and each view should have enough information to classify the samples. Two classifiers ${f_{\mathrm{ssl}}}^{(1)}$ and ${f_{\mathrm{ssl}}}^{(2)}$ are trained on $\mathcal{X}_{\mathrm{l}}^{(1)}$ and $\mathcal{X}_{\mathrm{l}}^{(2)}$, respectively. The unlabelled data may also need to be divided into two views $\mathcal{X}_{\mathrm{u}} = \left[{\mathcal{X}_{\mathrm{u}}}^{(1)}{\mathcal{X}_{\mathrm{u}}}^{(2)}\right]$ or may the whole unlabelled data provided to both classifiers depending on the method that be used to implement the two

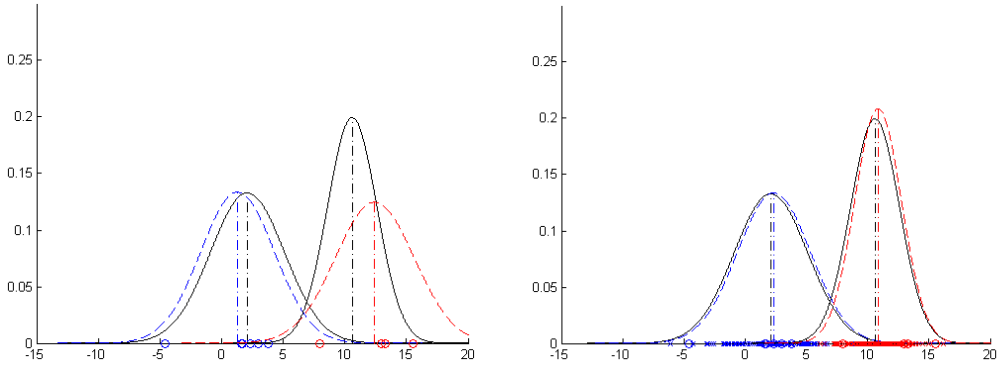
views on the labelled data. Then, the two classifiers ${f_{\text{ssl}}}^{(1)}$ and ${f_{\text{ssl}}}^{(2)}$ are applied on the unlabelled data ${\mathcal{X}_{\text{u}}}^{(1)}$ and ${\mathcal{X}_{\text{u}}}^{(2)}$, respectively (if one view is available for in the unlabelled data then the whole data are applied to both classifiers). Since the goal of the co-training is to make cross learning between the classifiers, the samples of the most confident labels for each classifier should be added to the training data of the other classifier. In case of each view contains some features from the same sample, Let $\tilde{\mathcal{Y}}_1$ and $\tilde{\mathcal{Y}}_2$ be the most confident labels with respect to ${f_{\text{ssl}}}^{(1)}$ and ${f_{\text{ssl}}}^{(2)}$, respectively. The corresponding feature data will be $\tilde{\mathcal{X}}^{(1)}_{(1)}$ and $\tilde{\mathcal{X}}^{(1)}_{(2)}$ from the first view data and $\tilde{\mathcal{X}}^{(2)}_{(1)}$ and $\tilde{\mathcal{X}}^{(2)}_{(2)}$ from the second view data. Then the data $\{\tilde{\mathcal{X}}^{(1)}_{(2)}, \tilde{\mathcal{Y}}_2)\}$ and $\{\tilde{\mathcal{X}}^{(2)}_{(1)}, \tilde{\mathcal{Y}}_1)\}$ should be added to training data ${\mathcal{X}_{\text{l}}}^{(1)}$ and ${\mathcal{X}_{\text{l}}}^{(2)}$ of the classifier ${f_{\text{ssl}}}^{(1)}$ and ${f_{\text{ssl}}}^{(2)}$ respectively. After updating the training data of each classifier, the classifiers should be retrained. This process can be applied iteratively to all subsets of the unlabelled data. The subset should be removed from the unlabelled pool after added it to the training data. Finally, the labels of the test data can be obtained by voting the two classifiers or by averaging their output (Didaci and Roli, 2006).

The co-training is submitted by Blum and Mitchell (1998) and used in a wide semi-supervised learning applications e.g. (Zhou et al., 2007; Chawla and Karakoulas, 2005). More details about the analysis of the co-training can be found in (Balcan et al., 2004).

3.4.3 Probabilistic generative models

The unlabelled data may help to estimate the distribution of all classes together. The idea of the *mixture models* is to break down this mixture into individual classes. The labelled data may tell us the number of the classes, but the parameters of their distribution are not accurate, e.g. the mean, variance and prior probabilities for Gaussian distributions. Using the whole data (labelled and unlabelled) helps more accurate estimation of these parameters. Fig. 3.1 explain how the unlabelled data help to estimate more accurate parameters. In this example, there are two classes of one-dimensional data. The original distribution is plotted in solid black while the estimated distributions of the two classes are plotted in dashed blue and dashed red, respectively. The means of all distributions are plotted by centreline with the corresponding colour. Additionally, the symbol ('o') represents the labelled and the symbol ('x') represents the unlabelled samples. The distribution and the mean of each class that estimated from the labelled data only (Fig. 3.1(a)) are shifted and not accurate, while those are estimated by using both of the labelled and the unlabelled data (Fig. 3.1(b)) are close to original values. The parameters of a proposed model should be optimised to give a maximum probability for the data generated from this model and similar to the training data (Goldberg, 2010).

Since the dataset $\mathcal{D} = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_l, y_l), \vec{x}_{l+1}, \cdots, \vec{x}_{l+u}\}$ in the semi-supervised learning consists of the labelled and the unlabelled data, the maximum likelihood esti-



(a) Only Labelled Data (b) Labelled and unlabelled data

Figure 3.1: The accuracy of the estimated parameters is improved when the unlabelled data are used. The black solid line is the original distribution, while the blue and the red dashed lines represent the estimated distribution for the two classes, respectively. In (a), only the labelled data are used to estimate the distribution of the classes, while in (b) both data are used to obtain more accurate approximation.

mation (MLE) cannot be solved analytically. The log likelihood function in the semi-supervised learning is described as:

$$\log\left(\mathcal{D}|\theta_{\text{prob}}\right) = \log\left(\prod p(\vec{x}_i, y_i|\theta_{\text{prob}})_{i=1}^{l} \prod p(\vec{x}_i|\theta_{\text{prob}})_{i=l+1}^{l+u}\right) \tag{3.5}$$

$$= \sum_{i=1}^{l} \log p(y_i|\theta_{\text{prob}}) p(\vec{x}_i|y_i, \theta_{\text{prob}}) + \sum_{i=l+1}^{l+u} \log p(\vec{x}_i|\theta_{\text{prob}}). \tag{3.6}$$

Where the model parameters are represented by θ_{prob}. The first term in Eq. (3.6) is the standard log likelihood in the supervised learning. The second is an additional term used in semi-supervised learning to include the unlabelled data in the parameters estimation (Zhu, 2011). Hence, the MLE needs to fit both data in non-concave problem. Non-concave problems can be solved by finding a local maximum using algorithms such Expectation-Maximization (EM) (Dempster et al., 1977) or may solve by direct optimization methods e.g. (Liu and Nocedal, 1989). The generative model assumes that the data come from a mixture, where all probabilities e.g. prior and conditional are correct. This is the weakness of the generative models since it is hard to appraise the correctness of the model due to the lack of the labelled data (Goldberg, 2010).

Some of the theoretical analysis of mixture model is proposed in (Chapelle et al., 2006a; Ratsaby and Venkatesh, 1995) whereas, Cozman et al. (2003) show that the unlabelled data may decrease the model performance if the assumption was wrong. Several

applications of the mixture model are implemented in (Nigam et al., 2000; Fujino et al., 2008)

3.4.4 Semi-supervised support vector machines

Initially, Vapnik (1998) proposed the Transductive Support Vector Machine (TSVM) to use the labelled and unlabelled data by using the transductive setting. Bennett et al. (1999) developed the learned function of the TSVM to apply it to the unseen test samples. By this update the method becomes not transductive but inductive, hence it is more relevant to called as *Semi-Supervised Support Vector Machine* (S3VM). The S3VM uses the labelled data to train a normal SVM and uses the unlabelled data by the optimization process i.e. finding the optimum margin with the existence of the unlabelled data. It uses the *cluster assumption* of the semi-supervised learning, which means the data in the same cluster should have same label (Chapelle et al., 2006c). Olivier Chapelle proposed numerous implementation of S3VM (e.g. Chapelle et al., 2008, 2006a,c). Due to different types of the optimization, there are a wide range of implementations of the S3VM e.g (Yang et al., 2009b; Karlen et al., 2008).

3.4.5 Graph-based methods

The labelled and unlabelled data are used as vertices to construct a graph-based semi-supervised learning. Commonly, the obtained graph is large if the unlabelled data size is big. These vertices are connected to each other (or to the k-nearest neighbour) by weighted edges, which represent the similarity between the pairs of the data points. The graph-based models adopted the smoothness assumption, which assumes the labels in the graph is varied smoothly. The labelling process is started by finding labels to the unlabelled vertex. The unlabelled sample is labelled with the same label of the sample that is connected to it with heavy edges (Zhu and Goldberg, 2009). Suppose the edge weight θ_{ij} connects the sample $\vec{x}_i$ and the sample $\vec{x}_j$ then it is computed as follows:

$$\theta_{ij} = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2{\sigma_{\text{graph}}}^2}\right) \tag{3.7}$$

where σ_{graph} is the bandwidth parameters, which control the drop of the weights. There are several heuristics to specify the weights including:

- Fully connected graph: all vertices are connected to each other.
- kNN graph: each vertex is connected to the kth neighbour vertices.
- b-matching graph: each vertex is connected to b other vertices with symmetric weights. Sometimes the weights set to one for the b connected vertices and set to zero for other vertices

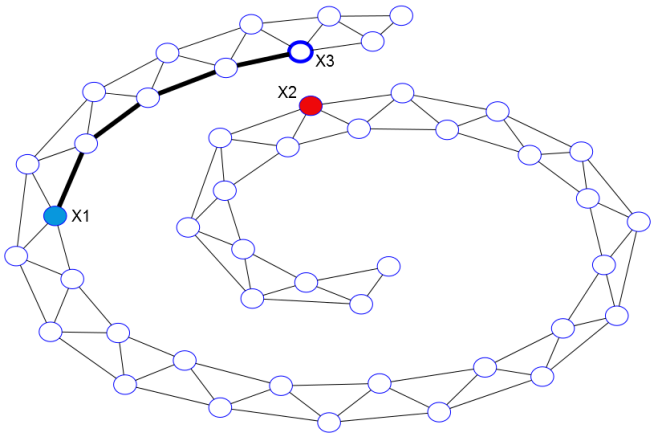


Figure 3.2: The graph-based Label interpretation. Where the unlabelled sample $\vec{x}_3$ should be labelled with the same label of the sample $\vec{x}_1$ because it is the closer labelled sample in the graph. Although, the sample $\vec{x}_2$ is closer than $\vec{x}_1$ in the Euclidean space, it is considered further than $\vec{x}_1$ in the graph. The figure is adapted from Zhu and Goldberg (2009)

.

- ϵNN graph: every two samples are considered connected if the Euclidean distance between them is equal or less than ϵ. All edges between unconnected samples are set to zero, while the edges of the connected samples are either set to one or computed by using Eq. (3.7).

Fig. 3.2 shows an example of how the samples are labelled. If $\vec{x}_1$, $\vec{x}_2$ are two labelled samples (vertices). Then, the label y_3 of an unlabelled sample $\vec{x}_3$ is estimated from the neighbours in the graph. The neighbours may in turns labelled by the neighbour's neighbours. Using this stepping stones, y_3 is assumed to be similar to y_1 although it closer to y_2 in Euclidean space.

The graph-based models try to minimise the loss of the labelled data at the same time they ensure smoothness of the whole data. Choosing different loss functions, regularisations, and the ways to solve the problem leads to several Graph-based semi-supervised learning algorithms. These algorithms include: Mincut, randomized Mincut, Boltzmann machine, graph random walk, Gaussian Random Fields and Harmonic Functions, Manifold regularization, Local and Global Consistency, Graph Kernels, Spectral Graph Transducer, local averaging, density based regularization, alternating minimization, boosting, Tree-Based Bayes, and some other methods (Subramanya and Talukdar, 2014; Zhu, 2005). Some literature discussed the graph construction itself (Hein et al., 2007). The theoretical analysis is described in (Zhang and Ando, 2006; Johnson and Zhang, 2007) and some application is proposed in (Goldberg and Zhu, 2006; Niu et al., 2005).

3.5 Data Streams

A data stream is a sequence of samples that are continuously presented to the system for an unlimited time. These samples are originated according to a probability distribution, which may have "concept-drift" i.e. it changes in the course of time. In this scheme, new concepts may emerge, and known concepts may disappear (Faria et al., 2013). For example, the gesture recognition system on some devices like medical devices, robotics and smart devices, may continuously receive several types of gestures from different users for unlimited time.

The basic problems of data streaming are the "infinite length", the "concept-drift" and the "concept evolution". The first two issues were addressed by most of existing online algorithms (Masud et al., 2011). While a limited number of algorithms are presented to solve the third issue alone or in a combination of the other two issues (e.g. Masud et al., 2012). So, we proposed several algorithms, each of them solves all the problems altogether. For example, the solution of the concept evolution is discussed in Section 3.7. Due to the infinite length, the size of the data collected from the stream will increase, and then both tremendous hardware resources and an indefinite time (infinitely long time) will be required to train the system with these data. In contrast, the system should be retrained with the data continuously to overcome the effects of concept-drift, e.g. statistical properties of the classes or the appearance of novel classes. It will be very complicated to retrain the system continuously if the data are originated from streams of infinite length. In addition, the emersion of new classes within time or vanishing some of the existing classes are significant problems. Thus, contrary to traditional static data, data streaming is characterised by the following properties (according to Farid et al., 2013):

- it is dynamic,
- the number of samples is infinite, and their dimension is high,
- the samples are presented only once and are not repeated,
- the data arrive at high speed,
- and the data properties may change over time.

The most recent studies offer solutions to a single problem either by the incremental learning for "infinite length" or by novelty detection technique for arriving data belong to unknown classes or outliers. Additionally, most of the novelty techniques were implemented as one-class classifiers. But in the reality all problems mostly happen together hence, we need to resolve all these issues ("infinite length", "concept-drift", "novel classes", and "multi-class system") in one algorithm. However, in order to apply it to online classifications, these problems need to be solved while maintaining a sufficient speed of

classification. The combined "infinite length" and "concept-drift" problem may be solved by incremental learning. The incremental learning allows a continuous retraining of the system without the need for the old data when updating the system. This incremental learning algorithm, however, should have the ability of novelty detection, and it should be also able to work in a multi-class system. Although, applying the novelty detection on the data stream is an essential challenge since concepts in data streaming are hardly ever constant (Gama, 2010).

3.6 Incremental Learning

An intelligent biological system regularly can learn through their lifetimes and make the experience over time. So, the incremental learning is a significant facility to build a brain-like intelligence machine (He, 2011). The definition of the incremental learning is "Incremental learning is a machine learning paradigm where the learning process takes place whenever new example(s) emerge and adjusts what has been learned according to the new example(s)." (Geng and Smith-Miles, 2009).

According to He (2011) there are three aims of the incremental learning algorithms: applying the previous knowledge to the presently received data to facilitate learning from these new data, expanding the knowledge over time to maintain a reasonable decision-making accuracy, and realising global generalisation through the continuous learning to achieve the goals. The incremental learning is not only needed to emulate the intelligent biological system, but it is also necessary for several real-world applications including the stream data. In contrast to the traditional environment where the whole training data are available for the initial training, the data are streamed continuously in the data stream environment. Therefore, the machine learning system should be able to update itself incrementally to reacts with the continuously arrived data, which may have concept-drift. Additionally, the classifiers should be updated in real-time to be able to follow the data stream.

Different approaches have been proposed to deal with data streams and incremental learning. Generally, the techniques may be subdivided into single model classifiers and ensemble classifiers, i.e. a combination of multiple classifiers. While single model classifiers are required to update themselves and/or their structure as soon as new data become available. Ensemble approaches create more than one classifier. Each classifier may be based on different hypotheses and different subsets to simplify the update. New classifiers are trained on the new training data and then added to the ensemble. Unfortunately, this method may be inappropriate for very large amounts data, because such ensembles may grow an immense number of classifiers.

An incremental SVM classifier was proposed by Diehl and Cauwenberghs (2003). The parameters are constantly adapted to new data. Chen et al. (2008) used the history of the data stream to build higher order models for each concept that are constantly changed.

Ensemble-based classifiers were proposed by Masud et al. (2011) and Ditzler and Polikar (2013). The former one addresses novelty detection while the later one focusses on the concept-drift in imbalanced data systems.

In some semi-supervised learning approaches, a manual assignment to some samples that are used in the next training is possible. This procedure is called "active learning" (Settles, 2010), which is implemented by Schumacher et al. (2012) for gesture recognition. These techniques solve just a part of the problems which is using all or some of the unlabelled data to retrain the classifier, but it still needs all the old data in the retraining process. Additionally, small amounts of incorrect labels can decrease the accuracy of the system in the next training, which leads to failure of the system in the end.

Since neural networks may be used to approximate and function, they have been used for decades in machine learning. Consequently, a variety of incremental neural networks exist throughout the literature. Polikar et al. (2001) introduced the well-known learn++ algorithm, which is a supervised learning algorithm that uses an ensemble of neural network classifiers to achieve incremental learning even if new classes are introduced. Huang et al. (2004) and Xun and Chang-shan (2005) presented a simple sequential growing and pruning algorithm for additive or RBF network (GAP-RBF). It uses the concept of "significance" of a neuron and links it to the learning accuracy. Feng et al. (2009) proposed a method to automatically increment the number of the hidden neurons to minimise the error. The output weights are updated incrementally while the network grows. Huang et al. (2006a) proposed an incremental extreme learning machine (IELM) by adding nodes that are randomly generated to the single-layer feed-forward network (SLFN) and computed the output weight analytically for the new nodes only. Liang et al. (2006) developed IELM for batch learning of chunks of data. Huang and Chen (2007) proved that the convergence rate of IELM might be improved by recalculating the output weights of the existing nodes using a convex optimisation method when new random nodes are added. Huang and Chen (2008) proposed a method to add only the nodes that decrease the residual error the most to make the IELM network compact and reduce the complexity of the calculations. Lan et al. (2009) adapted the algorithm proposed by Liang et al. (2006) to be more stable using an ensemble. Yang et al. (2013) proposed a parallel chaos search based incremental extreme learning machine (PC-ELM). It uses an additional step to obtain a more compact network architecture. Ye et al. (2013) proposed a time-variant extreme learning machine network to work with non-stationary environments. Guo et al. (2014) modified the algorithm presented by Huang et al. (2006b). The new algorithm is based on an extreme learning machine (ELM), a unified least square support vector machine (Suykens and Vandewalle, 1999) and a proximal support vector machine (Fung and Mangasarian, 2005) to provide a method for incremental multi-class regression. Yin et al. (2014a) presented an online fault diagnosis method based on incremental super vector data description (ISVDD) and an ELM with incremental output structure (IOELM) to solve the problem when the class number in fault diagnosis is not constant.

3.7 Incremental Class Learning

Recently, several techniques for machine learning related to incremental learning have been presented (e.g. Guo et al., 2014; Ye et al., 2013). Most of the proposed incremental algorithms for multi-class classification can be updated only for the new data that belong to fixed classes, which the classifier has been trained initially. The learning process is not limited to get more information about the objects that already known. However, the unique property of learning is updating the old information with new concepts. Hence, adding new classes or concepts to the trained classifiers is essential in the learning process. Regrettably, if new classes are available, the traditional classifiers should be retrained with the entire data i.e. the initial data and the new data of the new classes. This technique is time-consuming, and it is not appropriate for streamed gesture or online recognition. In data stream, the number of classes may be changed, as mentioned in Section 3.6, and the old data are tough to store. Hence, in addition to updating the classifier incrementally to new samples that belong to the known classes, the classifier should be able to learn new concepts without a need for the old data.

Zhao et al. (2014) and Mańdziuk and Shastri (2002) proposed an incremental class learning using neural network. Zhang et al. (2006) used the SVM to present incremental class learning algorithm. Faria et al. (2013) used an ensemble classifier to implement the incremental class learning.

3.8 Ensemble Learning

Commonly, humans collect several opinions and then they weight these opinions to obtain a final decision. A similar technique is adopted in the machine learning compensating the classification results. It is implemented by combining multiple diverse classifiers to obtain a classifier that outperforms every one of basic classifiers (Rokach, 2010). The outputs of the classifiers may be weighted or voted by majority to get the final output. This methodology is called "ensemble methods" in the literature. Regularly, the combined classifiers belonging to the same basic model with some minor variations. However, models belong to a different family are also used in the ensemble method (e.g. Masud et al. (2011)). Initially, the ensemble method was established when Tukey (1977) used an ensemble of two regression models to fit the data and the residuals separately. Afterwards, Dasarathy and Sheela (1979) used two classifiers to partition off the input space. During the Nineties, several algorithms have been developed to achieve the ensemble method (e.g. Martyna et al. (1992) and Berg and Neuhaus (1992)). Simultaneously, Schapire (1990) suggested the fundamentals of the well-known ensemble algorithm "AdaBoost" (Freund et al., 1996). It used weak classifiers, i.e. their performance was a bit better than arbitrary classification, to construct a strong ensemble classifier. The ensemble methods are also employed in the unsupervised learning (e.g. Valpola and Karhunen, 2002). The impact of successful

improvement of the ensemble classification performance, they are applied in a wide range of fields. Some of these fields are: image retrieval (e.g. Lin et al., 2006), bio-informatics (e.g. Tan et al., 2003), finance (e.g. Leigh et al., 2002), medicine (e.g. Mangiameli et al., 2004), geography (e.g. Bruzzone et al., 2004), manufacturing (e.g. Maimon and Rokach, 2004) and chem-informatics (e.g. Merkwirth et al., 2004). According to Rokach (2010) there are different algorithms of ensemble methods including:

- the wisdom of crowds,
- the Bagging algorithm,
- the Boosting algorithm,
- the AdaBoost algorithm.

Each algorithm has different features and is used in various applications. In this thesis, the first type "wisdom of crowds", which is also called "voting by majority" in other literature, particularly, in Chapter 8.

3.9 Novelty Detection

"Novelty detection" is a technique that is used to indicate the samples which do not belong to the concept learned by the machine learning system. In the literature, novelty detection is also known as "anomaly detection", "one-class classification" or "outlier detection" (Pimentel et al., 2014). Novelty detection is mainly applied by training a one-class classifier which assigns the labels new data normal if the sample is recognized and abnormal if it is not recognized. We call the normal data "known", i.e. it is generated from the same distribution of the known data, and the abnormal data "unknown",i.e. it is generated from a distribution that differs from the distribution of the known data, to reflect that the mineralogy of the abnormal spectral data is unknown. Any distribution can be modelled if a sufficient amount of data is available. This is often the case for training data, i.e. the data which are used to adapt the classifier. In contrast, different types of abnormality have a small number of, at least at the time of training, and maybe solely contained within the test data, i.e. that data that the classifier is applied to. Hence, it is commonly possible to construct a model of the known data while it is not possible to estimate a model of the unknown data. Although the novelty detection is formally a one-class problem, the known concept may be composed of different classes of normal data, and different types of abnormality belong to the unknown concepts (Faria et al., 2013), e.g. new types of minerals, camera calibration, noisy data. The unknown samples in the data stream may belong to new classes that the classifier did not learn, an outlier, or a noisy sample of known data. Hence, the classifier should indicate this sample to exclude it from the data that are used for updating the classifier in the incremental semi-supervised

learning. The fails of indicating these samples lead to training the classifier with wrongly-classified samples thus the classifier may fail to detect further outliers. Consequently, the classifier will accept more outliers in the continuous updating, which leads to breaking down the system. The incoming known data may be added to update the model by the incremental learning as described in Chapter 5, Chapter 6, and Chapter 7. In the case of collecting enough amount of unknown data that originated from the same concept, a new class may be added to the model by using the incremental class learning described in Chapter 8.

The novelty detection approaches should detect most of the unknown samples while maintaining the *false alarm rate* low as possible i.e. minimising the rate of the known samples that incorrectly indicated as novel samples. Hence, the main metrics that are used to evaluate the novelty detection technique attributes are the *true positive rate* and the *false alarm rate*. The true positive rate is the number of novel samples that are correctly indicated divided by the total number of novel samples. The false alarm rate is the number of the normal samples that are indicated as novel samples divided by the total number of normal samples. In the case of a one-class system, receiver operating characteristic (ROC) curves are used to compromise the true positive rate and the false alarm rate. These curves are difficult to use in multi-class system, hence we used M_{new} and F_{new} (explained in more details in Section 5.2) to represent the above two metrics.

Due to the importance of the novelty detection in a wide range of applications, there were different approaches proposed to solve the problem. Pimentel et al. (2014) categorised the novelty detection approaches into five main techniques:

- distance-based techniques,
- reconstruction-based techniques,
- information-theoretic techniques,
- probabilistic techniques and
- domain-based techniques.

Clifton et al. (2006); GarcíA-RodríGuez et al. (2012); He et al. (2006) are examples of distance-based approaches, reconstruction-based approaches, and information-theoretic approaches, respectively. There are two main techniques for the probabilistic approaches: parametric and non-parametric. The parametric techniques assume a specific distribution of samples and measure some parameters of that distribution based on the initial training data set. A sample is considered as belonging to a novel class or being an "outlier" if it does not match well with this trained distribution. Different techniques are used in these approaches such as: mixture models approach (e.g. Tseng et al., 2006; Zorriassatine et al., 2005), extreme value theory (e.g. Clifton et al., 2008, 2009; Clifton, 2009; Hugueny

et al., 2009; Worden et al., 2002) and state space model (e.g. Gwadera et al., 2005; Lee and Roberts, 2008; Ntalampiras et al., 2011). The non-parametric techniques use the old data themselves, e.g. kernel density estimates, to predict the distribution. Hence, they are not restricted to a specific distribution that is represented by some parameters. There are two main techniques for the non-parametric approach: kernel density estimator (e.g. Bishop, 1994; Yeung and Chow, 2002; Tarassenko et al., 1995) and negative selection (e.g. Dasgupta and Majumdar, 2002). A common kernel density estimator is the Parzen window kernel density estimator (PKDE), which was originally introduced by Parzen (1962). The problem of most non-parametric techniques is that all of the old data are required to model the distribution. Hence, they are difficult to use with data streaming, i.e. a continuously increasing dataset. We update the PKDE to fit the incremental requirements (cf. Section 5.3). In this thesis, most of the proposed methods outputs are compared with the support vector data description (SVDD) which is a well-known approach proposed by (Tax and Duin, 2004). SVDD is a novelty detection algorithm belong to the domain-based category, and it can learn incrementally (cf. Section A.1). Further details on novelty detection are given by Pimentel et al. (2014).

Neural networks have been applied in a wide range of applications, and they are considered to be the earliest algorithms that were used for novelty detection. Due to the variety of artificial neural networks, there is a broad range of approaches used for novelty detection. We category them depending on the neural network types.

Vasconcelos et al. (1995) used Multilayer Perceptron Neural Networks (MLP) for novelty detection. They considered the sample under test as novel if the output of the winner unit didn't exceed the second winner unit by a pre-defined threshold. Cordella et al. (1995) used the same approach. They, however, added a second threshold for the winner output unit. If the winner output unit did not exceed the threshold, it was considered novel. Singh and Markou (2004) added random patterns during the training to simulate samples of outliers.

Li et al. (2002) used Radial Basis Function Networks (RBF) neurons as a hidden layer in an MLP. Each output neuron corresponds to one known class. They pre-defined a threshold for each output neuron, respectively, to indicate normality. Oliveira et al. (2003) and Neto et al. (2004) used a dynamic decay adjustment (DDA) approach to perform novelty detection over short length time series.

Auto-associative neural networks (AANN) (also called auto-associator), i.e. neural networks that try to reconstruct the input at the output neurons, contain three hidden layers. The middle-hidden layer, which is called "bottleneck", has a number of hidden neurons that is less than the number of input neurons. They try to reconstruct the applied input at the output. Hence, it is sometimes used for dimensionality reduction. They were widely used for the novelty detection (Surace et al., 1997; Ko and Jacyna, 2000; Sohn et al., 2001; Manevitz and Yousef, 2000). They also have been proposed for novelty detection by Hawkins et al. (2002) and Williams et al. (2002). Since novelties are supposed to differ from the learned concept, the novelty detection is based on the reconstruction error of the

network. Similar techniques have been applied to novelty detection by Thompson et al. (2002) and Diaz and Hollmén (2002). Tax (2001) used the AANN (cf. Section A.2) as a one-class classifier within the data description, outlier and novelty detection toolbox (ddtools) (cf. (Tax, 2015)). This implementation is used as a reference to compare the proposed algorithm of the semi-supervised neural network (cf. Section 6.1).

Self-Organising Maps (SOM) are unsupervised, i.e. clustering, neural networks proposed by Kohonen (1988). A threshold is used to determine whether a sample belongs to the cluster or not. It has been successfully applied to novelty detection (Harris, 1993; Ypma et al., 1997; Emamian et al., 2000; Labib and Vemuri, 2002; Theofilou et al., 2003).

Many other neural networks are used for novelty detection, for example, Liao et al. (2007) used adaptive resonance theory for the novelty detection. More details on novelty detection and neural networks are given by Markou and Singh (2003) and Haggett (2008).

In most of the novelty detection approaches novelty scores $N_{\mathrm{Sc}}(\vec{x})$ are computed for the test sample $\vec{x}$. The sample $\vec{x}$ is classified as novel if $N_{\mathrm{Sc}}(\vec{x}) > N_{\mathrm{th}}$, where N_{th} is the novelty threshold. The novelty threshold in most of traditional methods is estimated using a validation data, which are labelled data differ from the training data.

3.9.1 Conventional novelty threshold setting

The conventional novelty detection approaches mostly require one threshold or a set of class-wise thresholds computed by additional manually labelled data called *validation data*. However, the availability of a sufficient amount of labelled data is not assumed in semi-supervised learning. A cross-validation is thus frequently applied to derive the thresholds. The cross-validation process is just an approximation to the threshold (Esbensen and Geladi, 2010), and it is time-consuming. Additionally, we need to keep all the training data because the thresholds are highly related to the information gained from the training data. Consequently, it does not work in the case of streamed data. Furthermore, the conventional approaches implicitly assume that the data and the novel class are uniformly distributed (Lee and Roberts, 2008) and thus they are not applicable to complex data distributions and thus it cannot be applied to non-linear distributions, which is the case of gesture data. In any case, the threshold distance or the threshold probability is first computed by validation data or cross-validation. The inspected sample is considered as novel in the distance-based techniques if the distance between this sample and the centres of the existing classes exceeds each distance threshold, respectively. Similarly, it is considered as novel in the probabilistic techniques if the highest probability is less than the threshold probability.

3.9.2 Extreme value theory

In contrast to conventional threshold setting, the extreme value theory (EVT) as described by Roberts (1999) and Clifton et al. (2008) avoids most of the stated inconveniences. It

is a statistical theory used to model the distribution of extreme values, i.e. maxima or minima in the one-dimensional case, in the tails of the distributions. Following Roberts (1999), let there be a set of N i.i.d. random samples $\mathcal{X} = \{x_1, x_2..., x_N\}$ each of them is d-dimensional ($d = 1$ in the classical EVT, hence all variable will be in scalar form) and distributed as given by the probability density function $f_1(x)$ (in general form $f_d(x)$). Furthermore, let the extreme value of $\mathcal{X}$ be $x_{\max}$. A cumulative distribution function is defined for the $x_{\max}$ to be $\mathcal{H}^+(x_{\max} \leq x)$, where $\mathcal{H}^+$ is the extreme value distribution (EVD); it models the location of the maxima ('+' sign refers to maxima distribution) of $\mathcal{X}$.

According to the Fisher and Tippett (1928) theorem, there are three types of EVT distributions. These distributions are the Gumbel distribution, the Weibull distribution and the Frechet distribution (Clifton et al., 2011), which are computed as:

$$\text{Gumbel,} \quad \mathcal{H}^+{}_1(y_e) = \exp\left(-\exp\left(-y_e\right)\right) \tag{3.8}$$

$$\text{Fréchet,} \quad \mathcal{H}^+{}_2(y_e) = \begin{cases} 0 & \text{if } \leq 0 \\ \exp\left(-{y_e}^{-\alpha_{\text{evt}}}\right) & \text{if } > 0 \end{cases} \tag{3.9}$$

$$\text{Weibul,} \quad \mathcal{H}^+{}_3(y_e) = \begin{cases} \exp\left(-\left(-y_e\right)^{\alpha_{\text{evt}}}\right) & \text{if } \leq 0 \\ 1 & \text{if } > 0 \end{cases} \tag{3.10}$$

where $\alpha_{\text{evt}} \in \mathbb{R}^+$ and y_e is the *reduced variate*, which is for the input x

$$y_e = \frac{x - \mu_{\text{evt}}}{\sigma_{\text{evt}}} \tag{3.11}$$

is defined by the location parameter

$$\mu_{\text{evt}} = (2\ln(N))^{0.5} - \frac{(\ln(\ln(N)) + 4\pi)}{2(2\ln(N))^{0.5}} \tag{3.12}$$

and the scale parameter

$$\sigma_{\text{evt}} = (2\ln(N))^{0.5}. \tag{3.13}$$

Both parameters depend only on N, i.e. the number of samples drawn from the underlying distribution f_1 (Embrechts et al., 2013). This helps to minimise or absorb the effect of the number of the training data on the final results and thus the threshold unchanged with increasing the training data in incremental training.

Here, the Gumbel distribution is supposed to follow the one-dimensional one-sided normal distribution with zero mean and unit variance, i.e. $\mathcal{X} \sim |\mathcal{N}(0, 1)|$. The cumulative distribution function (cdf) of Gumbel distribution for a one-dimensional variable x

$$P_{\text{evt}}(x|\mathcal{X}) = \exp\left[-\exp\left(-\frac{x - \mu_{\text{evt}}}{\sigma_{\text{evt}}}\right)\right], \tag{3.14}$$

A predefined threshold P_{th} placed at the extreme value of the known classes and novelty may be detected by exceeding P_{evt} this threshold. As is seen from the equations above, the threshold in EVT has a direct statistical interpretation and does not depend on the distribution of the classes whereas the conventional thresholds depend on the distribution of the classes.

3.9.3 Extreme value theory in multi-variate and multi-modal novelty detection

Unfortunately, the EVT works for a uni-modal, a uni-variant data only. In contrast, gesture data are multi-variate, multi-modal data. Some literature (e.g. Resnick (2013)) proposed a solution for the multi-variate data by using *component-wise extreme*, i.e. find extreme in each dimension separately. This method is not appropriate for novelty detection, where the extreme should be with respect to a multivariate model (Clifton et al., 2011). Roberts (1999) and Roberts (2000) used a mixture of Gaussian distribution to represent the normality model of the EVT for the multi-variate data. By using this method, they reduce the multi-variate problem to uni-variate. Clifton et al. (2011) showed that the variation of the extreme value distribution along a radius r of a single Gaussian distribution follows a univariate Gumbel distribution. However, they proved that using the classical uni-variate equations (Eq. (3.12) and Eq. (3.13)) to estimate the EVT parameters (μ_{evt} and σ_{evt}, respectively) will be not accurate for this method. Fig. 3.3 shows the error in the μ_{evt} and σ_{evt} between the optimum values using the MLE and the estimated values using the classical relations for $d = 2$ and $N = 100$.

However, for multi-modal distribution, the extreme values are no longer sufficient for both uni- and multivariate data (Clifton et al., 2011). Especially, when there is overlap between the components of the distribution or when they have different variance. For example, the sample $x = 10$ in Fig. 3.4 should be treated as extreme in terms of probability density $f_1(x)$ as same as $x = 1$ and $x = 28$. This fact made Clifton et al. (2011) redefine the aim of the EVT as determining the "*extremely unlikely*" events rather than the extremely large or small magnitude as in classical EVT.

3.9.4 Transformation

To find the optimal solution to the multi-variate, multi-modal problems, the extreme value distribution (EVD) should be deeply understood. Clifton et al. (2011) show that the EVD ${f^{\text{e}}}_d$ for a distribution ${f}_d$ follows the probability contours of ${f}_d$ distribution. This fact leads them to consider the EVD as a transformation of equiprobable contours on ${f}_d$. Additionally, they show that the equiprobable contours of the EVD ${f^{\text{e}}}_d$ occur at equiprobable contours of ${f}_d$. Thus, they consider the EVD to be a weighting function of

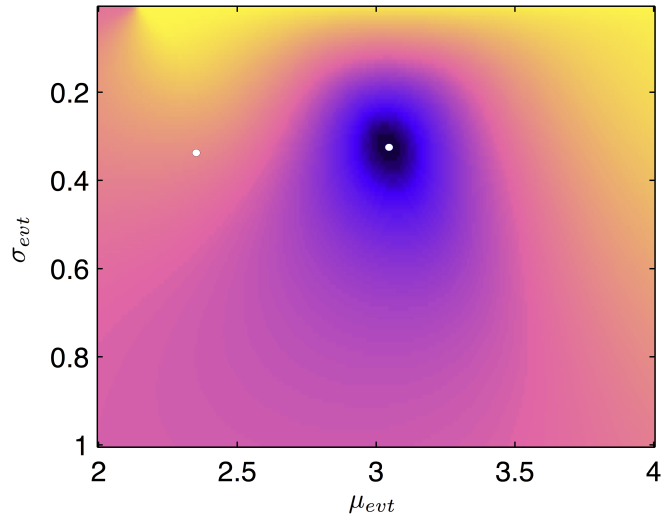


Figure 3.3: The error of using Eq. (3.12) and Eq. (3.13) in estimates of Gumbel parameters μ_{evt}, σ_{evt} for 2-dimensional data EVT. The MLE parameters located at $\mu_{\text{evt}} = 3.04$, $\sigma_{\text{evt}} = 0.32$, while the classical EVT estimates located at $\mu_{\text{evt}} = 2.37$, $\sigma_{\text{evt}} = 0.33$, which is clearly far from the desired MLE values. The figure is adapted from Clifton (2009)

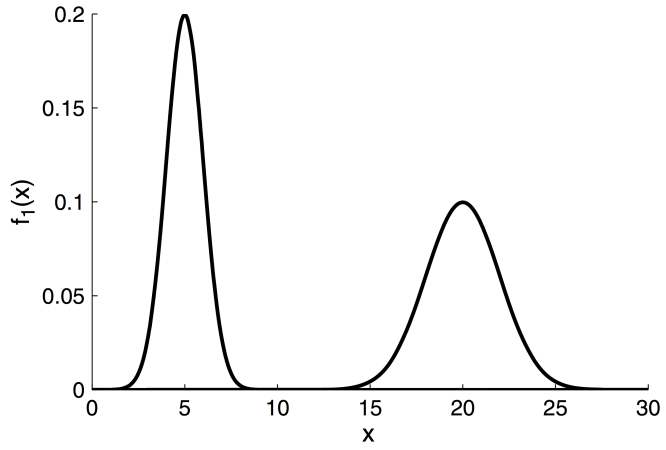


Figure 3.4: The classical definition of the extreme value is inappropriate for the multi-modal distributions. This figure shows a bimodal distribution f_1, where the region falls between the two modes should be considered in addition to the minimum and maximum values on the x-axis. The figure is adopted from Clifton (2009)

the contours of f_d,

$$f^{\mathrm{e}}{}_d(\vec{x}) = g[f_d(\vec{x})] \tag{3.15}$$

where g is a weighting function that defines the EVD regarding f_d. Thus, if the form of g is found then the $f^{\mathrm{e}}{}_d$ can be accurately determined for complex, multi-modal, multivariate distributions. Clifton et al. (2011) suggest a transform of the extreme $\vec{x}$,

$$\Psi[f_d(\vec{x})] = \begin{cases} (-2\ln(f_d(\vec{x})) - d\ln(2\pi))^{0.5} & \text{if } f_d(\vec{x}) < (2\pi)^{-d/2} \\ 0 & \text{if } f_d(\vec{x}) \geq (2\pi)^{-d/2} \end{cases} \tag{3.16}$$

This transformation represents the Mahalanobis radius of the standard unimodal Gaussian distribution between $\vec{x}$ and the mean of the distribution $\vec{\mu}$. The Ψ-transform thus maps the probability density values $f_d(\vec{x})$ back into a Mahalanobis distance space. This allows for the estimation of a Gumbel distribution. This procedure is valid for multi-modal distributions because $f_d(\vec{x})$ for extrema are distributed similarly for mixtures of negative exponentials of varying number of kernels, priors, and covariances (Clifton et al., 2011; Clifton, 2009). Therefore, multivariate, multi-modal EVD problems are successfully solved by evaluating all samples $\vec{x}$ in the original data space $\mathcal{D} = \mathbb{R}^d$ concerning $f^{\mathrm{e}}{}_d(\Psi[f_d(\vec{x})])$. This transform reduces the EVD analysis in multivariate data space to a simpler, but equivalent, problem in univariate probability space.

The novelty threshold at some contour $f_d(\vec{x}) = k$, which results from the EVD of the Ψ-transform is set based on the probability that generating N samples from f_d exceeds the threshold is given by $1 - F^{\mathrm{e}}{}_d(k) = 1 - 0.99 = 0.01$. It thus has a valid probabilistic interpretation.

In general, the Ψ-transform is a numerical method. It requires sampling of extrema from f_d then applying the MLE Gumbel distribution on the output of the Ψ-transformation. Clifton et al. (2011), however, suggest closed-form solutions (or approximations) for multivariate, unimodal distributions f_d called multivariate EVT (mEVT). They used the normal Gaussian distribution form as the probability density function:

$$f_d(\vec{x}) = \frac{1}{C_d} \exp\left(-\frac{M(\vec{x})^2}{2}\right) \tag{3.17}$$

where the dimensionality is $d \in \mathcal{N}$, $M(\vec{x}) = ((\vec{x} - \vec{\mu})^T \boldsymbol{\Sigma}^{-1}(\vec{x} - \vec{\mu}))^{1/2}$ is the Mahalanobis distance, and $C_d = (2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}$ is the normalisation coefficient. $\vec{\mu}$ and $\boldsymbol{\Sigma}$ are the centre and the covariance matrix, respectively. The probability space that is associated with the data space $\mathcal{D} = \mathbb{R}^d$ is $\mathcal{P} = f_d(\mathcal{D}) =]0, \frac{1}{C_d}]$. They proved that sampling in the probability space is equivalent to sampling in the data space. Thus, the time-consuming operation of sampling the extrema in the multivariate data space is solved by sampling in the uni-variate probability space.

If $\mathcal{X}$ is a random variable distributed according to $F^{\mathrm{e}}{}_d$, the form of the distribution function (df) G_d according to which $f_d(\mathcal{X})$ is distributed on $\mathcal{P}$, should be determined.

In other words, the probability distribution over probability density values should be computed. The details of the derivation and determining G_d can be found in (Clifton et al., 2011).

3.9.5 Parameters Estimation

Clifton et al. (2011) proved that G_d is in the maximum domain of attraction of the minimal Weibull distribution ($\mathcal{H}_3^-$) for all values of d. Therefore, the minimal Weibull parameters are adopted to estimate the EVT parameters:

$$\sigma_{\text{evt}} = 0, \quad \alpha_{\text{evt}} = 1, \tag{3.18}$$

$$\mu_{\text{evt}} = G_d^{\leftarrow}\left(\frac{1}{d}\right). \tag{3.19}$$

Eq. (3.19) computes the scale parameter μ_{evt} analytically with accurate approximation as compared to the values that obtained by using the MLE. Theoretically, the shape parameter α_{evt} should tend to 1 when the limit $d \to \text{inf}$. However, it seems to be decreased as the dimensionality of the data space increases, and it overestimated even for large values of d. Depending on the properties of the class equivalence of $\mathcal{H}_3^-$, the formula below is used to compensate the value of the α_{evt} (Clifton et al., 2011):

$$\alpha_{\text{evt}} = \mu_{\text{evt}} \frac{g_d(\mu_{\text{evt}})}{G_d(\mu_{\text{evt}})} \tag{3.20}$$

where $g_d(\mu_{\text{evt}})$ is the derivative of $G_d(\mu_{\text{evt}})$. Finally, the EVD of G_d is:

$$G_d^e(y_{\text{e}}) = 1 - \exp(-(y_{\text{e}}/\mu_{\text{evt}})^{\alpha_{\text{evt}}}), \quad y_{\text{e}} = f_d(\vec{x}) \tag{3.21}$$

$G_d^e(y_{\text{e}})$ gives the probability of drawing an extremum of lower probability, hence the probability of drawing an extremum of higher probability is $1-GG_d^e(y_{\text{e}})$ and the extremum is abnormal with probability $1 - G_d^e(y_{\text{e}})$. Hence the novelty score given as:

$$F^{\text{e}}{}_d(y_{\text{e}}) = 1 - G_d^e(y_{\text{e}}) = \exp(-(y_{\text{e}}/\mu_{\text{evt}})^{\alpha_{\text{evt}}}). \tag{3.22}$$

and in case of Eq. (3.17), the novelty score will be

$$F^{\text{e}}{}_d(\vec{x}) = \exp\left[-\left(\frac{1}{C_d \mu_{\text{evt}}} \exp(-\frac{M(\vec{x})^2}{2})\right)^{\alpha_{\text{evt}}}\right]. \tag{3.23}$$

Since our methods are based on the minimisation of the squared residuals, the residuals follow a normal distribution. Dividing the residuals by the root-mean-squared error yields a normal distribution with zero mean and unit variance. Therefore, the normalised residuals are interpreted as the Mahalanobis distance and thus are a replacement of the Ψ-transform and we directly use them to estimate the EVD Eq. (3.17).

3.9.6 Confidence bands

Models which are adapted to noisy measurements differ for each acquired set of data. The confidence bands are a measure or the probability that encloses all models derived from limited or noisy data (Kardaun, 2005). In other words, they are the set of intervals at which the true model is possible to lie with a probability of $1 - \alpha_{\text{conf}}$. The α_{conf} is, commonly, set to 0.05. Thus, the probability of enclosing the actual model by the confidence bands is 95%. They are estimated for linear equations as a measure of how well the predicted function fits the distribution (e.g. Kendall et al. (2007)). The output of the polynomial classifier (cf. Section 6.3) and the extreme learning machine classifier (cf. Section 6.1) are a weighted linear combination of some variables (the polynomial basis features or the hidden layer activations, respectively), where the weights represent the parameters of a linear model (Huang et al., 2006b). Hence, the confidence bands are computed for these proposed classifiers and used in the novelty detection. The confidence bands are used in the polynomial classifier to detect the outliers by Sakic (2012). Based on the general derivation of the confidence bands of a linear multivariate regression function given by Kardaun (2005), the confidence band $\eta_{\text{conf}}(\vec{t})$ of the classifier output $\vec{t}$ for a test sample $\vec{x}$ can be estimated according to

$$\eta_{\text{conf}}(\vec{t}) = t_{\text{v},\alpha_{\text{conf}}} \sqrt{\vec{t}^{\,T} (\boldsymbol{X}^T \boldsymbol{X})^{-1} \vec{t}} \sqrt{\Sigma_i^N r_i^2 / v}, \tag{3.24}$$

The $\boldsymbol{X}$ represents the whole vectors used in the linear computation of the training. Where r_i is the difference between the estimated and the target values i.e. the residual of sample i, $v = N - N_{\text{p}}$ is the number of degrees of freedom, with N_{p} as the number of free parameters of the model, $t_{\text{v},\alpha_{\text{conf}}}$ is the critical value of the t-distribution which depends on v and the probability threshold α_{conf}. The number of free parameters N_{p} is equal to the polynomial variables or the hidden neurons number, respectively. The standard value 0.05 of α_{conf} is used in all implementation. The full mathematical expressions are provided by Kardaun (2005).

4

Datasets and Feature Extraction

We have acquired a dataset which comprises 3D trajectories performed with a single hand that is used to evaluate the proposed algorithms. In addition to the captured gesture dataset, several datasets are used to evaluate the proposed methods. Since, the proposed algorithms achieve several tasks e.g. incremental learning, novelty detection and classify the non-linearly separable data, the proposed gesture dataset may not enough to evaluate all of these features. Some other datasets are used to evaluates the methods regarding specific features e.g. Iris dataset is used to evaluate the accuracy of the proposed algorithms within non-linearly distributed data. Additionally, different features set is computed to represent the same gesture data, which may give different characteristics to the same data.

This chapter has been adapted and/or adopted from: (Al-Behadili et al., 2014, 2015b, 2016b)

4.1 Gesture Data Set

A well-known database of gestures acquired with the Kinect sensor is described in Fothergill et al. (2012). These gestures, however, are mainly performed simultaneously with both hands. The database by Richarz and Fink (2011) comprises emblematic gestures performed with a single hand and the forearm. The 3D trajectories of these gestures, however, were computed from stereo images of two asynchronous cameras. In order to develop a classification system that copes with 3D trajectories but avoids the additional complexity of two asynchronous cameras or two-arm gestures, we have acquired a dataset which comprises 3D trajectories performed with a single hand[1]. The dataset was previously published in German language (Al-Behadili et al., 2014). The Microsoft Kinect sensor is used to acquire the gestures. The publicly available Kinect for Windows Software Development Kit (SDK) version 1.7 (Microsoft, 2013b) is used to control the Kinect sensor.

[1]The complete data set is available at http://www.bv.e-technik.tu-dortmund.de

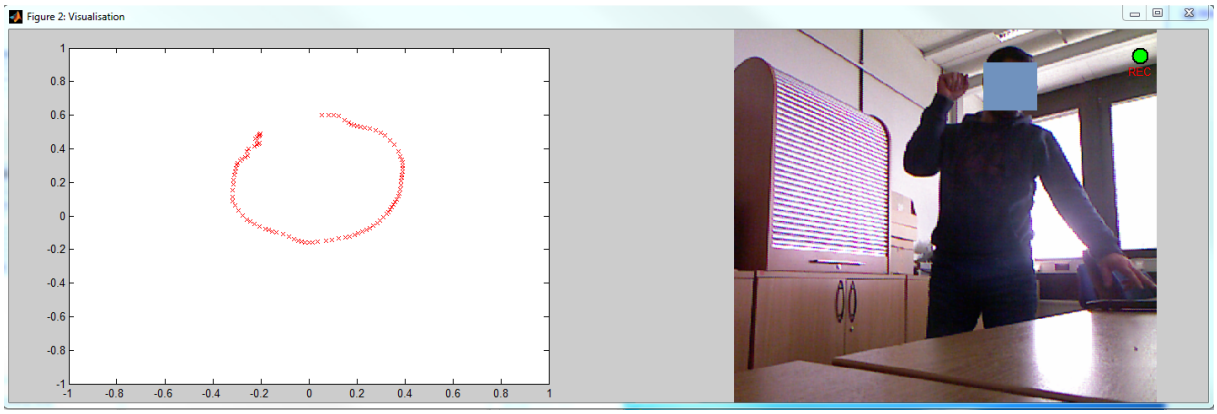


Figure 4.1: Screenshot of the gesture acquisition program.

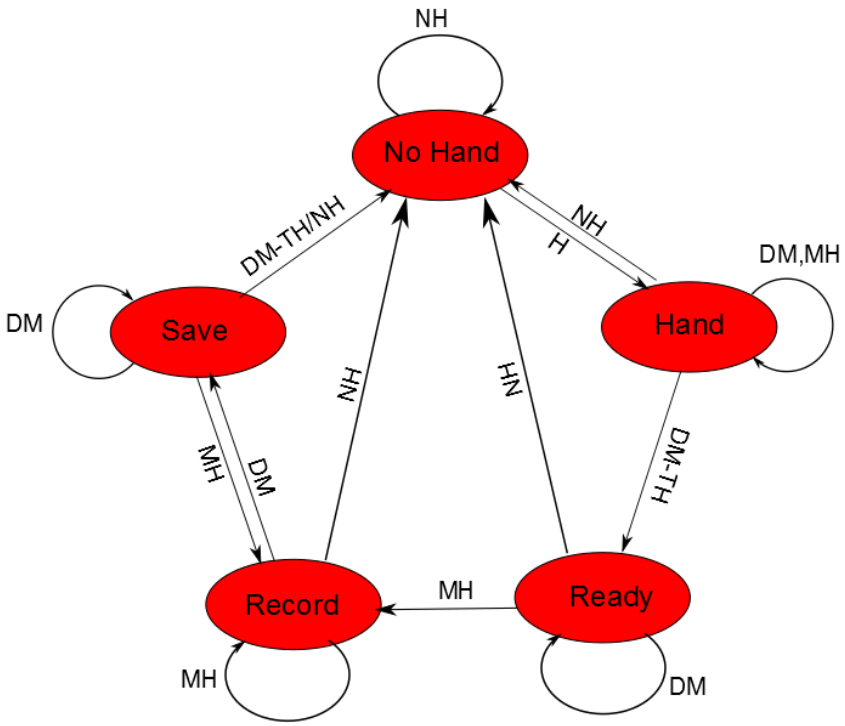


Figure 4.2: Flowchart of the gesture recording algorithm using the spotting algorithm (Yoon et al., 2001; Hoste et al., 2013). The symbols are described in Table 4.1. At the start, no hand is visible and the algorithm thus starts in the state “No Hand”.

Table 4.1: Explanation of the symbols used in Fig. 4.2

Symbol	Explanation	Action
NH	No hand is visible	The algorithm lost the hand, aborts and moves back to the initial state "No Hand"
H	A hand has been detected	Move to the state "hand" and wait till the hand does not move.
DM	The detected hand does not move	It is used for the spotting in the start and the end of the gesture. If the current state is "Hand", the algorithm goes to the state "Ready" if the hand does not move for a specified period. Otherwise, the algorithm is still waiting. If the current state is "Record", the recording stops if the hand does not move for a specified period, i.e. the algorithm activates the state "Save" otherwise the algorithm is still recording.
DM-TH	The threshold duration of the spotting	If a gesture has been recorded, i.e. the current state is "Save", the algorithm terminates by moving back to the initial state "No Hand" and thus prepares for the next gesture. If the hand has been localised for a new gesture, the algorithm moves to the state "Ready" and will record a new gesture starting from the next movement of the hand.
MH	The detected hand is moving	If a gesture is being recorded, i.e. the current state is one of the states "Ready", "Record" or "Save", the algorithm records the movement. Otherwise, it waits for the hand to stand still before recording a new gesture.

The SDK configures the system so that all the outputs, the RGB colour image, the depth image and the 3D coordinates of the skeletal joints of the person standing in front of the sensor, are synchronised as described in Han et al. (2013) and Microsoft (2013b). For completeness, this section describes the data acquisition and the extraction of features from the 3D hand trajectories. A screen-shot of the gesture acquiring program is shown in Fig. 4.1

Data acquisition The gesture dataset consists of 3D hand trajectories that were performed by ten persons and recorded using the Kinect sensor. Each person performed ten

gestures of each class with the left hand and another ten gestures with the right hand, respectively. Each gesture from these ten gestures has one pre-stroke and one post-stroke, respectively; but it has three strokes without rest position in between these three consequent strokes. The gesture recording is started by standstill the hand for a while and standstill the hand again to end the recording. Thus, the hand should not be stopped during the gesture or between the three consequent strokes, unless the person wants to end the gesture. The flowchart Fig. 4.2 explains this process and Table 4.1 summarises the symbols of the flowchart. The hand standstill is used to segment the gestures using the spotting idea (Yoon et al., 2001; Hoste et al., 2013). This algorithm removes random movements occurring before and after the actual gesture. Our system starts acquiring the gesture after the hand of the user appears in front of the Kinect sensor and has been idle for several seconds. The acquisition is terminated with the beginning of a second idle state of a certain duration.

The nine classes (i.e. different movement patterns) were adopted from Richarz and Fink (2011) and Schumacher et al. (2012). They are comprised of the emblematic gestures "circle", "point", "stop", "come here", "go away", "up", "down", "wave", and "wave vertically". These gestures appear natural to humans and may be used to command (mobile) robots or other intelligent systems that require user interaction. Thus, the total number of the gestures is 10 persons $\times$10 repetitions $\times$2 hands $\times$9 classes = 1800 gestures; and 1800 $\times$3 Strokes = 5400 strokes. Some of these gestures are destroyed due to the limited area where the gestures are recorded, hence they manually removed. The remaining gesture data set contains 1662 gesture (1662$\times$3 strokes). These data are used to evaluate the algorithm explained in (cf. Section 8.3). To simulate the online streaming, we wrote a code to spot only one stroke. This code depends on the direction of the movement, velocity and acceleration, shape of the trajectory, angles and orientation. The last data set is spotted by using this code. The obtained dataset contains 2878 gestures in total that are subdivided into nine classes. The distribution of remaining gestures is shown in Table 4.2.

Features used for classification Each gesture is represented by the sampled 3D trajectory of the moving wrist, from which eight features are derived for each sample. At first, the 3D coordinate of each sampled wrist position is recalculated with respect the point between shoulders, i.e. the 3D coordinate of the point between the shoulders is subtracted from each wrist coordinate, to eliminate the effect of the difference in person's heights and their location when they did the gestures.

Since each class of gestures is typically performed at a different position in space relative to the body of the person, the first three features are the average 3D wrist positions of the whole gesture. A further spatial normalisation is obtained by subtraction of the mean value from all trajectory points and dividing the result by the maximum distance, i.e. the Euclidean norm between the mean and these points. The second set of

Person	up		down		come here		go away		stop		point		circle		wave normal		wave vertical		Total /person
	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	
P - 1	30	30	36	36	23	37	36	38	22	18	23	22	10	10	11	11	12	12	417
P - 2	30	30	30	30	0	0	30	30	10	9	29	28	10	10	10	10	14	11	321
P - 3	33	30	27	30	0	0	29	33	9	8	28	30	10	11	10	10	10	10	318
P - 4	33	36	33	33	0	0	26	22	11	11	20	22	8	11	11	10	11	11	309
P - 5	28	26	23	26	23	12	20	12	10	10	24	29	10	10	10	10	10	10	303
P - 6	27	23	21	22	25	30	0	0	10	7	26	27	9	10	9	10	10	10	276
P - 7	29	29	33	30	10	0	21	11	9	10	10	14	10	10	10	10	10	10	266
P - 8	24	22	20	20	13	18	0	0	9	3	11	9	10	10	10	9	10	11	209
P - 9	6	1	7	2	0	0	18	20	10	10	26	27	10	12	10	10	10	10	189
P - 10	29	0	22	20	6	0	3	4	10	10	16	15	17	9	0	0	0	0	161
Unkown	0	0	0	0	0	0	0	0	11	9	28	33	2	3	12	11	0	0	109
Total/hand	269	227	252	249	100	97	183	170	121	105	241	256	106	106	103	101	97	95	
Total	496		501		197		353		226		497		212		204		192		2878

Table 4.2: Gesture Dataset: Amount of recorded gestures per person. L: left-handed execution. R: right-handed execution.

three features are the extensions of the gesture in x, y and z directions, i.e. the difference between the maximum and the minimum value. The seventh feature which is calculated from the direction of movement:

1. The principal components of the 3D trajectory are computed and analysed. Let λ_1 and λ_2 be the largest and the second largest eigenvalues of the covariance matrix of the 3D coordinates, respectively. If $\lambda_2 > 0.6\lambda_1$ the gesture is considered a two axes gesture. Otherwise, the gesture is considered a one axis gesture. In the former case, we keep the first two principal components, and in the latter case, we keep only the first principal component of the remaining analysis.

2. The 3D coordinates are projected onto the selected principal components, and the sign of each projected coordinate is computed.

3. Based on the amount of positive and negative values, we assign the following value to each principal component, respectively. We assign a value of 1 and two if more than 80% of the coordinates are positive or negative, respectively. Otherwise, the gesture has no predominant direction and is assigned a value of 3. Furthermore, a value of 0 is assigned to principal components that were not selected in the first step.

4. The three direction values are then interpreted as a base four digit and the corresponding decimal representation is computed to combine all directions in one value.

Finally, the last feature represents the total length of the normalised gesture. This feature set has been chosen after many experiments with other, more extensive feature sets including position, speed, direction, orientation, curvature, chain code, etc. (e.g. Bhuyan et al., 2008; Yoon et al., 2001). The compact representation of only eight features results in a low computational complexity of the algorithm and thus helps in the context of online learning.

4.1.1 Different features for the gesture dataset

Another set of features is computed to represent the data of 1662 gestures. This set of features is used in the evaluation experiments of Section 8.3. The first six features are computed exactly like the explained features in Section 4.1. The Dynamic Time Warping (DTW) algorithm (Müller, 2007, chap.4) is applied to each gesture to normalise it in the time domain. The DTW approach aims for warping a temporal sequence of data points such that it optimally corresponds to a reference sequence. The cost value corresponding to such a transformation can be viewed as a distance and depends on the sequence of the mutual assignments between the points of the sequences ("warping path"). It is minimised by dynamic programming (see Müller (2007, chap.4) for more details). Accordingly, the seventh feature is the minimum DTW distance between a gesture and the templates and the eighth feature is the label of the most similar template.

Dynamic time warping

The dynamic time warping algorithm is used in many applications especially in speech recognition (Rabiner and Juang, 1993; Bhuyan et al., 2008). It has the ability to measure the similarity between two sequences even they differ in time, speed, acceleration and length. It is clear that each person can't make an identical gesture in spatial-temporal domain even if he was insisted. Hence, the gestures for the same type that made by the same person never be identical rather than the gestures made by different persons. Even if we assume the gestures are identical, and we compute the similarity between them, we will get an inaccurate result if there is a time shift between the two gestures. Initially, the Dynamic time warping used in the speech recognition; hence, it was a one dimension algorithm. Thereafter, some researcher developed it for other applications to work with multi-dimensional data e.g. Gillian (2011). We implement the DTW to fit our data after projecting it on a 2D plane; i.e a two-dimensional series.

According to Müller (2007, chap.4), the DTW distance between two sequences $\vec{x} = {x_1, x_2, \ldots, x_N}^T$ and $\vec{x}' = {x'_1, x'_2, \ldots, x'_{N'}}^T$ is computed by finding the optimum warping path $\vec{w} = {\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_z}^T$ so that:

$$\max(N, N') \leq z < (N + N') \tag{4.1}$$

where the k_{th} value of $\vec{w}$ is given by:

$$\vec{w}_k = (i, j)_k \quad \text{where} \quad i \in \{1, 2, \cdots, N\} \quad \text{and} \quad j \in \{1, 2, \cdots, N'\} \tag{4.2}$$

The warping path should satisfy the following constraints:

- The warping path starts at $w_1 = (1, 1)$ and end at $w_z = (N, N')$.
- Monotonicity, which means the warping path should not move backwards.
- Continuity that's mean if $w_k = (i, j)$ then $w_{(k+1)}$ must be either $(i, j), (i+1, j), (i, j+1)$ or $(i + 1, j + 1)$.

Only the warping path that gives a minimum total warping cost c_W is selected among other paths, which may be satisfied the warping path conditions also. The minimum total warping cost c_W is given by:

$$c_W(\vec{x}, \vec{x}') = \min \frac{1}{k} \sum_{l=1}^{k} d_{\text{Eucl}}(x_{il}, x'_{jl}) \tag{4.3}$$

$d_{\text{Eucl}}(x_{il}, x'_{jl})$ is equal to the Euclidian distance (other types of distances cab used) between the x_{il} and x'_{jl} features, given by w_l. In order to compute this path, the dynamic programming is used to create an $N \times N'$ cost matrix $\boldsymbol{C}_{\text{W}}$ that contains the accumulated

minimum warping cost up to the position of that cell. The element of cost matrix is hence given by:

$$\boldsymbol{C}_{\mathrm{W}}(i,j) = d_{\mathrm{Eucl}}(i,j) + \min\{\boldsymbol{C}_{\mathrm{W}}(i-1,j), \boldsymbol{C}_{\mathrm{W}}(i,j-1), \boldsymbol{C}_{\mathrm{W}}(i-1,j-1)\} \tag{4.4}$$

The $\boldsymbol{C}_{\mathrm{W}}(i,j)$ represent the distance between the ith feature in the vector $\vec{x}$ and the jth feature in the vector $\vec{x}'$, plus the minimum value in the previous three neighbour cells. The minimum distance is then obtained by finding the optimum path by starting from the end point $\boldsymbol{C}_{\mathrm{W}}(N,N')$ back to the first point $\boldsymbol{C}_{\mathrm{W}}(1,1)$ going through the minimum value among the left, below and diagonally adjacent cells with respect to the current cell as illustrated in Fig. 4.3. Finding the optimum path is based on using the optimisation methods of dynamic programming (Müller, 2007). The early discussion of the optimisation method of dynamic programming can be found in (Bellman and Kalaba, 1959). The DTW is used to normalize the gesture data in the experiments Section 8.3 and it explained in Section 4.1.1. The final distance (DTW distance) between the two vectors is the sum of the cell's values of the cost matrix $\boldsymbol{C}_{\mathrm{W}}$ that be selected in the optimal path.

4.2 Artificial data set

To evaluate the methods on large datasets, we randomly draw samples from a multivariate normal distribution using the methods of the PRToolbox (Tax, 2015). The resulting dataset is comprised of a total of 9000 6-dimensional samples from three classes. The three centroids are positioned at $[1,-1,-2,-1,-1,2]^T$, $[0,0,0,0,0,0]^T$ and $[2,1,-1,-2,1,1]^T$, respectively. The covariance matrices of the three classes are diagonal matrices. The main diagonal of each covariance matrix, respectively, is initialized to $\frac{1}{2.5}$ times the minimum absolute difference between the corresponding class centroid and the other centroids. The absolute difference is computed for each coordinate, respectively, and increased until the random sampling produces an overlap of the three classes. The overlap is measured by the Mahalanobis distance. At each centroid, respectively, a hypersphere is constructed using a radius of the maximum Mahalanobis distance of the class members, respectively. The covariance matrices are then modified until the hypersphere of each class, respectively, contains as many samples from other classes as class members. All covariance matrices are divided by a factor of 1.1 if the overlap was too large, i.e. more samples of other classes than class members reside within the maximum Mahalanobis distance of the class. If the overlap was too small, in contrast, a factor of 1.1 is multiplied to all covariance matrices to increase the spread of the samples.

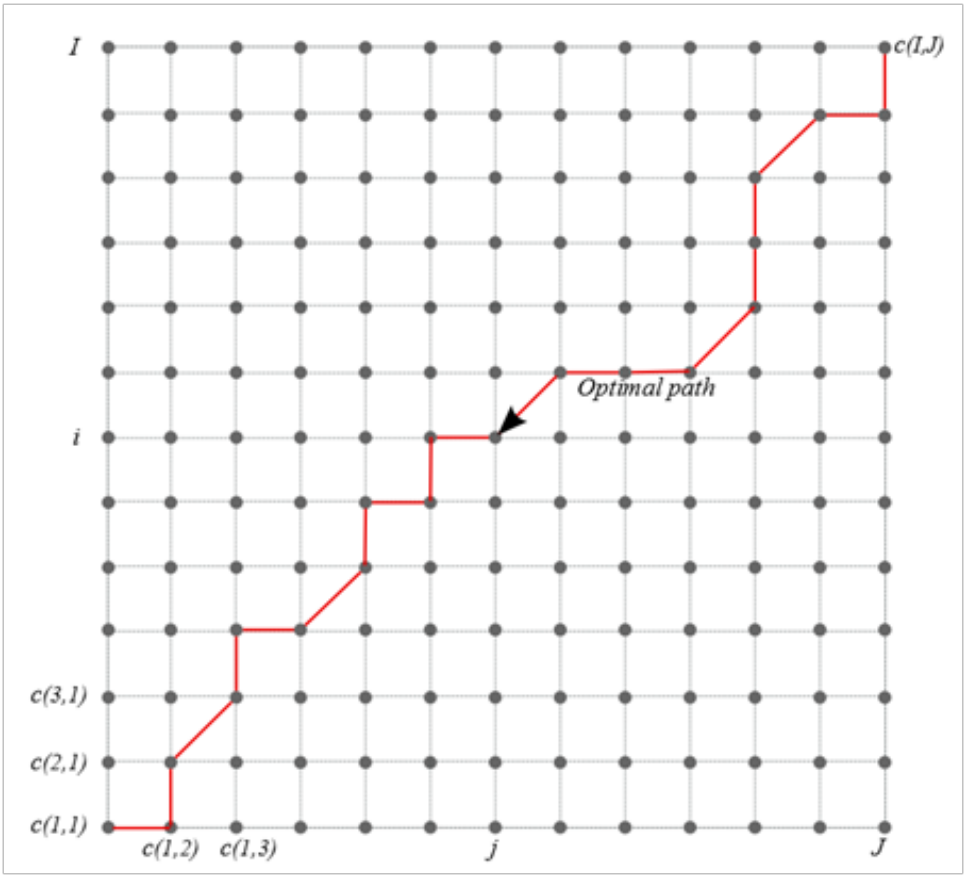


Figure 4.3: Finding the optimal warping path of DTW that gives the minimum distance between two series

4.3 Iris dataset

The Iris dataset introduced in (Fisher and Tippett, 1928)[2] consists of three classes. One class of the dataset is not linearly separable from the other classes. Consequently, it is used to estimate the ability of some proposed algorithms to separate non-linear distributions. The rather small Iris dataset is comprised of 150 samples in total, i.e. 50 samples of each class. Due to the number of samples limitation in the dataset, it is difficult to show the difference in the runtime for the incremental experiment. Therefore, the evaluation is restricted to separating the classes.

[2]The data can be downloaded from https://archive.ics.uci.edu/ml/datasets/Iris/

4.4 Lunar data set

The Jet Propulsion Laboratory designed the instrument of the Moon Mineralogy Mapper (M^3) accurately to provide geologic context. The solar radiation reflected from the lunar surface is measured and diagnosed by M^3 to find mineral absorption bands. For most of the M^3 data set, the spatial resolution corresponds to 140 m per pixel in 85 spectral channels (Pieters et al., 2011).

Section 4.4.1 and Section 4.4.2 describe the datasets used for the training and the evaluation of the automatic segmentation algorithm, respectively. In order to reduce the dimensionality of the spectral data, we infer the spectral parameters, which are defined in Section 4.4.3, from the hyperspectral dataset.

4.4.1 Near-global mosaic

In order to derive the most common lunar spectra, we use a low-resolution near-global mosaic. The near-global mosaic is created by downscaling the M^3 data and the GLD100 (Scholten et al., 2012) to a resolution of 20 pixels per degree longitude and latitude, respectively corresponding to about 1.5 km per pixel, which is the smallest size of surface details that are fully covered by the GLD100 (Scholten et al., 2012). The digital elevation model (DEM) is required to compensate the influence of the local topography and thus the best possible resolution without topographic artefacts is the scale of the smallest fully visible surface details. We then applied the method of Grumpe et al. (2015) to normalize the M^3 data to standard geometry, i.e. an incidence and a phase angle of 30 °, respectively, and an emission angle of 0 ° (Pieters, 1999), using the Hapke model (Hapke, 2002, 1984). Based on the M^3 reflectance data and the GLD100 topographic model, the single-scattering albedo is inferred. Using the standard illumination conditions and the inferred single-scattering albedo, the resulting normalized reflectance is computed. The resulting near-global reflectance mosaic is shown in Fig. 4.4.

Based on the normalized reflectance, we constructed maps of the spectral parameters (see Section 4.4.3) and used a polynomial regression, which is similar to the regression models of Wöhler et al. (2011) and Shkuratov et al. (2005) applied to Clementine data, to map the downscaled version of the spectral parameter maps, i.e. 60 km per pixel, to the Lunar Prospector Gamma-Ray Spectrometer elemental abundance data (Lawrence et al., 1998). Using the estimated polynomials, we construct full resolution near-global elemental abundance maps.

To reduce the noise and limit our analysis to the most common lunar reflectance spectra, we downscale all maps to a resolution of two pixels per degree. The resulting elemental abundance maps are then clustered using the algorithm of Grumpe and Wöhler (2014b), i.e. a self-organizing map (Kohonen, 2001) is used to cluster the pixels according to their mineral abundance values and the number of clusters is iteratively increased or decreased based on the correlation of their median reflectance spectra. Fig. 4.5 shows the

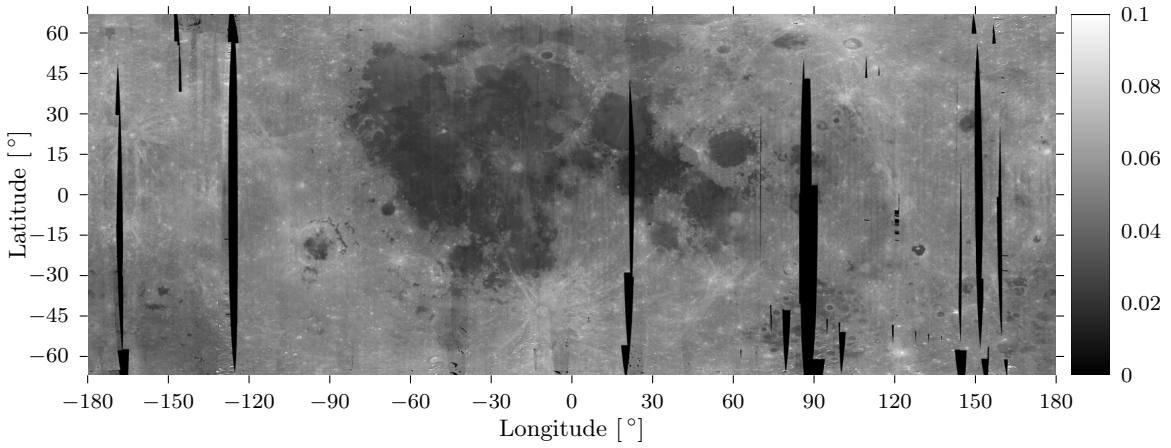


Figure 4.4: The near-global normalized reflectance mosaic derived from the M^3 channel centred at 1579 nm, constructed using the framework of (Wöhler et al., 2014).

distribution of the resulting ten clusters. These data are used to train both, the ELM predicting the cluster label and the AE-ELM (cf. Section 6.2), which detects novelties.

4.4.2 Region of interest

To evaluate the proposed algorithm, we applied it to the western Moscoviense basin (24 ° N–30 ° N, 143 ° E–145 ° E), which was also studied by Pieters et al. (2011). We select the M^3 images M3G20081229T101650 and M3G20090125T172601 covering almost the whole region. The M^3 data are mapped to a cylindrical grid where the horizontal axis corresponds to the selenographic longitude, and the vertical axis corresponds to the selenographic latitude. The spatial resolution of the grid is set to 300 pixels per degree, which corresponds to about 100 m per pixel. Since the GLD100 does not cover all small-scale surface details at this level, we apply the method by Grumpe and Wöhler (2014a) to refine the GLD100 using the M^3 channel centred at 1579 nm. This channel was selected because the target wavelength is weakly affected by the spectral absorption bands and still is sufficiently low and thus not subject to thermal emission. We then estimate the thermal emission component and normalize the M^3 data to standard geometry using the method of Grumpe et al. (2015) and infer the spectral parameters introduced in Section 4.4.3.

4.4.3 Spectral parameters

At first, the normalised reflectance spectra are smoothed along the wavelength axis applying the smoothing spline. This smoothing method simultaneously minimizes the mean squared of both the second derivative and the deviation from the reflectance (Marsland,

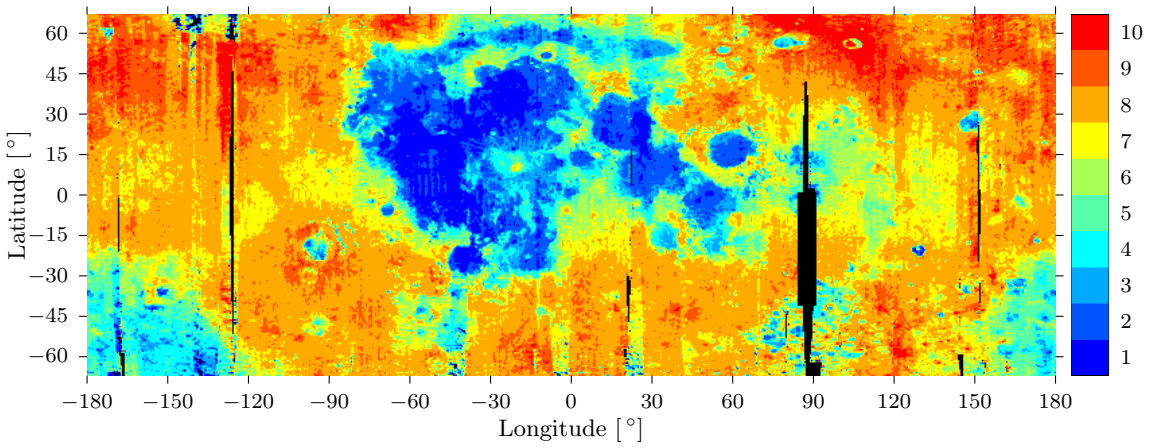


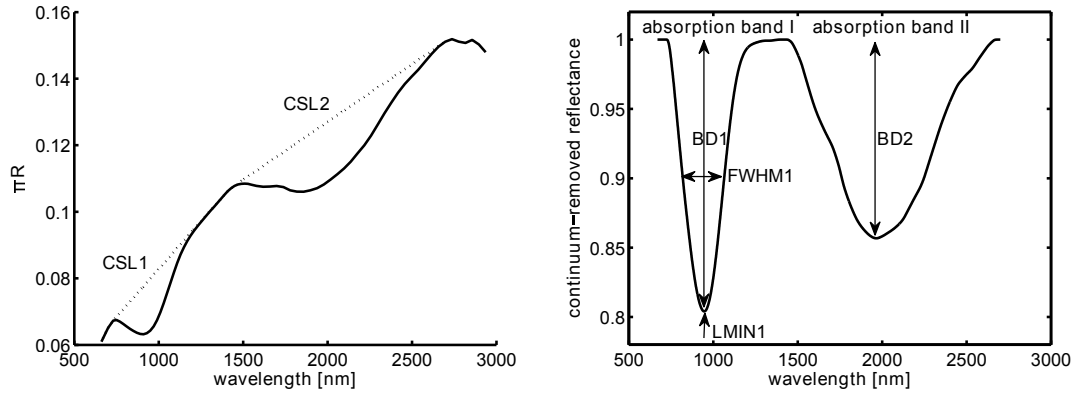
Figure 4.5: The spatial distribution of the determined clusters. Due to the properties of the self-organizing map, neighbouring clusters have similar spectra. Consequently, the lower cluster labels correspond to mare basalt while the higher cluster labels correspond to highland. The cluster labels in between occur at the boundary between mare and highland regions and represent a mixture of both. Black denotes missing data. Image from Grumpe and Wöhler (2014b).

2009). Following Akima (1970), the resulting smoothed spectrum is interpolated to integer wavelength values in nm. The continuum-removed spectrum (Fu et al., 2007) is then obtained as the ratio of the interpolated spectrum over the convex hull of the smoothed spectrum.

Lunar reflectance spectra are typically characterized by two absorption bands centred near 1000 nm and 2000 nm, termed "band I" and "band II", respectively (Burns et al., 1972; Adams, 1975). Consequently, Wöhler et al. (2014) deduced several spectral parameters from the continuum-removed spectrum, which are shown in Fig. 4.6(b). These parameters are the wavelength of the bandcenter wavelength (LMIN1), the relative band depth (BD1) and the full width at half maximum (FWHM1) of band I. The slope of the convex hull is used to infer the slopes of the continuum (CSL1) Fig. 4.6(a). Additionally, bandcenter wavelength (LMIN2), the band depth (BD2), the full width at half maximum (FWHM2), and the continuum slope (CSL2) of absorption band II are used as spectral parameters. The logarithmic band depth ratio (LBD) is an important spectral parameter in the context of lunar impact melt (Wöhler et al., 2014):

$$\mathrm{LBD} = \log_{10} \frac{\mathrm{BD1}}{\mathrm{BD2}}. \tag{4.5}$$

Additionally, we compute the integrated band depths (IBD1 and IBD2) of band I and band II, respectively.



(a) (b)

Figure 4.6: (a) A typical lunar reflectance spectrum and its convex hull (dotted curve). (b) The continuum-removed spectrum with inferred spectral parameters. Image from Wöhler et al. (2014).

5

Contribution: Non-parametric Learning Based Semi-supervised Methods

Many machine learning algorithms are trained only using data with manually assigned labels, where all data should be available at the beginning of the training. It's hard to fulfil these two conditions of such traditional algorithms since most real-life data are streamed continuously as unlabelled data. Additionally, streamed data are often affected by drift, outliers, an emergence of new classes, and disappearance of others (see Section 3.6). An example of the concept drift, a recognition system in HCI is, commonly, trained on gestures, which are performed by a limited number of persons. Although the system is trained on all possible classes of the gestures, it often faces difficulties in the recognition of gestures which are performed by unknown "new" interaction partners. Since the commonly used training methods pre-suppose a manually labelled training data set, usually the interaction system cannot be adapted autonomously to new interaction partners during its operation. In many cases, this means the person is enforced to learn how the gestures must be performed such that they are recognised by the system.

Hence, semi-supervised learning can offer a solution for the scarcity of labelled data, concept-drift or the data stream problems altogether. One of the methods used in semi-supervised learning is that the classifier is trained using a small number of manually labelled training samples and updating the classifier on new samples with the classifier assignment. The semi-supervised learning approach also known as "self-training" (Rosenberg et al., 2005). This method is adopted here for all proposed classifiers since it exactly fits the situation of streaming data. In order to achieve a satisfactory classification with semi-supervised learning, the other challenges should also be resolved, since training the classifier with wrongly labelled samples will decrease its accuracy. To avoid this problem, we used a novelty and outlier detection to exclude the samples that are dissimilar to the known classes. We call these samples "unknown" samples, which may belong to a novel concept or be an outlier. Furthermore, using all data for retraining the classifier is time-

consuming and needs considerable hardware resources. Hence, we used the incremental learning to update the classifier using only the new data.

The active learning solves a part of the data stream problems which is using all or some of the unlabelled data to retrain the classifier, but it still needs all the old data in the retraining process besides human interventions, Schumacher et al. (e.g. 2012). Consequently, the system is desired to be automatically updated on-line, i.e. the classifier is updated incrementally, without storing the old data, on the newly assigned data during its operation without human intervention. By adding the ability to distinguish real gestures from random movements, it is possible to update the classifier to new persons or new gestures fully autonomously. Furthermore, the usage of labels that are assigned by the classifier without human intervention requires that these labels are believable, i.e. the outputs of the classifier should have additional measures for the confidence.

Due to the extreme varieties of the gesture types as they were seen in Chapter 2, the gesture datasets are expected to vary from simple, which may be possible to be classified by using simple classifier e.g. Euclidean distance, to very complex and non-linearly separable data. Consequently, a single classifier type may not fit all the variations of the different gesture types. Thus, we extend the capabilities of various models of classifiers to cover most of the possible characteristics of the gesture data. These extensions include several features such: increment learning, increment class learning, novelty detection, multi-class, unbalanced data and some of the proposed classifiers fit non-linearly separable data.

The extended classifiers that have been adopted to work on the semi-supervised learning of the gesture are based on different classification techniques, which are the Parzen window, the support vector machine (SVM), the extreme learning machine (ELM), the polynomial classifier and several types of metric learning. The ability of the incremental learning, i.e. update the classifiers to the new samples without needs to store the old data, is added to all of these classifiers in combination with the ability to detect the samples that belong to "unknown" concept. The characteristics of the classifiers regarding these features are discussed in three chapters depending on the technique that the classifier is based. The Parzen window classifier and the SVM are non-parametric classifiers, which are addressed in this chapter. The ELM and the polynomial classifier use non-linear transformation to map the data to another space. Both of them are discussed in Chapter 6. Finally, the methods that are based on metric learning are discussed in Chapter 7.

Additionally, we enable the classifiers to add new classes to their knowledge incrementally without the need of retraining the classifiers from scratch. This feature is called as "incremental class learning", which is critical in case of streaming data since the incoming data may be originated from new concepts. Studying the characteristics of this feature is covered in the Chapter 8. Moreover, an ensemble classifier is built from the proposed classifiers to compensate the classifiers' features and to automate the artificial learning revolution. The proposed ensemble classifiers are discussed in details in the same chapter (Chapter 8).

Since the novelty detection is implemented in most of the proposed classifiers by using

the Extreme Value Theory (EVT) with some modifications, we will discuss it in a separate section, and the modification will be covered inside the section of the specified method. Additionally, the chapters (Chapter 5 – Chapter 7) will study the classifiers regarding incremental learning and novelty detection by using the same template of experiments but with different settings. Hence, the experiments template will be covered in a separate section in this chapter.

This chapter has been adapted and/or adopted from: (Al-Behadili et al., 2015f, 2016a,b)

5.1 Extreme Value Theory Implementation

The aim of using the Extreme Value Theory (EVT) is to implement the novelty detection by the incremental classifiers to use the streamed data in the semi-supervised scenario. The conventional methods use an additional data to set the novelty threshold. The classifiers are supposed to learn using the streamed data where a scarce of the labelled data and a concept-drift are available. However, using additional labelled data to set the threshold of the novelty decreases the usefulness of the proposed algorithms. Contrary, the extreme value theory (EVT) enables us simply to establish an accurate novelty threshold for the streamed data (Section 3.9.2). Changing of the classes distribution and the concept drift do not affect setting the novelty threshold of the EVT since it is updated automatically with the classifier updating. Furthermore, the additional labelled data that are used for validating the novelty threshold is not required and the threshold is unaffected by changing the number of classes. These features have made us include the EVT within the proposed classifiers. Unfortunately, the classical EVT (cf. Section 3.9.2) works only within univariate, unimodal data. Normally, the gesture data are multivariate and in most cases multi-modal. To solve this problem, we adopt the method of using the EVT with multivariate, multi-modal data proposed by Clifton et al. (2011). Instead of using the data themselves in the EVT, they map the data to a uni-variant, unimodal data by using a transformation function. They proposed a Gaussian function (Eq. (3.17)) to map the data and they apply its outputs to the general equation (Eq. (3.21)). Using the same manner, we used the output of the classification function of the proposed algorithms as a transform function, and we applied the outputs to the general equation (Eq. (3.21)). If we assume Ψ is the residual of the proposed classifiers outputs after subtracting the target values and dividing by the class-wise root-mean-squared error, then the $\vec{P}_{\text{evt}}$ is computed as

$$\vec{P}_{\text{evt}}(x) = \exp\left(-\left(\Psi(x)/\mu_{\text{evt}}\right)^{\alpha_{\text{evt}}}\right), \tag{5.1}$$

where the μ_{evt} and α_{evt} are the scale and shape parameters, respectively. They are computed by using the Eq. (3.19) and Eq. (3.20). Here, the exponent α_{evt} is applied element-wise to $\Psi(x)$.

The EVT output for the unknown sample is 1, and it is decreased proportionally with the departure toward the centre of the distribution where it equals zero. In most cases, EVT scores are computed in a class-wise manner i.e. the $\vec{P}_{\text{evt}}$ is a vector containing one score corresponding to each of the known classes; where the lowest score corresponds to the highly probable class. Thus, a test sample is labelled with the class that corresponds to the lowest score. Then, this score (P_{evt}) is compared to the novelty threshold P_{th}. The P_{th} represents the contour surrounding a specific fraction of the normal data equals to $P_{\text{th}} \times 100\%$ e.g. if we set $P_{\text{th}} = 0.95$, then we select the contour that contains 95% of the training data for that distribution. If the selected component from $\vec{P}_{\text{evt}}$ is larger than the P_{th}, then the novelty flag is set. Additionally, the believability flag is set if and only if the sample is not indicated as novel and another condition (may be more than one condition depending on the classifier type) is fulfilled.

5.2 Experimental Set-up

We have used more than one data set to evaluate the proposed classifiers. These datasets may differ in the characteristic of the data, the number of classes and samples, and the field of the data. The variety of the data is chosen to evaluate the proposed classifiers for different possible characteristics, which may be faced by using the data stream. Here, we will explain the general pattern of the experiment, and we will discuss the details through the classifiers sections. Commonly, the proposed classifier is evaluated in comparison with the original or similar classifier. In most experiments, the SVDD (cf. Section A.1) is used since it indicates novel samples and learns incrementally.

The total data samples of a particular dataset are randomly divided into three (in some cases four) disjoint data sets: the initial training set, the learning set, the independent test set, and in some cases validation set. For example, the gesture data set in the evaluation experiment of the polynomial classifier is divided into fractions of 25%, 50%, and 25%, respectively. Each class is split into the given fractions separately to ensure that all sets include all classes with distinct fractions. Since the classes in the original gesture database are unbalanced, i.e. they have different numbers of samples, each produced set will be unbalanced. Excluding the samples of one class from the initial training set simulates the novel class or outliers. Additionally, to emulate the data streaming, the learning data set is subdivided into "buckets", each of which consists of a fixed number of samples e.g. 100 samples in case of the gesture data set and five samples in the case of the Iris data set. Moreover, outliers (random movements) are added to the learning and testing sets in some experiments.

Each classifier is trained using the initial training set and adapted using the buckets of the learning set, respectively. They are evaluated by computing the accuracy and other measures based on the test set. The buckets of the learning set are sequentially presented to the classifiers to update them incrementally after removing the samples, which have

a low reliability. Since the classifiers have different outputs, they will generate different training sets. This process is continued until the last bucket in the learning set is classified. Two flags control the semi-supervised learning process. The believability flag to indicates trusty labels and the novelty flag to indicates unknown samples. Only trusty labels are used in the next training. The novelty flag is set if the condition(s) of the novelty indicate the sample as a novel, whereas the believability flag is set if it is not indicated as novel and it fulfils the believability condition.

As mentioned above, the training data sets of the classifiers may develop differently, i.e. they may have different sizes and may contain different samples or labels. This behaviour is because each classifier may indicate different samples as unknown and may assign different labels to the same sample. This diversity will lead to a different handling of the data and will have an effect on the processing time. In some experiment, both classifiers are trained with the correct labels during all iterations. This is to enforce them using the same samples in each updating to compare the processing time only rather than the accuracy. Because of the data are randomly divided, the whole experiment was repeated 100 times for each class, i.e. each class was considered a novel class in 100 runs, respectively. Notably, identical data sets were ensured for the compared classifiers during the total of $(N_{\text{class}} \times 100)$ runs, where N_{class} is the number of the classes. Since the data are multi-class, we cannot use the commonly applied sensitivity and specificity measures to evaluate the ability to detect novelties. Instead, we adopt the novelty detection metrics proposed by Masud et al. (2011) for this purpose. These metrics are M_{new}, F_{new}, and E_{total}, which represent the fraction of missed novel samples, the fraction of samples belonging to known classes indicated as a novel class by the classifier, and the total misclassification rate, respectively. These metrics are computed according to

$$M_{\text{new}} = F_{\text{n}}/N_{\text{novel}} \cdot 100 \tag{5.2}$$

$$F_{\text{new}} = F_{\text{p}}/(N_{\text{total}} - N_{\text{novel}}) \cdot 100 \tag{5.3}$$

$$E_{\text{total}} = (F_{\text{p}} + F_{\text{n}} + F_{\text{e}})/N_{\text{total}} \cdot 100 \tag{5.4}$$

The value of F_{n} is the number of the unknown samples that the classifier misses, which corresponds to the false negatives for one-class classifiers. N_{novel} represents the total number of the outliers within the total set of N_{total} presented samples. F_{p} is the number of existing class samples which are wrongly indicated as unknown by the classifier. It corresponds to the number of false positives for one-class classifiers. F_{e} is the number of samples belonging to a known class and assigned to an incorrectly known class. From (5.4) we notice that E_{total} is not necessary equal to the sum of M_{new} and F_{new} (Masud et al., 2011). Furthermore, we compare the total processing time to show that the proposed approach is suitable for on-line classifications.

Notably, in all experiments, the median, the 25% and the 75% quantile over the 100 runs is computed. In all figures, a solid line draws the median while the 25% and the 75%

quantile are drawn in dashed lines. In most of the time figures, both quantiles are not recognised Since the time is approximately similar in all the 100 runs.

5.3 Semi-Supervised Learning Using Parzen Window Kernel Density Estimators

This approach targets all problems of data streaming using a simple model that is insensitive to the size of the training dataset. The proposed approach differs from Jain and Nikovski (2008) by eliminating the time-consuming part of the algorithm that tries to find the least significant samples in the old training data set every time the classifier is updated. The proposed algorithm replaces the kernels located at the position of all training data with a specified number of centroids. These centroids are efficiently updated to handle the concept-drift and novelty detection while the constant number of centroids is maintained, and thus the problem of growing datasets does not occur.

The presented approach differs from the existing methods by several features. First, it can update itself incrementally, i.e. it needs fewer hardware resources and less time. Hence, it will work with data streaming and still efficiently with novelty detection. Secondly, it is a multi-class classifier, which can label many classes as well as the ability to indicate the outliers or novel classes. Finally, it responds to the concept-drift and it is robust to the incorrectly labelled samples since the effect of each sample is divided by the number of the samples which are members of the same cluster. The latter is important in semi-supervised learning where the correctness of a label is not guaranteed.

5.3.1 Parzen window kernel density estimators (PKDE)

The estimation of the probability density function for a specific distribution is crucial for most of the classification approaches. The kernel density estimator (KDE) (Lee and Roberts, 2008) is a non-parametric method. It is used to deduce the probability density function by locating kernels (mostly Gaussian) all over the data. Probably the most familiar method of such estimators is the Parzen window kernel density estimator (Parzen, 1962; Jain and Nikovski, 2008; Tarassenko et al., 1995; Bishop et al., 1995) in which the density function at any point in the data set is simply the linear combination of the neighbour kernels.

Here we follow the description of Clifton (2009). If we have N_c samples in the training data set that belong to the class c drawn independently from a class-conditional probability, $p(\vec{x}|c)$ which needs to be estimated, then amounts to

$$p(\vec{x}|c) = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{1}{\left(2\pi {h_w}^2\right)^{\frac{d}{2}}} e^{-\frac{\|\vec{x}-\vec{x}_i\|^2}{2{h_w}^2}}, \tag{5.5}$$

where h_{w} is Parzen window size and $\vec{x}_i$ is feature vector of the i-th training sample. The normalisation constant includes the width of the Parzen window and the dimension d of the feature vector. We use the PKDE implementation of Clifton (2007) who used PKDE as a one-class classifier. The width of the Parzen window is set as the average of the mean Euclidean distance over k neighbours of each sample in the training data. The value of k is a fraction of the total number of the training samples N_{c}, e.g. $k = N_{\mathrm{c}}/10$.

According to the Bayesian data analysis, the posterior $p(c|\vec{x})$, i.e. the probability of a sample belonging to class c if the feature vector $\vec{x}$ is observed, amounts to

$$p(c|\vec{x}) = \frac{p(\vec{x}|c)p(c)}{\sum_i p(\vec{x}|c_i)p(c_i)} \tag{5.6}$$

where $p(\vec{x}|c)$ is the class conditional-probability, which is also termed likelihood, and $p(c)$ is the prior, i.e. the probability that class c appears. A multi-class classifier detects the class with the maximum posterior (Tax and Duin, 1999). If the maximum posterior falls below a threshold, then the sample may be rejected or considered as a novelty (Chow, 1970). If the classes are equally distributed over all samples, i.e. the probability $p(c)$ is constant and equal for all classes c, the problem reduces to the computation of the maximum likelihood. PKDE is a prevalent method of non-parametric density estimation. The data size, however, grows with time in an online system, and since PDKE needs all old data in its calculations, it is not suitable for data streaming.

5.3.2 Incremental Parzen window kernel density estimators (IncPKDE)

As mentioned above, the critical point of PKDE is the growing dataset that needs to be maintained for the evaluation of new samples. Hence, we present a new technique to overcome this problem and to get an incremental Parzen window kernel density estimator (IncPKDE). Following Tax and Duin (1999), we update the PKDE of Clifton (2007) to work with multi-class systems by considering the sample that belongs to the class with the maximum likelihood assuming a constant prior probability of all classes. If the likelihood of all classes falls below a class-specific threshold, then it is considered as a novel sample.

Instead of accumulating all data of the stream in the estimator, we pre-cluster the initial training data to N_{cent} centroids for each class (Jeon and Landgrebe, 1994), where N_{cent} is chosen by the user, and update the centroids whenever new data arrive. Consequently, the data of the classifier are $N_{\mathrm{cent}} \times N_{\mathrm{class}}$ centroids at maximum with N_{class} as the number of classes.

In this work, $N_{\mathrm{cent}} = 20$ was selected. The clustering of the initial data set to N_{cent} clusters is achieved by applying the k-means algorithm (see Section 3.3.1) for each class, respectively. If the amount of data belonging to a class is less than N_{cent} samples, then we keep all the data of that class and place Parzen windows of width h_{w} at each sample.

Additionally, we save the number of samples that are replaced by each centroid, respectively. The width h_{w} is calculated here by averaging the distance of the $k = 3$ nearest neighbours from each centroid in the new training set.

First of all, when a new unlabelled sample is available, the system tries to get a label for it and then updates the training set with this new data. To assign the label to the new sample, we compute the likelihood concerning each class. The EVT scores $\vec{P}_{\mathrm{evt}}$ are computed for each class and compare it to the threshold P_{th}. The sample will be considered to belong to the class if its likelihood is inferior to the threshold. If the sample may belong to multiple classes, then we choose the class that has the maximum likelihood. In contrast, if it does not belong to any class then it will be considered as novel.

After the unlabelled sample has been successfully assigned to class during the first algorithm phase, the (initial) centroids of the corresponding classes are updated, respectively. Depending on the number of centroids in the (initial) training set N and the number of newly added samples N' one of the following techniques is applied:

$N + N' \leq N_{\mathrm{cent}}$: Since the joint set of the centroids and the new samples still does not contain N_{cent} centroids; each new sample will become a centroid.

$N < N_{\mathrm{cent}} \wedge N + N' > N_{\mathrm{cent}}$: The N is less than N_{cent} but the number of centroids would exceed N_{cent} if all samples $N + N'$ became centroids. Therefore, we apply the k-means algorithm to the joint set of the centroids and the new samples to reduce the number of centroids to N_{cent}. Notably, each centroid was formed using a single sample if $N < N_{\mathrm{cent}}$ and thus the clustering algorithm is applied to the full dataset.

$N > N_{\mathrm{cent}}$: This will be the standard case if the training dataset contains more than N_{cent} samples. In general, samples have been discarded and replaced by their centroid if the algorithm reaches this point. A re-clustering using k-means is thus neither feasible nor correct since the position of the centroid carries no information on the number of samples that were used to compute the position. Consequently, we update the centroids in a manner similar to the algorithm proposed by MacQueen et al. (1967), i.e. we update the centroid that is closed to the new sample. The following procedure maintains a high speed while it updates the position of the centroids and thus responds to the concept-drift.

To update the centroids, we first assign the new samples to the closest centroid. Let N_{cent_j} and N'_{cent_j} be a number of samples replaced by centroid j before the update and the amount of newly assigned samples, respectively. The new position of the centroid $\vec{\mu}_{\text{cent}}$ will be the point of gravity of all samples replaced by the centroid j. The new position thus updated

$$\vec{\mu}_{\text{cent}} \leftarrow (N_{\text{cent}_j} \cdot \vec{\mu}_{\text{cent}} + \sum_{i=1}^{N'_{\text{cent}_j}} \vec{x}_i)/(N_{\text{cent}_j} + N'_{\text{cent}_j}) \tag{5.7}$$

where $\vec{x}_i$ is the feature vector of the i-th new sample. To keep this update procedure consistent with the growing dataset, we adjust the number of samples replaced by the centroid

$$N_{\text{cent}_j} \leftarrow N_{\text{cent}_j} + N'_{\text{cent}_j}. \tag{5.8}$$

Using this technique, the size of the training data will never exceed N_{cent} centroids in each class, which results in a constant query time of the classifier and enables online classifications. Besides, the consecutive updates of the centroids respond to the concept-drift, the class-wise novelty calculations may give more accurate results. Finally, the centroids, which are determined by more than one sample, reduces the influence of a falsely assigned sample and the IncPKDE is thus is suitable for semi-supervised learning.

Novelty detection using EVT

By using the incremental version of the PKDE with additional labelled data to set the novelty threshold, we lost the advantage that we gained by incremental learning. Hence, we use the EVT for detecting the novelty. The output of the IncPKDE (Eq. (5.6)) is applied to the general equation (Eq. (5.1)) to compute the EVT scores. The target value here is a vector of length equal to the number of training classes. It contains a value equal to one in place corresponds to the class that has the highest probability of the Parzen outputs and zeros elsewhere. Initially, EVT scores are computed for each class i.e. the $\vec{P}_{\text{evt}}$ is a vector containing one score corresponding to each of the known classes. A test sample is labelled with the class that corresponds to the lowest score (P_{evt}). Then, this score P_{evt} is applied to the novelty detection threshold P_{th}. Here P_{th} is set to 0.95, which represents the contour that contains 95% of the training data of the class distribution. If it (P_{evt}) is larger than the P_{th} then novelty flag is set. The believability flag is set if and only if the sample is not indicated as novel and the difference between the smallest and second smallest scores in the vector $\vec{P}_{\text{evt}}$ is larger than a specific value. This value is set between zero and one and can be estimated using the known classes.

5.3.3 Experiments

Three different experiments are implemented to measure the performance of the IncPKDE. The IncPKDE's results are compared with the original PKDE and the SVDD classifiers. We used the SVDD from data description toolbox presented by Tax and Duin (1999) (cf. Section A.1) as both a multi-class novelty detection and an online classifier. Whereas, the PKDE is used from the Novelty Detection Toolbox[1] implemented by Clifton (2007).

The experiments are applied to two different datasets. Comparing the runtime and the accuracy of large data at the same time may yield misleading results. Due to the novelty detection accuracy, the different classifiers may detect novel classes at different points in time or not at all. Additionally, the semi-supervised learning scenario that is inherent to the novelty detection may result in different training sets for each classifier. The classification problem solved by each classifier may thus change after each bucket of samples from the data stream has been analysed. The corresponding changes in the classifier architecture have a severe influence on the runtime. Consequently, it is not feasible to analyse the runtime and the novelty detection accuracy of different classifiers in one complete experiment. Therefore, they are analysed in two distinct experiments. The former targets the runtime of the classifier while the latter targets the capabilities of novelty detection. Both experiments are applied to both data sets. The difference in the runtime of the classifiers should be clear in the experiments that use the large artificial data set (cf. Section 4.2). While the essential classification accuracy evaluation of different classifiers is based on the gesture data set (Section 4.1). We conduct both of experiments for training set sizes of 5% and 20% of the total data, respectively. In these experiments, additional labelled data (Validation data) are used to set a class-wise novelty threshold for the original PKDE.

Additionally, a third experiment is implemented to show the advantage of using the EVT in the novelty detection over the conventional methods. In this experiment, two versions of the proposed algorithm are compared, one uses the conventional novelty detection, and the other uses the EVT for the novelty detection. Only the gesture data set is discussed in this experiment in order to shorten the discussion of the evaluation. The training set size in this experiment is set to 10%.

The total dataset is randomly divided into four disjoint data sets: a training set, a learning set, a test set and a validation set except for the experiments that target the runtime. In which the validation set is not required since the novelty detection is not applied. All classifiers are adapted to the training set, and the accuracy is evaluated based on the test set. The learning set emulates the data stream and is presented to the classifier in buckets, i.e. subsequent chunks of data. The test set is comprised of 20% of the total data in all cases. The validation set is also comprised of 20% of the total data in the experiments that need validation process, and the learning set contains the remaining samples. Table 5.1 summarises the different scenarios.

[1]The NDtoolbox is available on: http://www.robots.ox.ac.uk/$\tilde{}$davidc/publications_NDtool.php

Classifier	Initial Training Set	Gesture Dataset		Gaussian Dataset	
PKDE	5%	Time	Novelty Detection	Time	Novelty Detection
SVDD					
IncPKDE					
PKDE	20%	Time	Novelty Detection	Time	Novelty Detection
SVDD					
IncPKDE					
IncPKDE-Conv	10%	EVT Benefits		–	
IncPKDE-EVT					

Table 5.1: Details of Experiments Set Up.

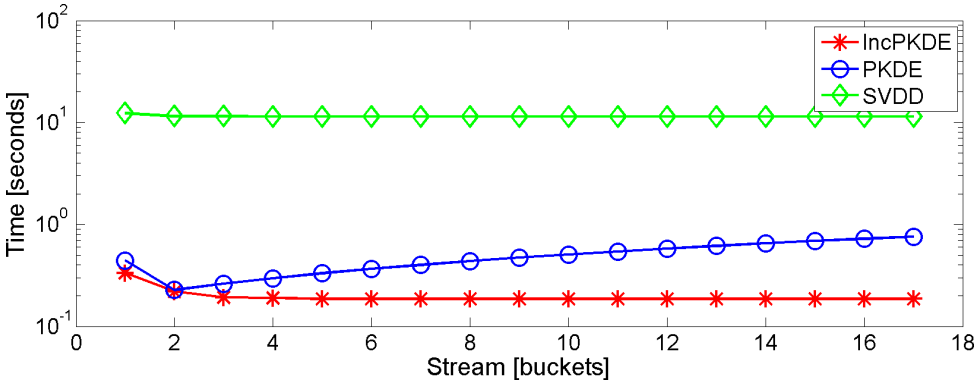
Hence, we will have nine experiments, four of them to compare the time-consuming in each dataset and each initial training set. The other four experiments are to evaluate the ability of the novelty detection, and classification accuracy. The last experiment is to evaluate the benefits of using EVT for the novelty detection. As mentioned in Section 5.2, we repeat each experiment 100 times for different random permutations of the samples while enforcing identical random permutations for the three classifiers, respectively, during each of the 100 runs.

Comparing the runtime

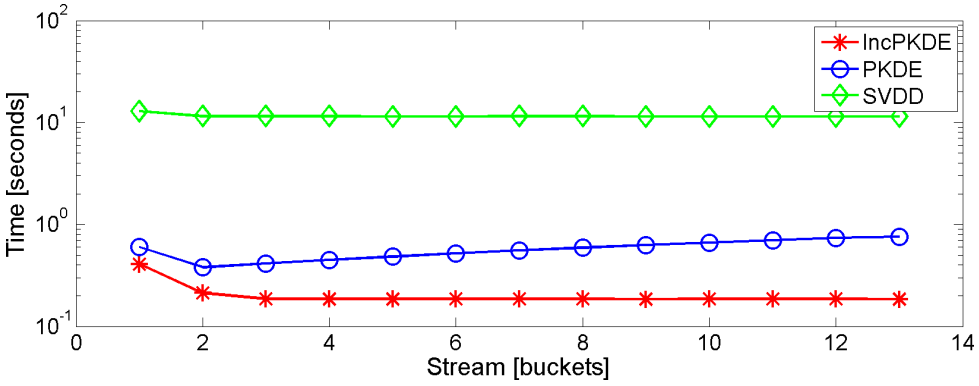
These experiments examine the time required by the algorithm for updating the classifier and classifying all data. Commonly, all classifiers are trained on the initial training set. Afterwards, the classifiers are evaluated on the test set and the current bucket of the learning set. The accuracy of the predictions is tracked and the bucket is added to the training set.

Since this experiment is performed to examine the time, we need to avoid the divergence of the classifier training sets when using the deduced labels. Hence, we enforce all algorithms to increment themselves with the correct labels for the samples in the buckets of the learning, i.e. we present the learning set to the classifiers and manually correct false classifications by enforcing correct labels. The training set of all classifiers will thus be identical at all instants of the algorithm.

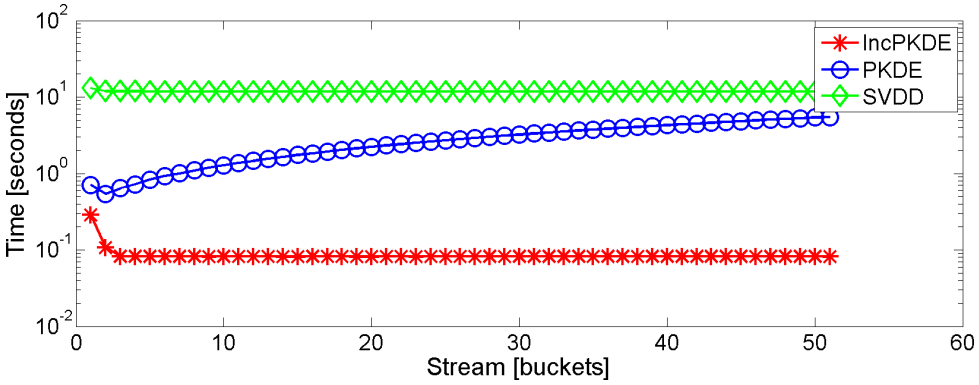
Based on the initial classifiers and the modified training set, we update all classifiers and repeat the evaluation with the next bucket of samples. Since the PKDE algorithm is not incremental, we retrain the algorithm on the modified training set while we use the incremental capabilities of IncPKDE and SVDD to update the classifiers. In addition to the accuracy, we track the time required for the update. The procedure is repeated until all buckets have been presented to the classifiers.



(a) Time consumed by the classifiers (gesture dataset, 5% training data).

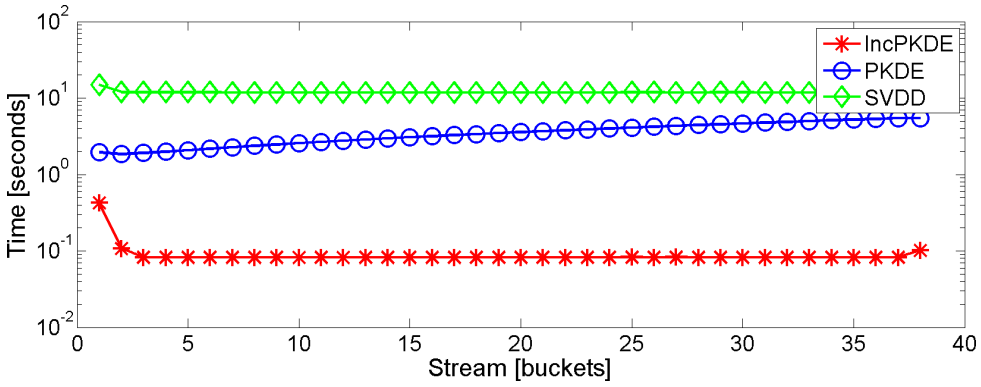


(b) Time consumed by the classifiers (gesture dataset, 20% training data).



(c) Time consumed by the classifiers (artificial dataset, 5% training data).

Figure 5.1: The runtime of the classifiers in each of the four-runtime experiment defined in Section 5.3.3. (a)–(b) Time consumed by the classifiers on the gesture dataset using 5% and 20% of the total data for the initial training, respectively. (c)–(d) Time consumed by the classifiers on the artificial dataset using 5% and 20% of the total data for the initial training, respectively.



(d) Time consumed by the classifiers (artificial dataset, 20% training data).

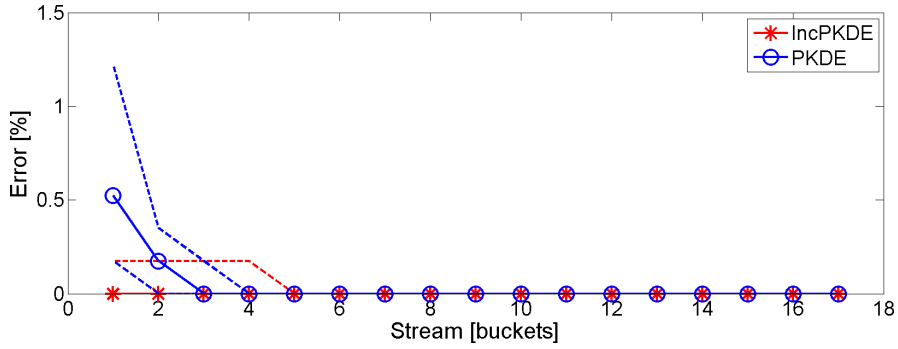
Figure 5.1: (Continued)

The gesture (9 classes, 2878 samples) and the artificial (3 classes, 9000 samples) data sets are divided into three sets. The training set contains all classes and the division is performed class-wise, i.e. the training set contains 5% or 20% of the samples of all classes, respectively. The test set is comprised of 20% of the total data in all cases, and the learning set is comprised of the remaining samples. The learning set is split into buckets of 100 samples.

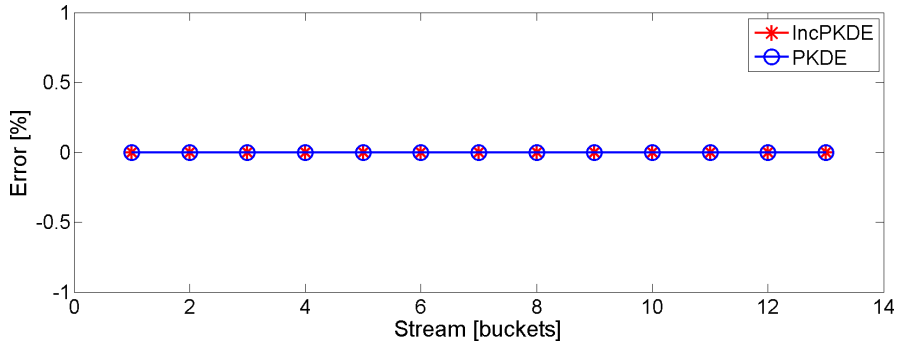
The results of the runtime experiments are summarized in Fig. 5.1–Fig. 5.3 and Table 5.2. Fig. 5.1 shows that, initially, the difference of the processing time between the original PKDE and IncPKDE is small and increases as the size of the training set increases. This result is expected because IncPKDE updates itself retaining a constant amount of Parzen windows while the original PKDE depends on all data. Let N_b and N be the number of samples in the bucket of the semi-supervised learning and the total number of samples in the dataset, respectively. The ratio of the times consumed by IncPKDE to original PKDE at the final step is then approximately equal to N_b/N. The SVDD

	Inc-Gaus-0.2		Inc-Gaus-0.05		Inc-Gtr-0.2		Inc-Gtr-0.05	
Classifiers	Error	Time	Error	Time	Error	Time	Error	Time
IncPKDE	2.39	0.10	2.33	0.08	0.02	0.19	0.02	0.19
KDPE	2.35	5.44	2.25	5.46	0.01	0.76	0.01	0.76
SVDD	2.61	11.71	2.60	11.77	0.35	11.43	0.57	11.44

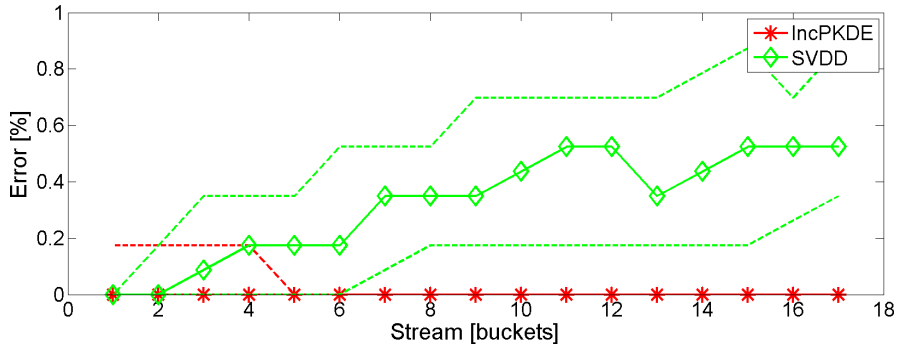
Table 5.2: Summary of the results after the last update, i.e. all learning data are added to the training set. The error rate thus equals the error rate obtained by a fully supervised learning and may be interpreted as the reference value for all iterations shown in Fig. 5.1, Fig. 5.2 and Fig. 5.3.



(a) Error rate of PKDE and IncPKDE (5% training of gesture data).

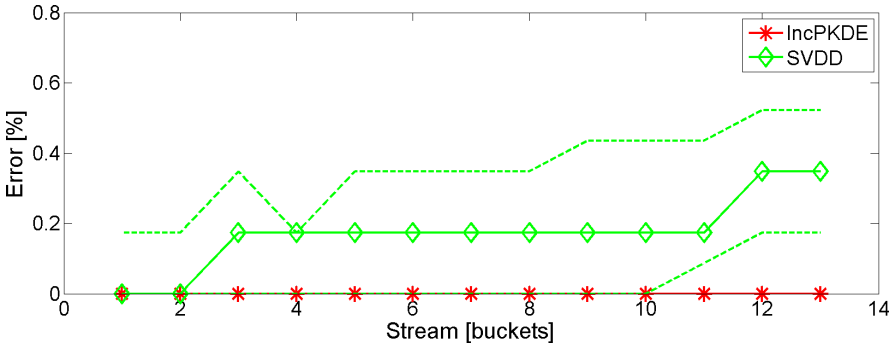


(b) Error rate of PKDE and IncPKDE (20% training of gesture data).



(c) Error rate of SVDD and IncPKDE (5% training of gesture data).

Figure 5.2: Error rates of the reference methods PKDE and SVDD in comparison to IncPKDE on the gesture dataset in the runtime experiment defined in Section 5.3.3. The solid line represents the median over one hundred runs while the dashed lines represent the 25% and the 75% quantile, respectively. (a)–(b) Error rates of the PKDE in comparison to IncPKDE using 5% and 20% of the total data for the initial training, respectively. (c)–(d) Error rates of the SVDD in comparison to IncPKDE using 5% and 20% of the total data for the initial training, respectively.

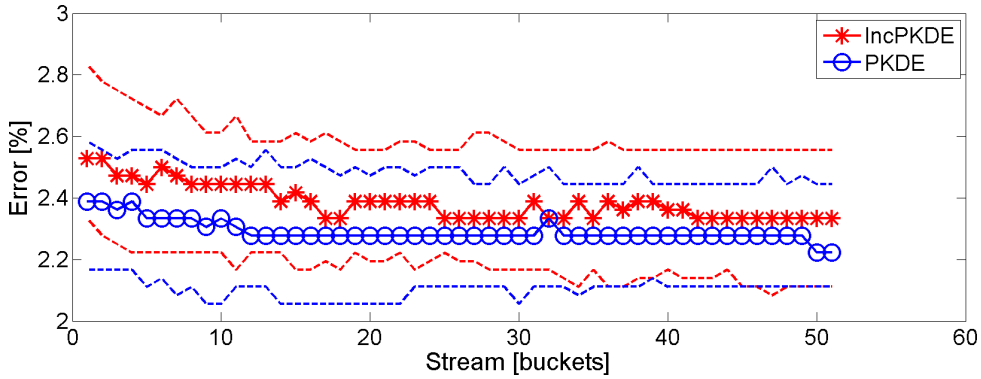


(d) Error rate of SVDD and IncPKDE (20% training of gesture data).

Figure 5.2: (Continued)

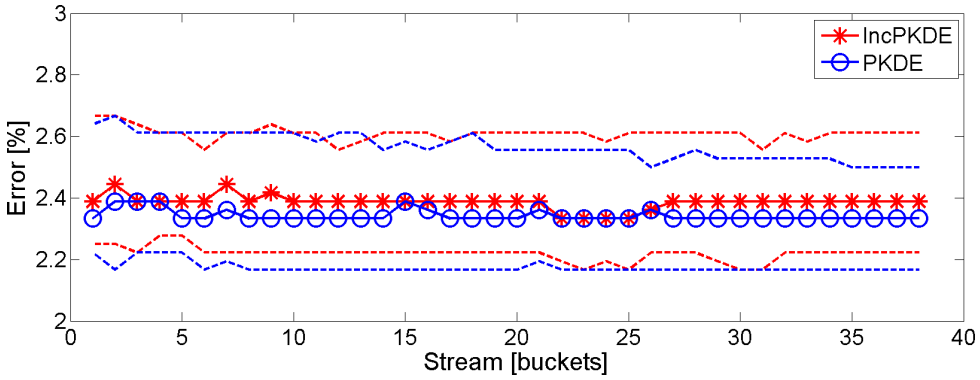
is an incremental classifier. However, it still consumes much more time than IncPKDE and original PKDE. This behaviour might be due to the non-linear optimisation problem that is solved during each update. Since the total number of the gesture samples in the dataset is small, the difference in time among the classifiers is small but still significant as shown in Fig. 5.1(a) and Fig. 5.1(b). The behaviour, however, is clearer in the case of a large artificial dataset (see Fig. 5.1(d)).

The error rates of IncPKDE and original PKDE, respectively, are very close to each other as shown in Fig. 5.2(a)–Fig. 5.2(b) and Fig. 5.3(a)–Fig. 5.3(b). The proposed

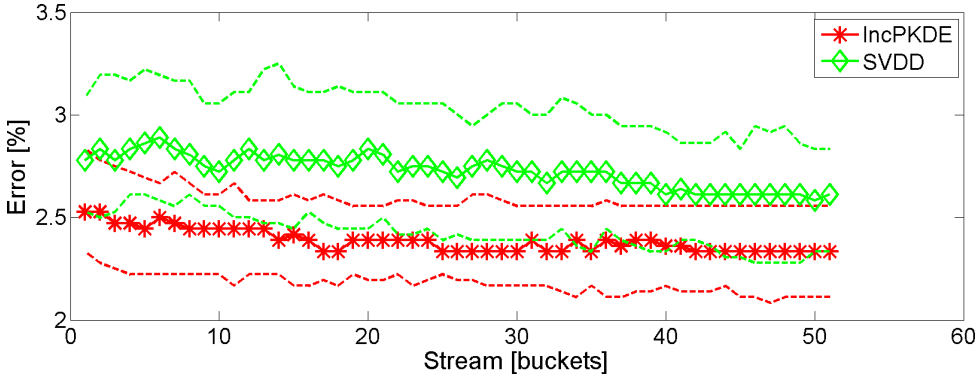


(a) Error rate of PKDE and IncPKDE (5% training of artificial data).

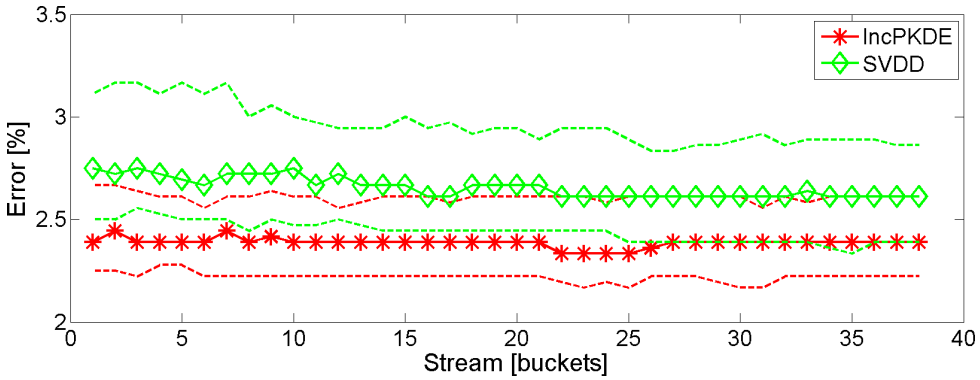
Figure 5.3: Error rates of the reference methods PKDE and SVDD in comparison to IncPKDE on the artificial dataset in the runtime experiment defined in Section 5.3.3. (a)–(b) Error rates of the PKDE in comparison to IncPKDE using 5% and 20% of the total data for the initial training, respectively. (c)–(d) Error rates of the SVDD in comparison to IncPKDE using 5% and 20% of the total data for the initial training, respectively.



(b) Error rate of PKDE and IncPKDE (20% training of artificial data).



(c) Error rate of SVDD and IncPKDE (5% training of artificial data).



(d) Error rate of SVDD and IncPKDE (20% training of artificial data).

Figure 5.3: (Continued)

IncPKDE thus successfully reduces the update time of the classifier while maintaining a similar accuracy. Both the original PKDE and IncPKDE perform much better than SVDD (Fig. 5.2 and Fig. 5.3).

Evaluate the novelty detection

To evaluate the ability of detecting the unknown samples, we define a different set of experiments. In contrast to the runtime experiments, we apply a fully semi-supervised learning approach.

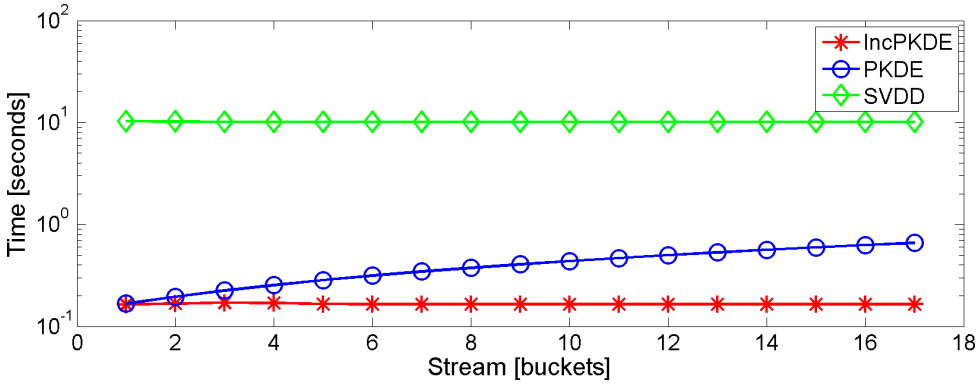
The original PKDE algorithm requires additional data to determine the thresholds of the novelty detection. Therefore, we introduce a fourth dataset: the validation set. The novelty detection of the IncPKDE in this experiment is based on the conventional method i.e. uses the validation set. Unknown samples are introduced by excluding one class from the initial training set and the validation set, respectively. All datasets are divided randomly, and the learning data are presented to the classifiers in buckets following the same strategy as in the previous experiment. The labels of the added data, however, are the outputs of the classifier.

In addition to the class label, the classifiers may set the believability flag or the novelty flag. The novelty flag may set if the classifier supposes that the sample belongs to novel class or outlier. In contrast, the believability flag is set if the classifier trusts the assigned label. New samples are added to the training set if and only if their believability flag is confirmed. The PKDE classifier set this flag if the posterior of the class exceeds the corresponding threshold by a factor of two. While the IncPKDE is set the flag if P_{evt} is less than P_{th} and the difference between the smallest and next smallest outputs of the EVT scores is larger than a specified threshold. This threshold is estimated using the known classes. In the case of gesture data, it mostly equals to 0.05. Since the SVDD is a distance based classifier, a sample is added to the SVDD training set if the classifier output below half of the threshold value. A sample is flagged as novel if no class passes the threshold check. It is thus possible that the training sets of the different classifiers may diverge and contain different samples and possibly false labels.

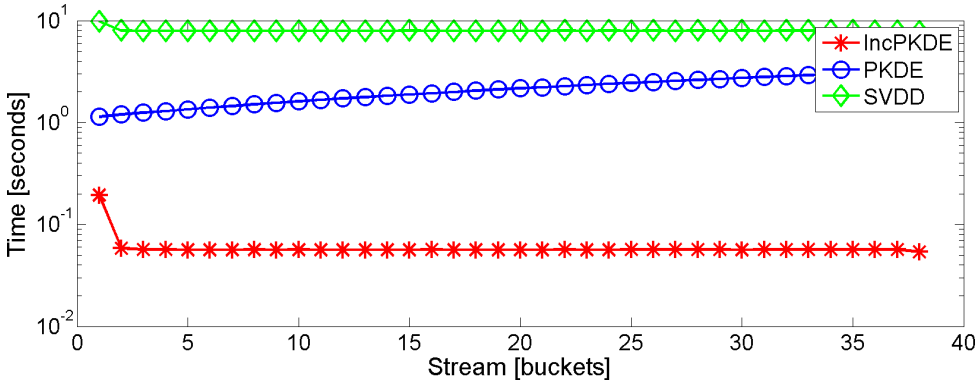
The whole procedure is repeated for 100 random subdivisions of the data per class, i.e. each class is omitted from the training set once. Again, we ensure the same random subdivision for all classifiers during one run.

The novelty detection experiments, again, are performed for two initial training set sizes, which are 5% and 20% of the total number of samples in the dataset. The fractions of the test set and the validation set are both 20%. The learning set is comprised of the remaining samples. The division is executed class-wise. One class, however, is omitted from the training set and the validation set, respectively.

Notably, the figures show only the results of the experiments those use 5% of gesture dataset and 20% of artificial dataset due to the large number of figures. Fig. 5.4 shows the values T_{update} of the three algorithms for the novelty detection experiment. In general,



(a) Time consumed by the classifiers (gesture dataset, 5% training data).

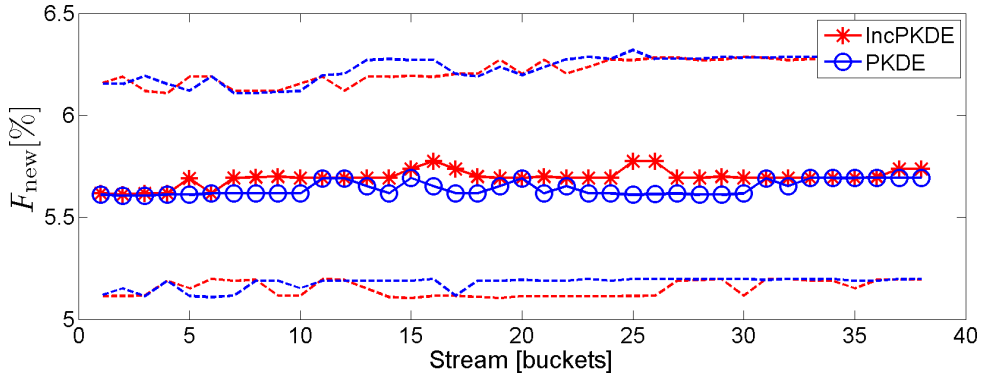


(b) Time consumed by the classifiers (artificial dataset, 20% training data).

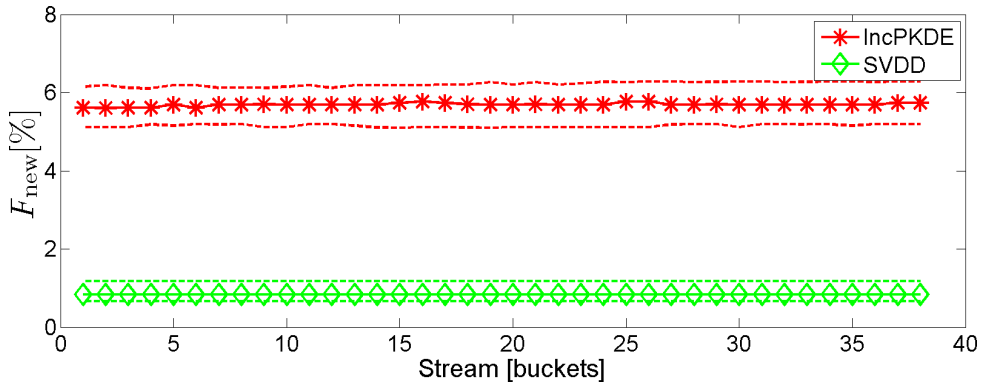
Figure 5.4: The time consumed by the classifiers (T_{update}) in the novelty detection experiment defined in Section 5.3.3, where the initial training is: (a) 5% of the gesture dataset. (b) 20% of the artificial dataset.

the results are similar to the runtime based experiment. The training set and thus the estimated update time, however, is now influenced by the novel classes. Consequently, the training set may differ for each classifier after another bucket has been presented and a direct comparison of the graphs is not reasonable. The results, however, show a general trend of IncPKDE resulting in lower update times than PKDE and SVDD, respectively, as shown in Fig. 5.4(a) and Fig. 5.4(b).

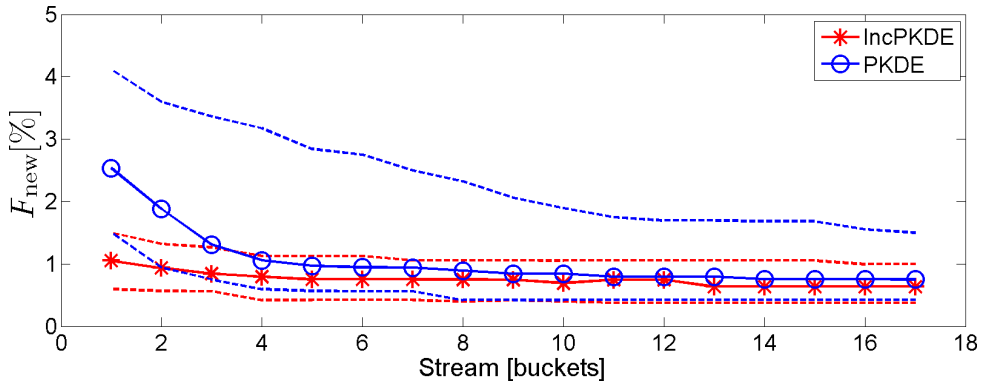
The F_{new} performance metric is shown in Fig. 5.5. In case of the artificial dataset, the IncPKDE performs similar to the original PKDE (see Fig. 5.5(a)) but worse than the SVDD (see Fig. 5.5(b)). In general, the ratio of M_{new} to F_{new} may be controlled by adjusting the threshold. However, if F_{new} is decreased the M_{new} might be increased due to the nature of the performance metrics. Here, we use the original threshold computation of Tax and Duin (1999) for the SVDD. In case of the gesture dataset, the IncPKDE



(a) PKDE and IncPKDE (artificial dataset, 20% training data).

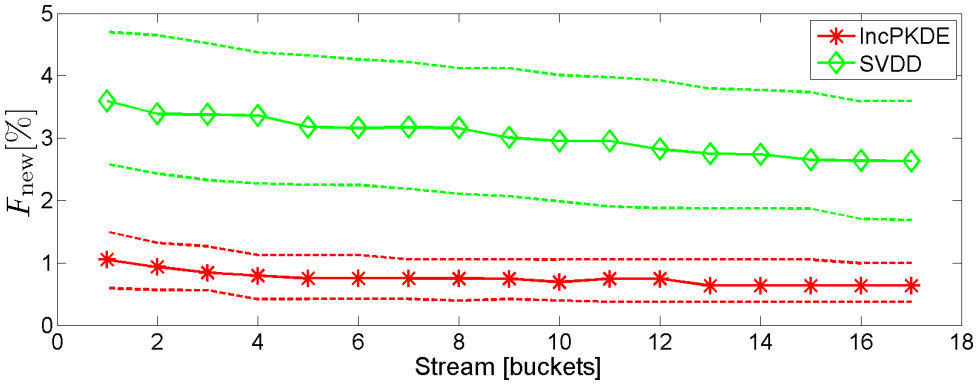


(b) SVDD and IncPKDE (artificial dataset, 20% training data).



(c) PKDE and IncPKDE (gesture dataset, 5% training data).

Figure 5.5: The F_{new} error by each classifier. (a)–(b) All classifiers were initially trained on 20% of the artificial dataset. (c)–(d) All classifiers were initially trained on 5% of the gesture dataset.



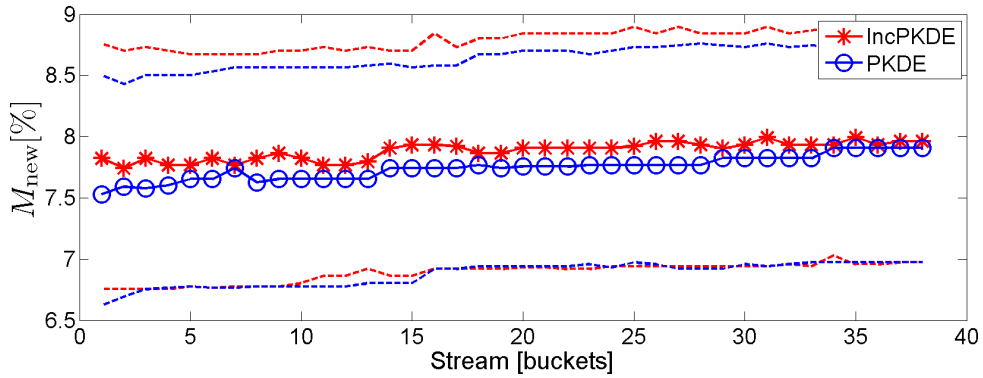
(d) SVDD and IncPKDE (gesture dataset, 5% training data).

Figure 5.5: (Continued)

outperforms the reference methods (Fig. 5.5(c) and Fig. 5.5(d)).

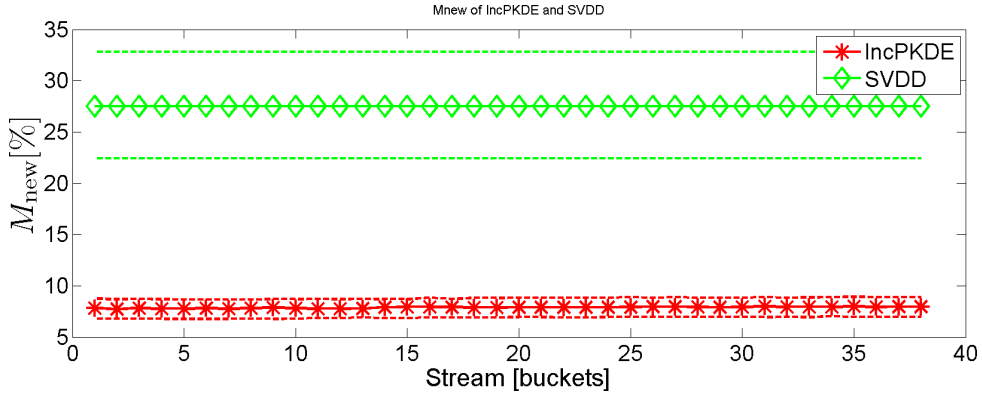
We note that the slope of the IncPKDE's errors is almost always negative and if it was positive it decreased with time more than original PKDE. This performance is due to the reduction of the effect of the outliers or falsely classified samples used to update the classifiers in the semi-supervised learning in each subsequent process by the factor $1/N_{\text{c}}$, where N_{c} is the number of samples in a particular class.

Fig. 5.6 shows the evolution of the M_{new} performance metric during the learning phase of the algorithm. In almost all cases, the IncPKDE shows lower median and quantile values of missed novel samples than PKDE and SVDD. The performance of the SVDD

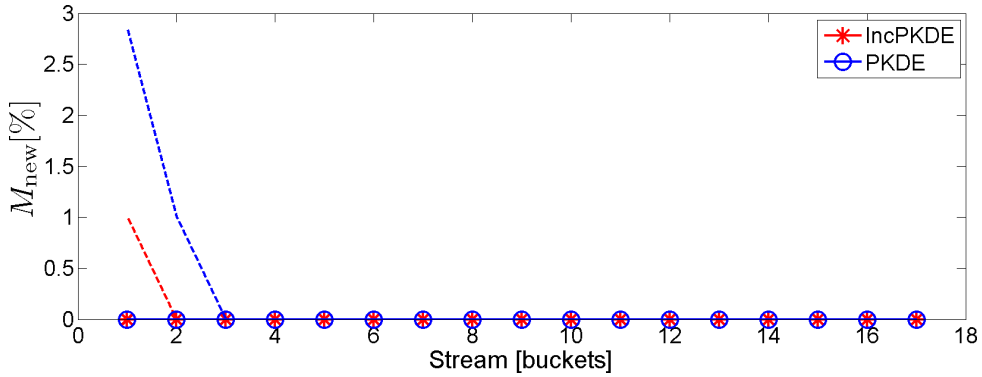


(a) PKDE and IncPKDE (artificial dataset, 20% training data).

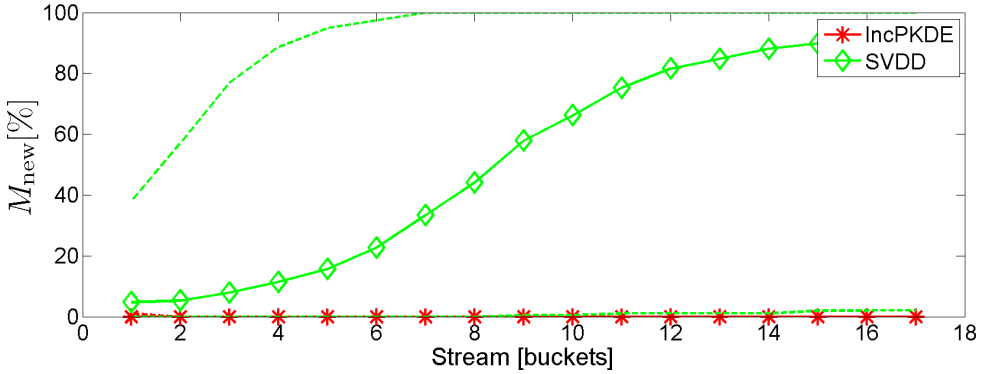
Figure 5.6: Novel class samples M_{new} missed by each algorithm. (a)–(b) All classifiers were initially trained on 20% of the artificial dataset. (c)–(d) All classifiers were initially trained on 5% of the gesture dataset.



(b) SVDD and IncPKDE (artificial dataset, 20% training data).

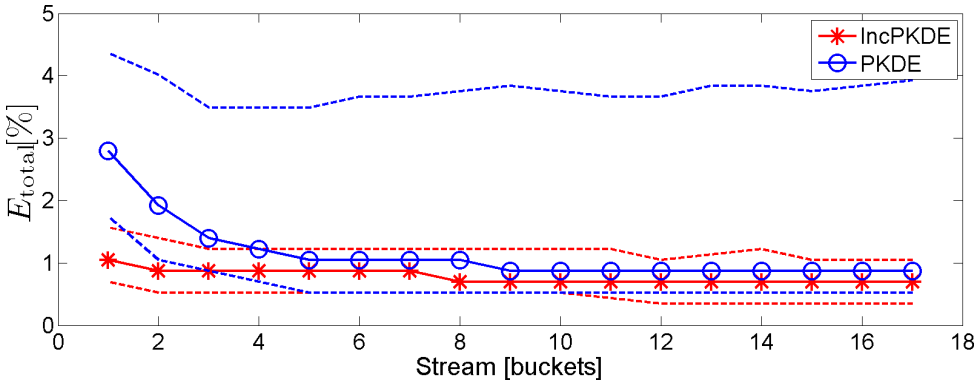


(c) PKDE and IncPKDE (gesture dataset, 5% training data).

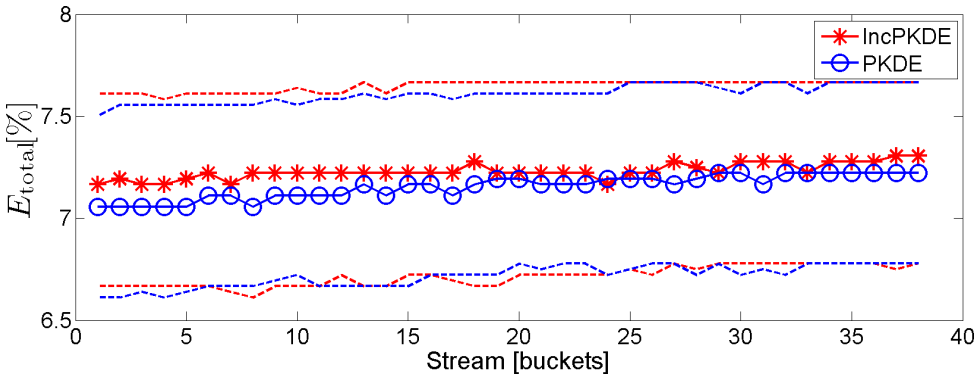


(d) SVDD and IncPKDE (gesture dataset, 5% training data).

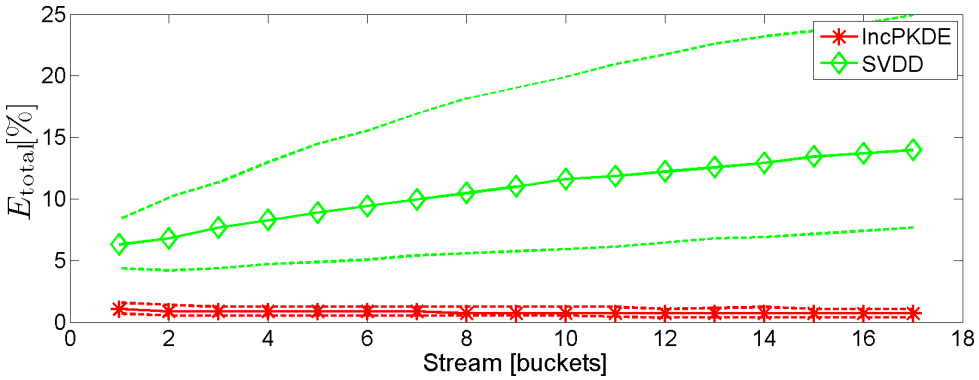
Figure 5.6: (Continued)



(a) Total error rate of PKDE and IncPKDE (gesture dataset, 5% training data).

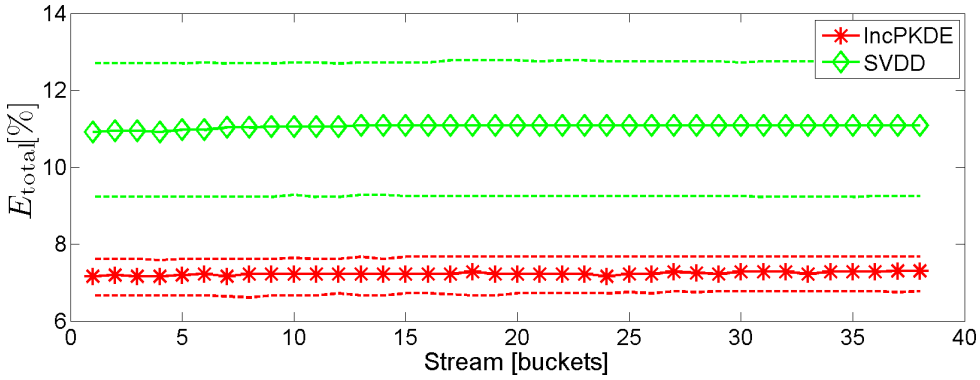


(b) Total error rate of PKDE and IncPKDE (artificial dataset, 20% training data).



(c) Total error rate of SVDD and IncPKDE (gesture dataset, 5% training data).

Figure 5.7: Total error rates E_{total} in the novelty detection experiments defined in Section 5.3.3. (a)–(b) E_{total} of the PKDE in comparison to IncPKDE on the gesture dataset using 5% and artificial dataset using 20% of the total data for the initial training, respectively. (c)–(d) E_{total} of the SVDD in comparison to IncPKDE on the gesture dataset using 5% and artificial dataset using 20% of the total data for the initial training, respectively.



(d) Total error rate of SVDD and IncPKDE (artificial dataset, 20% training data).

Figure 5.7: (Continued)

is exceptionally poor in the case of a small initial training fraction of the gesture dataset (see Fig. 5.6(d)).

Fig. 5.7 shows the total error E_{total} of the classifiers for the novelty detection experiment. Due to the divergence in the training data at the subsequent updating, the accuracy of the classifiers is expected to diverge also. However, the values of the total

Data Base	Error	IncPKDE	PKDE	SVDD
Inc-Gaus-0.2	M_{new} [%]	8.00	7.91	27.90
	F_{new} [%]	5.74	5.77	0.92
	E_{total} [%]	7.25	7.23	11.10
	T_{update} [s]	0.06	3.16	7.93
Inc-Gaus-0.05	M_{new} [%]	8.34	8.10	16.80
	F_{new} [%]	5.95	5.92	3.35
	E_{total} [%]	7.51	7.41	9.06
	T_{update} [s]	0.06	3.04	8.15
Inc-Gtr-0.2	M_{new} [%]	1.78	2.36	45.15
	F_{new} [%]	0.60	0.56	1.61
	E_{total} [%]	0.68	0.69	7.10
	T_{update} [s]	0.16	0.67	10.16
Inc-Gtr-0.05	M_{new} [%]	5.87	8.51	60.92
	F_{new} [%]	0.75	1.67	2.78
	E_{total} [%]	1.24	2.56	17.63
	T_{update} [s]	0.17	0.65	10.16

Table 5.3: Summary of the novelty detection performance metrics after the last update.

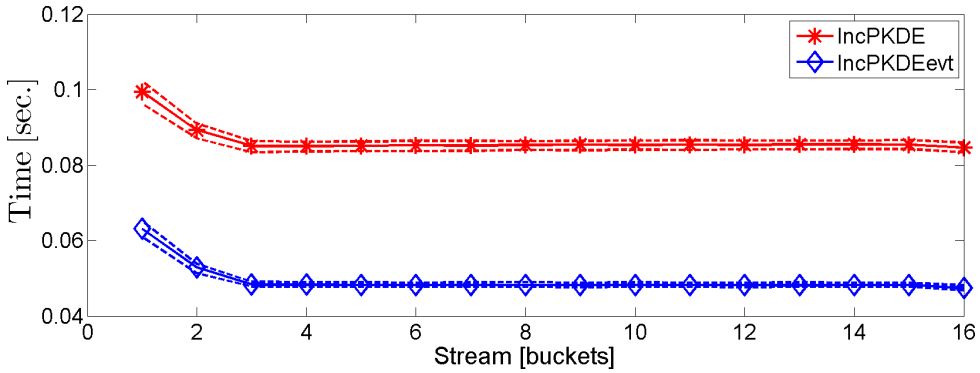
error of the IncPKDE classifier are similar to the error of the PKDE classifier or even better (Fig. 5.7(a) and Fig. 5.7(b)). Furthermore, the E_{total} metric of the IncPKDE is better than the SVDD classifiers in all cases as shown in Fig. 5.7(c) and Fig. 5.7(d).

Table 5.3 summarizes the results of the error metrics after the last update, i.e. all data have been presented to the classifier.

Comparing the conventional novelty detection and EVT

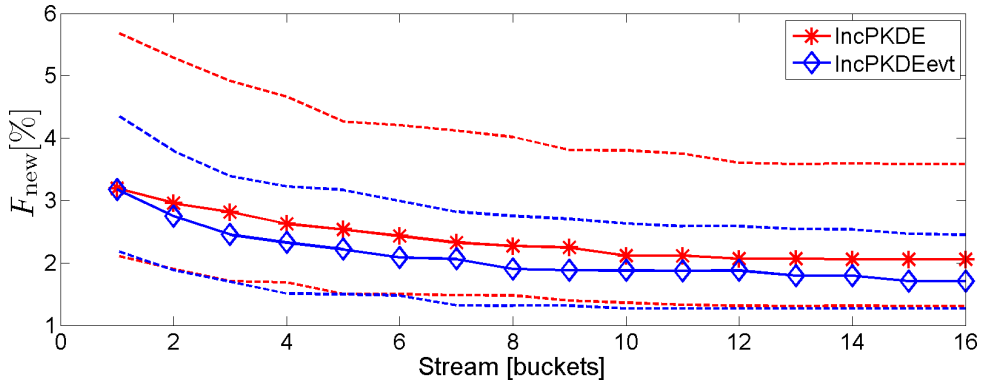
This experiment is to show the advantages of using the EVT for the novelty detection over the conventional methods. Two IncPKDE classifiers are trained on the same initial training set. In this experiment, the initial training set size is set to 10% of the total data and they are randomly selected. The first classifier uses the EVT technique to detect the novel samples while the second classifier uses additional 20% from the total labelled data to set the novelty threshold by using the conventional method.

The test set and the learning set sizes are 20% and 50% of the total data, respectively. As explained in Section 5.2, the novel class is emulated by excluding one class from the training and validation sets, and the learning set is submitted to the classifiers as buckets each contains 100 samples to simulate the data streams. Each classifier updates itself to the data which it trusts. In this experiment, the EVT threshold P_{th} is set to 0.95, while the other classifier uses the validation data set to compute the threshold for each classifier updating. The assignment of a sample is considered trusty in the EVT classifier if the

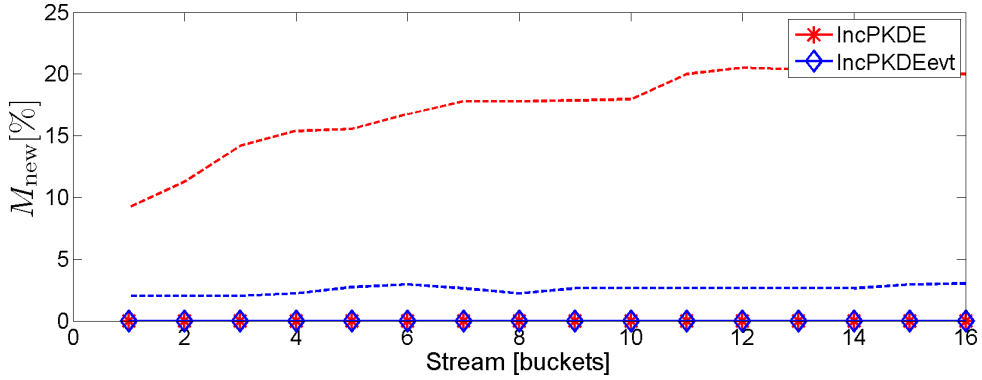


(a) T_{update}

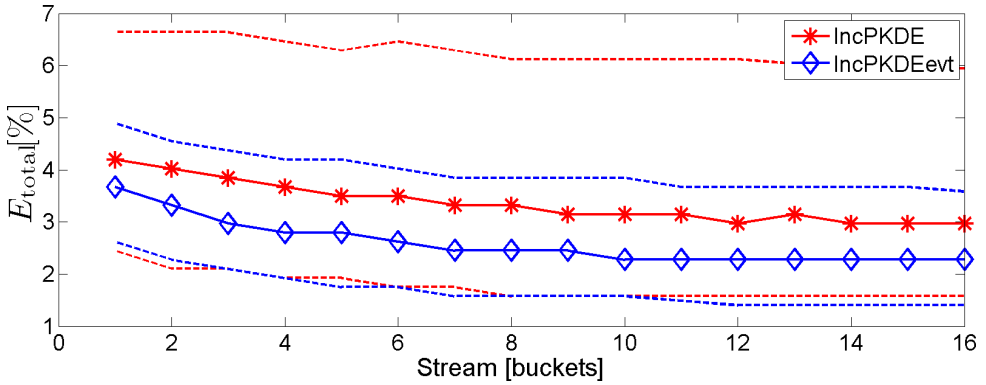
Figure 5.8: IncPKDEevt indicate the classifier that uses the EVT for the novelty detection and IncPKDE represent the classifier that uses the validation data in conventional method of novelty detection. Both classifiers were initially trained on 10% of the gesture dataset. (a) Time consumed by IncPKDEevt and IncPKDE classifiers (T_{update}). (b) Samples falsely identified as novel by each classifier (F_{new}). (c) Novel class samples missed by each classifier (M_{new}). (d) Total Error rate of IncPKDEevt and IncPKDE classifiers (E_{total}).



(b) F_{new}



(c) M_{new}



(d) E_{total}

Figure 5.8: (Continued)

sample is not indicated as novel and the difference between the smallest novelty score and the second smallest is more than 0.05.

The processing time of the classifier that uses the EVT for the novelty detection is less than half of the processing time of the compared classifier as shown in Fig. 5.8(a). The number of the samples F_{new} of the classifier uses the EVT is less than the corresponding values of the classifier that uses the conventional novelty detection (Fig. 5.8(b)). This mean using the EVT enables the classifiers to detect the samples, which belong to known classes accurately. Thus, there are less false alarms. Although the median of M_{new} in both classifiers are zero, the 75% quantile of the EVT classifier is less than the conventional classifier (Fig. 5.8(c)). Fig. 5.8(d) shows the total error is reduced by using the EVT. Thus, in addition to overcoming the needs of validation data, the performance of the classifier is improved in several aspects including less processing time, less false alarm, better novelty detection and hence better accuracy.

5.4 Semi-Supervised Support Vector Machine

After the introduction of the support vector machine (SVM), the first implementation of an incremental SVM algorithm was achieved by Syed et al. (1999). Remaining difficulties of the incremental SVM were resolved by Cauwenberghs and Poggio (2000). The latter method was extended by some researchers, e.g. Tax and Laskov (2003), but no successful practical application of any of these methods has been reported (Laskov et al., 2006). In the work of Laskov et al. (2006) the difficulties and the problems faced during the implementation of the incremental SVM were analysed in detail, and an efficient method of incremental SVM by addressing the bottlenecks of the SVM problem is proposed. It offers the possibility to add and remove single samples or a batch of samples in the incremental phase.

As the SVM classifier known to be accurate for both linearly and non-linearly separable data, we extend it here to work within a semi-supervised scenario. We present an SVM classifier that has efficient ability to detect outliers in a multi-class system using the extreme value theory.

The proposed method is based on the incremental SVM presented by Laskov et al. (2006), who proposed and implemented an efficient incremental SVM algorithm by combining a new storage technique with an intelligent organisation of the minor iteration computations. They minimised the selection operation by using column-wise and row-wise matrix storage together. Additionally, they used gaxpy-type matrix-vector multiplication for further minimization. They improve the computational efficiency of the SVM by a factor of 5–20. The SVM is used in a one-vs-one or one-vs-all manner, which means to determine a classifier between each class pair or between one class against all other classes, respectively. Since the training data is stream continuously, the emergence of new classes

is expected and using one-vs-all is required to simplify the extension of the classifier to new classes.

Unfortunately, the SVM output is not calibrated to represent a probability value (Platt et al., 1999), hence it is difficult to use it for novelty detection or obtain accurate results in a multi-class system. Several works attempt to fit the SVM output to a posterior probability, e.g. (Vapnik, 1998; Wahba et al., 1999; Hastie et al., 1998). Here we adopt the method of Platt et al. (1999), who estimate the posterior probability $p(y|f_{svm})$ based on a parametric model, where y denotes the class and f_{svm} the binary SVM classifier output. Empirically, they show that the class-conditional densities are far away from Gaussian and have a discontinuity at the positive and negative margins $f_{svm} = +1, -1$, respectively. Additionally, they propose a sigmoid approach according to

$$p(y = 1|f_{svm}) \approx \frac{1}{1 + \exp(Af_{svm} + B)} \tag{5.9}$$

with A and B as the sigmoid parameters that need to be optimised. Platt et al. (1999) use additional labelled data or the training data themselves to estimate the parameters A and B using maximum likelihood. With (f_{svm_i}, t_i) as the training data, where f_{svm_i} is the SVM output and t_i the target probability of the i_{th} training sample, the target probability computed by using the target class y_i as

$$t_i = \begin{cases} \frac{N^+ + 1}{N^+ + 2}, & \text{if } f_{svm_i} \geq 0 \\ \frac{1}{N^- + 2}, & \text{if } f_{svm_i} < 0 \end{cases} \tag{5.10}$$

with N^+ and N^- as the number of samples of the two classes, respectively. For simplicity, let we denote $p(y = 1|f_{svm})$ for the sample i by p_i, then the cross-entropy error function of this model is

$$-\Sigma_i t_i \log(p_i) + (1 - t_i) \log(1 - p_i), \tag{5.11}$$

and

$$p_i = \frac{1}{1 + \exp(A \cdot f_{svm_i} + B)}. \tag{5.12}$$

Unfortunately, the optimisation used by Platt et al. (1999) needs all the data at the beginning of the estimation such that no incremental update is possible. Instead, we propose an incremental method of estimating the parameters by using stochastic gradient descent. Hence, the parameters are updated as

$$A = A - \gamma \cdot \frac{\partial f_{svm}}{\partial A}, \quad and \quad B = B - \gamma \cdot \frac{\partial f_{svm}}{\partial B}, \tag{5.13}$$

where γ is the learning step (here: $\gamma = 0.01$). Now the estimation process of the parameters A and B is incremental. In order to make the output of the proposed SVM fit

the Gumbel distribution EVT, the input distribution to the EVT should be a one-sided normal distribution $|\mathcal{N}(0,1)|$ with the most believable sample lying close to the mean (zero), becoming less believable by travelling far from the mean. By subtracting one from the reciprocal of the probability, we obtain a similar distribution, which is equivalent to $\exp(A \cdot f_{svm} + B)$. To estimate the distribution of this function, we calculated the histogram of $(1/p - 1)$ of a randomly selected learning set and found a shape similar to a one-sided normal distribution (Fig. 5.9).

These values can be easily applied to the EVT to calculate the novelty of the samples. The parameters (Eq. (3.20) and Eq. (3.19)) of any of the EVT distributions depend only on the number of samples N drawn from the distribution (Clifton et al., 2008). This helps to decrease the influence of the amount of training data on the results of the EVT to obtain a constant threshold even for a changing number of samples. The distribution of the EVT outputs is shown in Fig. 5.10, showing that a threshold P_{th} can be set between the normal samples (near zero) and the abnormal samples (near one). Additionally, as apparent from (Eq. (3.14)), P_{th} does not depend on the distribution of the classes, which is in contrast to the conventional threshold methods, and it has a direct statistical interpretation (Roberts, 1999).

The proposed algorithm has three outputs for each test sample. These outputs are the most likely class label, the believability flag and the novelty flag. The believability flag is set to 1 if the label in the first output is highly believable. Contrary, the novelty flag is set to 1 if the sample is strongly believed to be novel. The second output is used in semi-supervised learning to select only the believable samples for updating the classifier. Here the second output is set to 1 if the sample is not detected as novel and its output

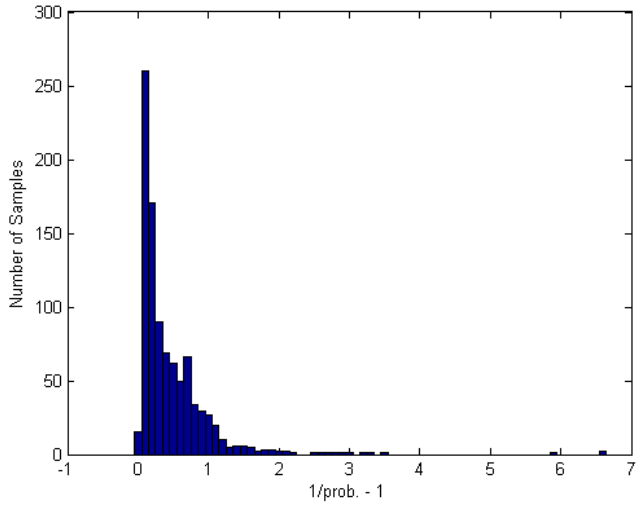


Figure 5.9: The distribution of $[\frac{1}{p} - 1]$ for 40% of the data.

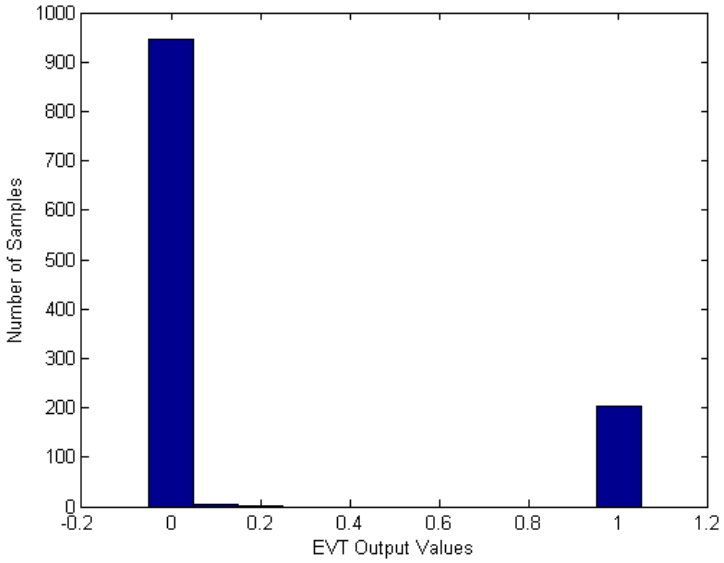


Figure 5.10: The distribution of the EVT's output of the 40% of the data.

probability is more than 0.5.

5.4.1 Experimental set-up

In this section, the proposed algorithm is evaluated on its performance. A comparison is performed with those of the SVDD (cf. Section A.1) and the original incremental SVM on the same data. The implementation of the incremental SVM according to Laskov et al. (2006) is also available online[2]. Since all of the algorithms are supposed to work as a semi-supervised classifier, the database is divided into three sets: a training set, learning set, and testing set. In the first experiment, the results of the proposed algorithm and those of the SVDD algorithm are compared. In this experiment, the data sets are divided class-wise into the fractions 40%, 40% and 20% for the training, learning and test set, respectively. In the second experiment, the results of the proposed algorithm and those of the original SVM are compared, showing the advantage of the incremental updating of the probability's parameters over the original one. In this experiment, the data sets are divided class-wise into the fractions 50%, 25% and 25% for the training, learning and test set, respectively. The learning set is used to simulate the streaming data by dividing it into buckets of 100 or 50 samples each in the first and second experiment, respectively. Initially, the training data contain 40%/50% from each class, then the novel class is simulated by excluding one class from the training set at a time. The classifier

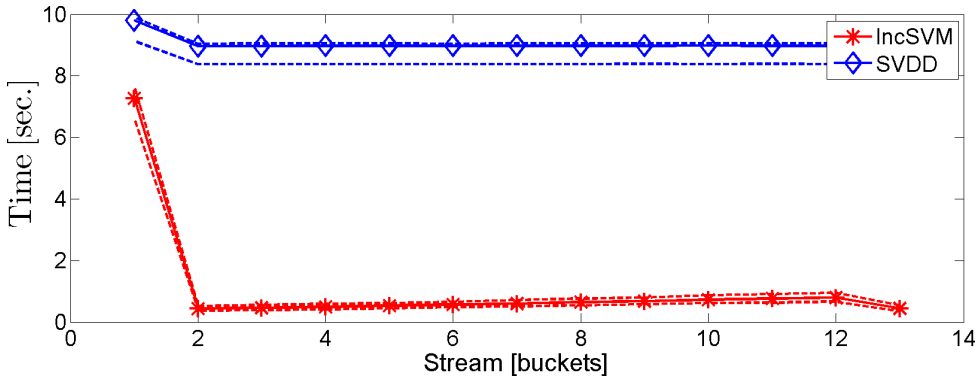
[2]The incremental SVM implementation of Laskov et al. (2006) is available at http://www-ti.informatik.uni-tuebingen.de/%7espueler/mcpIncSVM/

performance is determined based on the test set. The novelty threshold P_{th} of the EVT in the proposed algorithm set to 0.5. Thus, the sample will consider as novel if the corresponding EVT output is equal or greater than 0.5. In contrast, the believability flag will set if the sample is not detected as novel and its output probability is more than 0.5.

Since the SVDD described by Tax and Duin (2004) is a one-class classifier constructing a hypersphere enclosing the data, it considers some data which are far from the centre as outliers, and by selecting the ratio of the outliers within this class in the training phase, the SVDD code sets the threshold of the novelty. The default value of the outlier ratio to the total data of each class is 5%. We used the default value since it gives the best performance. The "multic" function in the PRToolBox (cf. Section A.1) selects the class of the maximum output and identifies the sample as an outlier if all classifiers indicate it as an outlier.

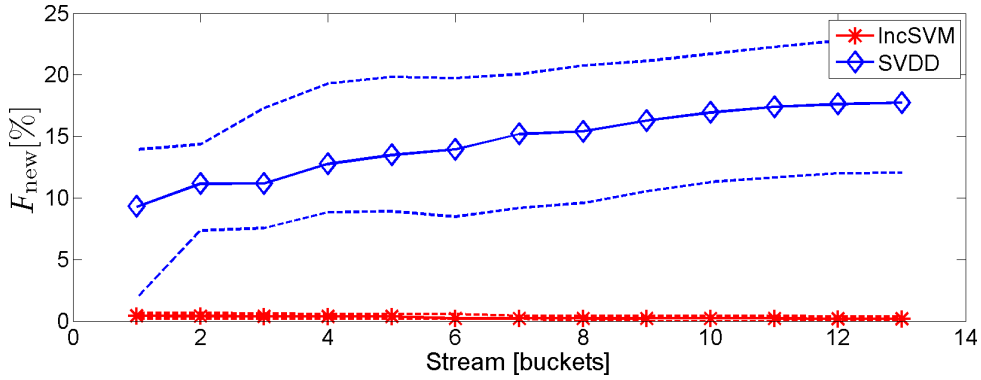
5.4.2 Results and discussion

The computation time of the proposed classifier is shorter than that of the SVDD, as shown in Fig. 5.11(a). The computation time required for updating itself with one bucket is just some milliseconds on a standard PC, which matches the incremental learning requirement. We used just the available data to estimate the probability distribution parameters in the original SVM because we suppose that not all old data can be stored, and the processing time is very close to that of the proposed algorithm (Fig. 5.12(a)). A

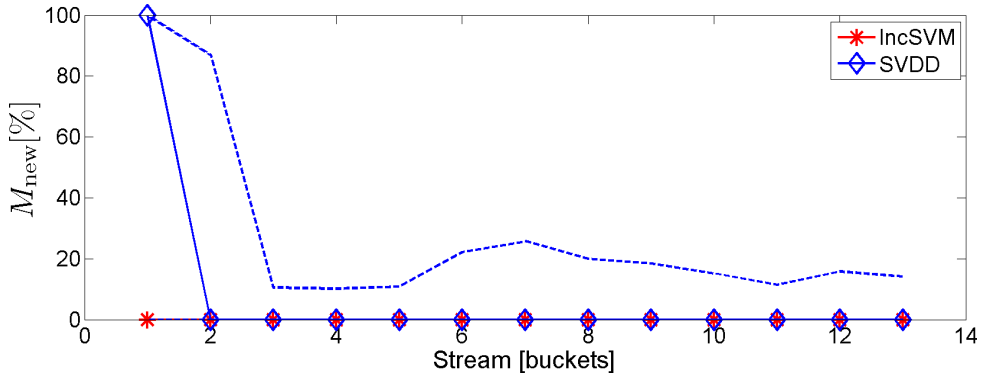


(a) T_{update}

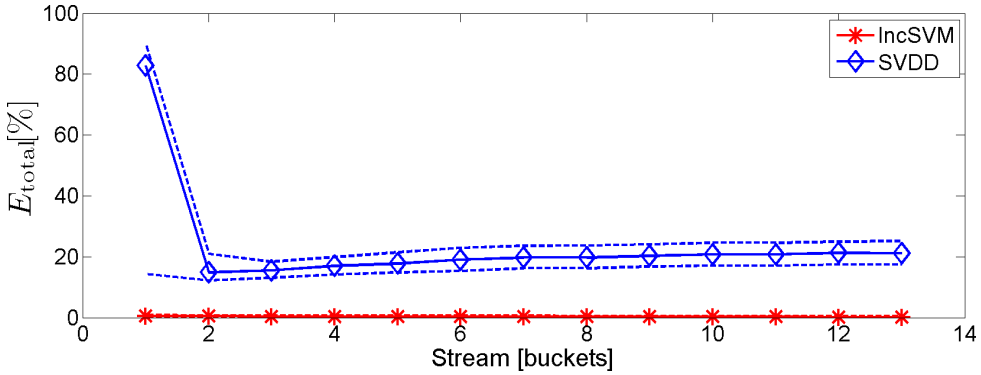
Figure 5.11: The performance metric of the proposed incremental SVM (IncSVM) and the SVDD classifiers. Both classifiers are initially trained on 40% of the gesture dataset, the learning set and the testing set sizes are 40% and 20%, respectively. (a) The computation time required for each bucket (T_{update}), i.e. classification and retraining. (b) Samples falsely identified as novel by each classifier (F_{new}). (c) The rate of missed novelties (M_{new}). (d) The total error rate (E_{total}).



(b) F_{new}



(c) M_{new}



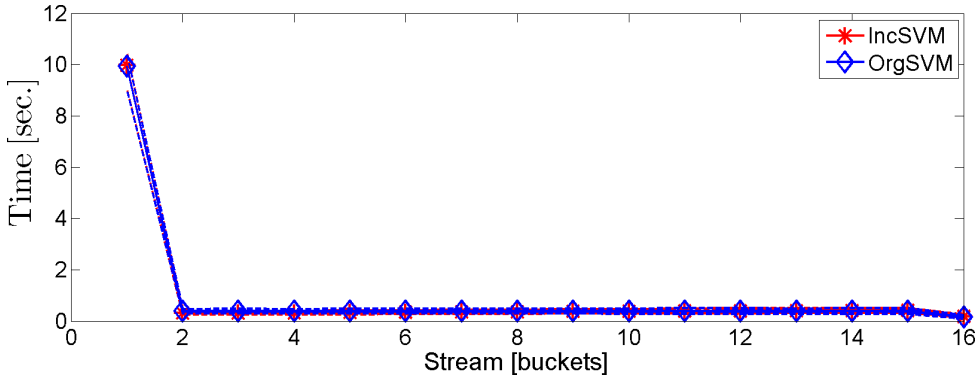
(d) E_{total}

Figure 5.11: (Continued)

sample indicated as an outlier by the classifier will be excluded from the next training phase. Hence, the value of F_{new} has no direct influence on the classifier performance except that the number of additional training sets will be smaller with increasing F_{new}. However, the value of it in the case of the proposed algorithm is constant and near zero but in the case of the original SVM and the SVDD classifier it increases with successive iterations, as depicted in Fig. 5.12(b) and Fig. 5.11(b), respectively.

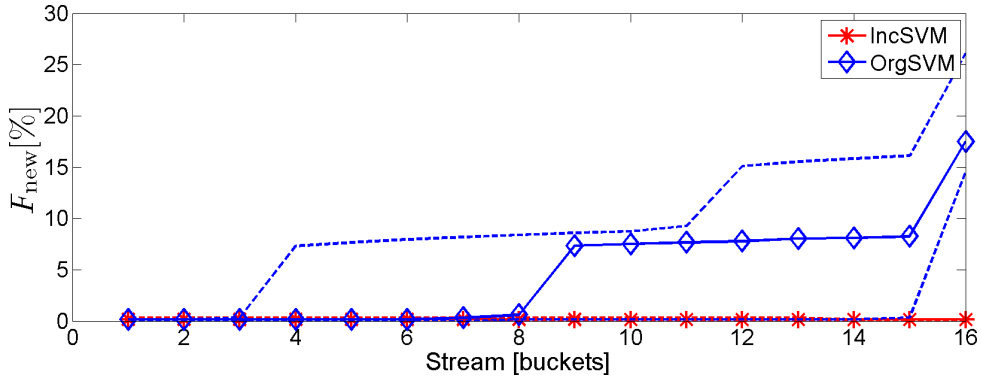
The ability of the classifiers to indicate the outliers, which have a negative effect on the accuracy of semi-supervised learning, is summarised in Fig. 5.11(c) and Fig. 5.12(c). From these figures, it becomes apparent that all outliers are indicated by the proposed algorithm in all experiments. In contrast, the 75% quantile error rate of the original SVM is about 2%.

The total error rate E_{total} (Fig. 5.12(d) and Fig. 5.11(d)) is essential for the evaluation of semi-supervised learning. Initially, it is also close to zero for the proposed algorithm then decreases with increasing the amount of the training data. This means that the new unlabelled data add new information to the classifier. In contrast, its value is larger and increases with increasing amount of training data in the case of other classifiers.

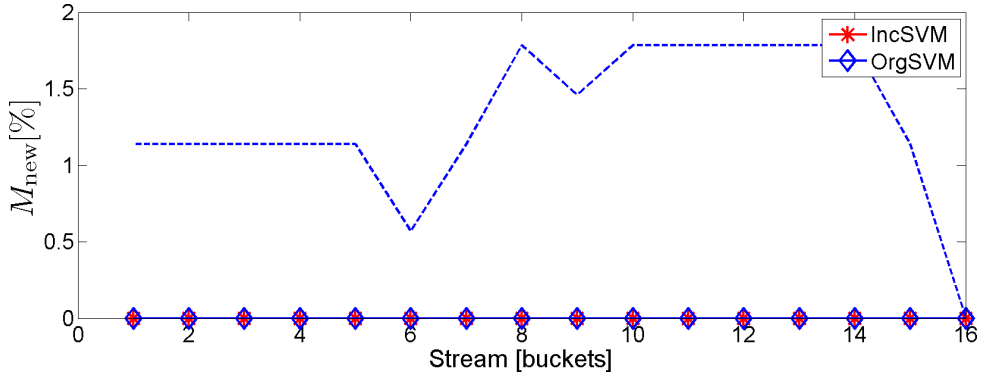


(a) T_{update}

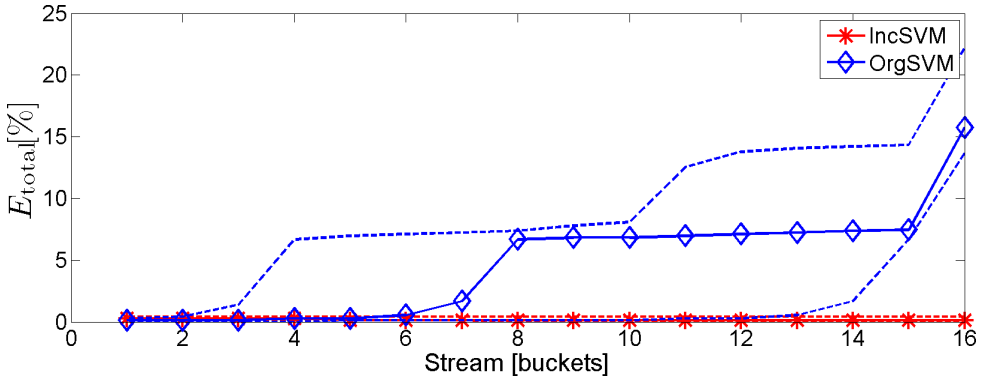
Figure 5.12: The performance metric of the proposed incremental SVM (IncSVM) and the original SVM (OrgSVM) classifiers. Both classifiers are initially trained on 50% of the gesture dataset. (a) The required update time for each bucket (T_{update}), i.e. classification and retraining. (b) Samples falsely identified as novel by each classifier (F_{new}). (c) The rate of missed novelties (M_{new}). (d) The total error rate (E_{total}).



(b) F_{new}



(c) M_{new}



(d) E_{total}

Figure 5.12: (Continued)

6

Contribution: Semi-supervised Methods Based on Non-linear Classifiers

This chapter describes the Extreme Learning Machine (ELM) and the polynomial classifier. Both of them use a non-linear transformation to map the data from the feature space to space where the data might be separated linearly. All calculations in the new space are computed linearly. These classifiers have many common features, but they use a different transformation. The extreme learning machine connects the inputs to hidden neurons (mapping the input features to the hidden layer space) through arbitrary weights while the hidden neurons are connected to the outputs through optimised weights. The polynomial classifier maps the input features using the polynomial series, and the parameters of the series should be optimised. These non-linear transformations make these classifiers appropriate for non-linearly separable data. They can simulate any classification function, particularly when the degree of the polynomial or the number of hidden layers is increased in the polynomial and ELM classifiers, respectively.

This chapter has been adapted and/or adopted from: (Al-Behadili et al., 2015a,e, 2016c,d)

6.1 Semi-Supervised Extreme Learning Machine

Extreme learning machine (ELM) is a type of single layer feed-forward network (SLFN), where the input is randomly mapped to the hidden neurons and the output is computed analytically (Huang et al., 2006b). The ELM tends to achieve good generalisation performance at exceedingly fast learning speed in comparison with conventional neural networks. Originally, it has been an SLFN but some extensions of the ELM use two or more layers (Cambria et al., 2013). According to Huang et al. (2006b), for an SLFN with

n hidden neurons the output is given by

$$f_{elm}(\vec{x}_j) = \sum_{i=1}^{n} \boldsymbol{\beta}_{\mathrm{E}i} G_{\mathrm{h}}(\vec{a}_i, b_i, \vec{x}_j) \quad \vec{x}_j \in \mathbb{R}^d, \vec{a}_i \in \mathbb{R}^d, b_i \in \mathbb{R} \tag{6.1}$$

with $\boldsymbol{\beta}_{\mathrm{E}i}$ is the ELM output weight, $\vec{a}_i$ and b_i are the learning parameters, $G_{\mathrm{h}}(\vec{a}_i, b_i, \vec{x}_j)$ is the output of the i-th hidden neuron, and $\vec{x}_j$ is the feature vector associated with training sample j. It is

$$G_{\mathrm{h}}(\vec{a}_i, b_i, \vec{x}_j) = g(\vec{a}_i \cdot \vec{x}_j + b_i), \tag{6.2}$$

with $\vec{a}_i$ and b_i as the ith neuron's input weight vector and bias. For hidden neurons with radial basis function (RBF) characteristic it is,

$$G_{\mathrm{h}}(\vec{a}_i, b_i, \vec{x}_j) = g(b_i \, \|\vec{x}_j - \vec{a}_i\|) \quad b_i \in \mathbb{R}^+ \tag{6.3}$$

with $\vec{a}_i$ and b_i as the centre and width of RBF neuron i. $\mathbb{R}^+$ refers to all positive real numbers (Huang et al., 2006b).

The ELM is an SLFN network, and the equations above apply to it. Following Huang et al. (2006b), suppose we have N arbitrary distinct samples ($\vec{x}_j \in \mathbb{R}^d$, $\vec{t}_j \in \mathbb{R}^{N_{\text{class}}}$) consisting of a feature vector $\vec{x}_j$ and a target vector $\vec{t}_j$ containing one value for each of the N_{class} classes, respectively. Notably, it is $\vec{t}_j = +1$ for the output neuron belonging to the class of the sample and $\vec{t}_j = -1$ for the other output neurons, respectively. An error-free approximation of the N samples by this n neurons ELM then implies the existence of a set of parameters $\boldsymbol{\beta}_{\mathrm{E}i}$, $\vec{a}_i$ and b_i fulfilling

$$f_{elm}(\vec{x}_j) = \vec{t}_j, j = 1 \cdots N, \tag{6.4}$$

which can be written as a matrix

$$\boldsymbol{H}_{\mathrm{E}} \cdot \boldsymbol{\beta}_{\mathrm{E}} = \boldsymbol{T}_{\mathrm{E}} \tag{6.5}$$

$$\boldsymbol{H}_{\mathrm{E}} = \begin{bmatrix} G(\vec{a}_1, b_1, \vec{x}_1) & \cdots & G(\vec{a}_n, b_n, \vec{x}_1) \\ \vdots & \ddots & \vdots \\ G(\vec{a}_1, b_1, \vec{x}_N) & \cdots & G(\vec{a}_n, b_n, \vec{x}_N) \end{bmatrix}_{N \times n}, \tag{6.6}$$

$$\boldsymbol{\beta}_{\mathrm{E}} = \begin{bmatrix} \boldsymbol{\beta}_{\mathrm{E}1}^T \\ \vdots \\ \boldsymbol{\beta}_{\mathrm{E}n}^T \end{bmatrix}_{n \times N_{\text{class}}} \quad \text{and} \quad \boldsymbol{T}_{\mathrm{E}} = \begin{bmatrix} \vec{t}_1^T \\ \vdots \\ \vec{t}_N^T \end{bmatrix}_{N \times N_{\text{class}}} \tag{6.7}$$

where $\boldsymbol{\beta}_{\mathrm{E}i}$ denotes the vector containing the ith neuron's output weight for all classes and the matrix $\boldsymbol{H}_{\mathrm{E}}$ denotes the hidden layer output. According to Huang et al. (2006b), the procedure of training the ELM is as follows:

- The first step is to assign the input parameters (i.e. $\vec{a}_i$, and b_i, $i = 1, \ldots, n$) randomly.
- Analytical computation of the matrix $\boldsymbol{H}_{\mathrm{E}}$ is performed according to (Eq. (6.7)).
- The output weights are then estimated using (Eq. (6.5)).

As shown by Huang et al. (2006b), the problem here is to minimize the error in (Eq. (6.5)), i.e. $\|\boldsymbol{H}_{\mathrm{E}} \cdot \boldsymbol{\beta}_{\mathrm{E}} - \boldsymbol{T}_{\mathrm{E}}\|$. Since (Eq. (6.5)) is a linear system in the output weights, the output weights are estimated by Huang et al. (2006b) using the pseudo-inverse of the output matrix of the hidden layer according to ${\boldsymbol{H}_{\mathrm{E}}}^{\dagger} = ({\boldsymbol{H}_{\mathrm{E}}}^{T}\boldsymbol{H}_{\mathrm{E}})^{-1}{\boldsymbol{H}_{\mathrm{E}}}^{T}$ (Rao and Mitra, 1971):

$$\hat{\boldsymbol{\beta}}_{\mathrm{E}} = {\boldsymbol{H}_{\mathrm{E}}}^{\dagger} \cdot \boldsymbol{T}_{\mathrm{E}} \tag{6.8}$$

As suggested by Huang et al. (2006b), the singular value decomposition (SVD) (Rao and Mitra, 1971) is used to compute the pseudo-inverse ${\boldsymbol{H}_{\mathrm{E}}}^{\dagger}$. The labels of the new samples can then be obtained by using the estimates $\hat{\boldsymbol{\beta}}_{\mathrm{E}}$ and ${\boldsymbol{H}_{\mathrm{E}}}^{\dagger}$ in (Eq. (6.7)).

Here, we propose a method to update the ELM incrementally and to apply the novelty detection using the EVT and the confidence band to the output of the ELM to reject the unknown samples (i.e. outliers or samples belong to novel concepts). The proposed Semi-Supervised Extreme Learning Machine (SSELM) consists of two phases.

6.1.1 Incremental learning phase

The incremental updating rule is derived based on the pseudo-inverse method introduced by Lan et al. (2009) according to

$$\boldsymbol{M}_{\mathrm{E}} = {\boldsymbol{H}_{\mathrm{E}}}^{T}\boldsymbol{H}_{\mathrm{E}} \text{ and } \boldsymbol{P}_{\mathrm{E}} = {\boldsymbol{H}_{\mathrm{E}}}^{T}\boldsymbol{T}_{\mathrm{E}}. \tag{6.9}$$

$$\text{Hence,} \qquad \boldsymbol{\beta}_{\mathrm{E}} = {\boldsymbol{M}_{\mathrm{E}}}^{-1}\boldsymbol{P}_{\mathrm{E}}. \tag{6.10}$$

The dimension of $\boldsymbol{M}_{\mathrm{E}}$ is $n \times n$ and the dimension of the $\boldsymbol{P}_{\mathrm{E}}$ is $n \times N_{\text{class}}$, where n corresponds to the number of hidden neurons and N_{class} to the number of classes.

Suppose that we have $N_{(0)}$ labelled samples for initial training. We then compute $\boldsymbol{M}_{\mathrm{E}(0)} = \boldsymbol{H}_{\mathrm{E}(0)}^{T}\boldsymbol{H}_{\mathrm{E}(0)}$ and $\boldsymbol{P}_{\mathrm{E}(0)} = \boldsymbol{H}_{\mathrm{E}(0)}^{T}\boldsymbol{T}_{\mathrm{E}(0)}$ according to (Eq. (6.9)). Hence, $\boldsymbol{\beta}_{\mathrm{E}(0)} = \boldsymbol{M}_{\mathrm{E}(0)}^{-1}\boldsymbol{P}_{\mathrm{E}(0)}$.

Incremental learning is achieved by adding chunks of samples to the training set. If the number of samples $\vec{x}'$ in the chunk $k+1$ is N' then the hidden layer output matrix $\boldsymbol{H}'_{\mathrm{E}}$ corresponding to the new chunk of data is

$$\boldsymbol{H}'_{\mathrm{E}} = \begin{bmatrix} G_{\mathrm{h}}(\vec{a}_1, b_1, \vec{x}'_1) & \cdots & G_{\mathrm{h}}(\vec{a}_n, b_n, \vec{x}'_1) \\ \vdots & \ddots & \vdots \\ G_{\mathrm{h}}(\vec{a}_1, b_1, \vec{x}'_{N'}) & \cdots & G_{\mathrm{h}}(\vec{a}_n, b_n, \vec{x}'_{N'}) \end{bmatrix}_{N' \times n} \tag{6.11}$$

From (Eq. (6.9)) and (Eq. (6.11)) it follows that $\boldsymbol{M}'_{\mathrm{E}} = \boldsymbol{H}'^{T}_{\mathrm{E}} \boldsymbol{H}'_{\mathrm{E}}$ and $\boldsymbol{P}'_{\mathrm{E}} = \boldsymbol{H}'^{T}_{\mathrm{E}} \boldsymbol{T}'_{\mathrm{E}}$ corresponding to the new chunk data. Then

$$\boldsymbol{M}_{\mathrm{E}(k+1)} = \boldsymbol{M}_{\mathrm{E}(k)} + \boldsymbol{M}'_{\mathrm{E}} \text{ and } \boldsymbol{P}_{\mathrm{E}(k+1)} = \boldsymbol{P}_{\mathrm{E}(k)} + \boldsymbol{P}'_{\mathrm{E}} \tag{6.12}$$

Finally, using (Eq. (6.10)) the updated output is $\boldsymbol{\beta}_{\mathrm{E}(k+1)} = \boldsymbol{M}^{-1}_{\mathrm{E}(k+1)} \boldsymbol{P}_{\mathrm{E}(k+1)}$.

6.1.2 Novelty detection phase

Novelty detection using EVT

According to Huang et al. (2006b), the output of the ELM is around +1 for the class that the sample belongs to and around −1 for the other classes. This assignment follows immediately from the target values used for the training of the ELM. If the output of the winner class is exactly +1, then this result is similar to the training data and thus highly believable. The confidence of the result decreases with an increasing distance of the winner output class from the ideal value of +1. Furthermore, the linear least squares optimisation applied in the training yields mean-free normally distributed residuals. Consequently, the absolute difference between the ELM output and the ideal value, i.e. a vector that contains +1 at the position of the winning neuron and −1 at all other positions, will originate from a mean free one-sided normal distribution. Additionally, we divide the absolute difference by the standard deviation of the residuals, i.e. the square root of squared residuals mean, to arrive at a $\mathcal{N}(0,1)$ distribution.

Recalling (Eq. (6.5)) and substituting (Eq. (6.8)), we arrive at the prediction $\hat{\boldsymbol{T}}_{\mathrm{E}}$ of the training set

$$\hat{\boldsymbol{T}}_{\mathrm{E}} = \boldsymbol{H}_{\mathrm{E}} (\boldsymbol{H}_{\mathrm{E}}^{T} \boldsymbol{H}_{\mathrm{E}})^{-1} \boldsymbol{H}_{\mathrm{E}}^{T} \boldsymbol{T}_{\mathrm{E}}. \tag{6.13}$$

Notably, each column of $\hat{\boldsymbol{T}}_{\mathrm{E}}$ contains the predicted values for one class. Let $\vec{\hat{t}}_c$ and $\vec{t}_c$ be the vector containing the predicted values and the target values of class c, respectively. The sum of the squared residuals is then given by

$$r_c = \left[\vec{\hat{t}}_c - \vec{t}_c\right]^T \left[\vec{\hat{t}}_c - \vec{t}_c\right] \tag{6.14}$$

$$= \vec{t}_c^{\,T} \boldsymbol{H}_{\mathrm{E}} (\boldsymbol{H}_{\mathrm{E}}{}^T \boldsymbol{H}_{\mathrm{E}})^{-1} \boldsymbol{H}_{\mathrm{E}}{}^T \vec{t}_c + \vec{t}_c^{\,T} \vec{t}_c. \tag{6.15}$$

Notably, $\boldsymbol{H}_{\mathrm{E}}{}^T \boldsymbol{H}_{\mathrm{E}} = \boldsymbol{M}_{\mathrm{E}}$ and $\boldsymbol{H}_{\mathrm{E}}{}^T \vec{t}_c$ is the cth column of $\boldsymbol{P}_{\mathrm{E}}$. Consequently, the first summand on the right side of (Eq. (6.15)) may be incremented using (Eq. (6.12)). The term $\vec{t}_c^{\,T} \vec{t}_c$ is the sum of the squared target values. Consequently, it may be incremented by adding the squared target values of additional samples. Dividing r_c by the number of samples and taking the square root results in the standard deviation of the residuals. Notably, this approach yields a class-wise standard deviation which represents the different models formed by the output layer of the ELM.

After division by the class-wise standard deviation, the absolute difference of the ELM output and the ideal value originates from a one-sided normal distribution and is thus modelled by the Gumbel distribution of the extreme value theory (Clifton et al., 2008). Accordingly, the highly believable samples have $P_{\mathrm{ev}} = 0$ and the ideal novel sample yields $P_{\mathrm{ev}} = 1$. Thus we set $P_{\mathrm{th}} = 0.9$ and flag a sample to be novel if at least two output neurons detect a novelty.

Novelty detection using confidence bands

Equation (Eq. (6.5)) is a system of linear equations for the output weights, i.e. the output of the ELM is a weighted linear combination of the hidden layer activations, where the weights represent the parameters of a linear model (Huang et al., 2006b). The confidence band intervals of the output decision may thus be estimated. They can be computed by applying the output of the classifier to Eq. (3.24). Notably, the residual r_i is the difference between the estimated output $\vec{\hat{t}}(\vec{x}_i)$ and the target output $\vec{t}(\vec{x}_i)$. Here, the number of free parameters N_{p} corresponds to the number of neurons. Since $\boldsymbol{H}_{\mathrm{E}}{}^T \boldsymbol{H}_{\mathrm{E}} = \boldsymbol{M}_{\mathrm{E}}$ and the residuals $r_i = \vec{\hat{t}}(\vec{x}_i) - \vec{t}(\vec{x}_i)$ appear in the squared sum, (Eq. (6.12)) and (Eq. (6.15)) are applied to update the confidence bands incrementally. The standard value of $\alpha_{\mathrm{conf}} = 0.05$ (Kardaun, 2005) is used.

The sample $\vec{x}$ is considered novel if the inequality

$$\vec{\hat{t}}_1(\vec{x}) - \vec{\hat{t}}_2(\vec{x}) < z \cdot (\eta_{\mathrm{conf}1}(\vec{x}) + \eta_{\mathrm{conf}2}(\vec{x})) \tag{6.16}$$

is fulfilled, where $\vec{\hat{t}}_1(\vec{x})$ and $\vec{\hat{t}}_2(\vec{x})$ correspond to the first and second largest decision values in the output of the ELM classifier for the sample $\vec{x}$ and $\eta_{\mathrm{conf}1}(\vec{x})$ and $\eta_{\mathrm{conf}2}(\vec{x})$ are the confidence band widths of the $\vec{\hat{t}}_1(\vec{x})$ and $\vec{\hat{t}}_2(\vec{x})$, respectively (condition (Eq. (6.16)) is proposed by Sakic (2012) in the context of semi-supervised learning). z is a constant (here z set to 75).

Finally, the sample $\vec{x}$ has been considered novel if both conditions Eq. (6.16) and at least two output neurons are greater than P_{th}, which correspond to the confidence band and EVT, respectively, indicate it as a novel.

6.1.3 Believability conditions

To ensure only trusted labels in the new training set, we apply additional conditions. If at least one output neuron detects a novelty, we do not consider the label trustworthy and do not add it to the training set. Furthermore, since the winner class output value, in the ideal case, should be $+1$ and all other neurons output -1, the difference between the winning neuron and the second largest output value is supposed to be 2. We have noticed that the ELM outputs two positive values in the case of unseen classes, resulting in a difference less than 1. Therefore, the newly labelled samples should fulfil another condition to be accepted in the next training phase: The difference between the first and second largest output values should exceed a specific threshold. Here, we set the threshold to 1.

6.1.4 Experiments and results

The normal ELM neural network has been proven to be a fast neural network (Huang et al., 2006b). Moreover, the incremental ELM is faster than normal ELM. Hence, the main comparison will focus on the accuracy rather than the time of processing. To show the additional features of the proposed algorithm we compare its results with the auto-encoder neural network (cf. Section A.2). The AANN from the dd-tools toolbox (Tax, 2015) are used. Similar to our algorithm, this auto-encoder algorithm can detect outliers. Hence, we used it in the semi-supervised process to compare the two algorithms.

Only the Gesture data are used to evaluate the classifiers. The Gesture data are randomly divided into three disjoint data sets with fractions 40%, 40%, and 20% for the training, the learning and the test set, respectively. As mentioned in Section 5.2, the training set contains all nine classes, and each class is divided separately, i.e. the training set contains 40% of the samples of all classes. The novel class is then introduced by excluding one class from the initial training set. The learning set is split into buckets of 100 samples.

Both classifiers are trained on the initial training set. Then the accuracy and other measures are evaluated based on the test set and all step of the experiment Section 5.2 are implemented. Since the auto-encoder algorithm is not incremental, the algorithm is retrained using the whole modified training set while we use the proposed incremental update rule for the ELM. New samples are considered novel if the novelty flag set, i.e. the EVT output exceeds the threshold of 0.9 and Eq. (6.16) is fulfilled. The believability flag controls the selection of the samples that are included in the training set. This flag is

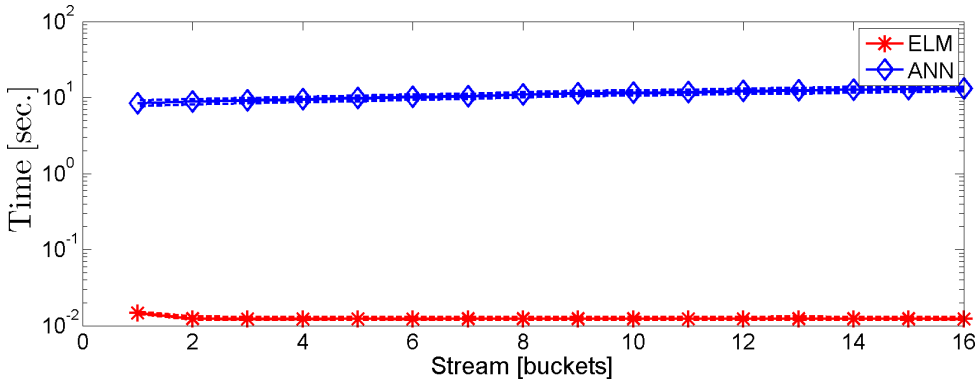
set if the EVT does not indicate it novel and Eq. (6.16) is not fulfilled, and the difference between the winner class and the second class exceeds 1.

The auto-encoder labels the new sample with the winner class label or as an "outlier". Originally, the auto-encoder is a one-class classifier. However, using the function "multic" in the toolbox by Tax (2015) allows for the classification of multiple classes. Training one classifier for each class and applying all of these classifiers to the "multic" function, achieves a multi-class auto-encoder. Each classifier outputs a real number between 0 and 1 similar to a probability. If this number is less than a predefined threshold, which was set by selecting the ratio of the outliers in the training set to be 5%, it indicates the new sample as an "outlier". Otherwise, the new sample is labelled as "target". The "multic" function labels the new sample as "outlier" if it does not match any class, i.e. if it is indicated as "outlier" by all classifiers, and it labels the new sample with the most probable class, i.e. the maximum output class, if more than one class is labelled as "target"[1].

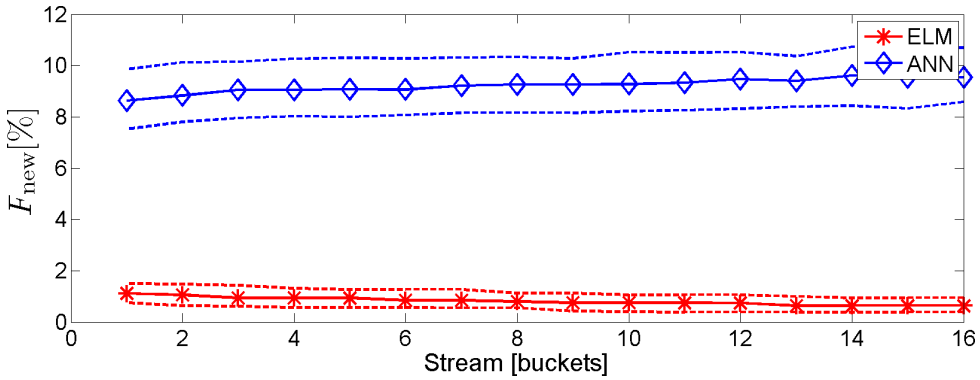
The runtime of the classifier matches the expectations (Fig. 6.1(a)). Since the proposed algorithm is incremental, it requires less time to adapt. In any case, the processing time of the incremental ELM is of the order of some milliseconds which helps to apply it to online data streams.

Fig. 6.1(c) shows that the values M_{new} of both algorithms are zero, i.e. no outlier or novelty has been missed. This is important since the outliers are supposed to be near the boundary of the sample distributions of all classes and thus accepting them significantly affects the performance of the classifiers in next iterations. Fig. 6.1(b) shows the values F_{new}, which is initially below 2 % for the proposed algorithm whereas it starts at more than 8 % for the AANN. The rate of false detections by the incremental ELM decreases with an increasing amount of training data reaching a final level of less than 1 %. In the case of the AANN, F_{new} is almost constant. Although this means that a small fraction of about (1–2) % of the samples belonging to known classes are supposed to be outliers and, consequently, they are rejected, this is not critical. In fact, this removes the (1–2) % most extreme samples from the semi-supervised training set and thus prevents possibly false labels or sloppily performed gestures from entering the training set. This leads to a slow gradual adaptation of the learned sample distributions leading to a final stabilised value reflected by E_{total} (Fig. 6.1(d)). This behaviour is favourable in slow concept-drifting data streams where the sample distributions change slowly over time. The auto-encoder, in contrast, rejects more samples, which leads to a very slow adaptation. The effect of this novelty detection is directly apparent in the total error E_{total} of both classifiers. Initially, the error of the proposed approach is less than 2 % and decreases to less than 1 %. On the other side, the total error of the auto-encoder is initially about 8 % and increases with an increasing F_{new}.

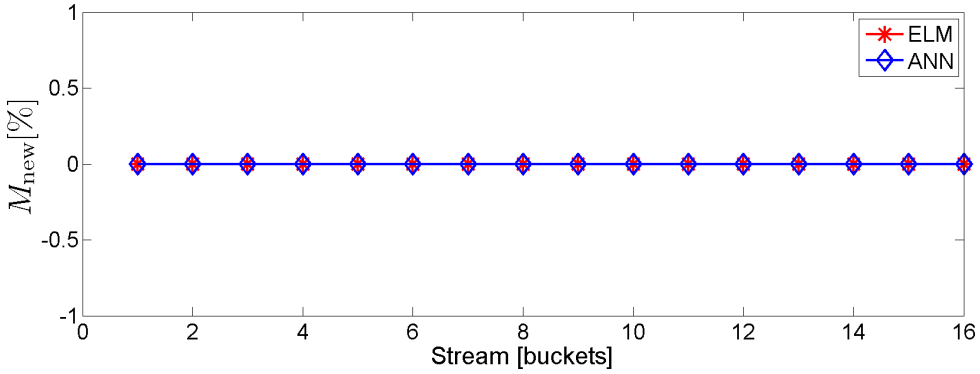
[1]The results of the individual 100 runs are available at http://www.bv.e-technik.tu-dortmund.de



(a) T_{update}: Gesture data

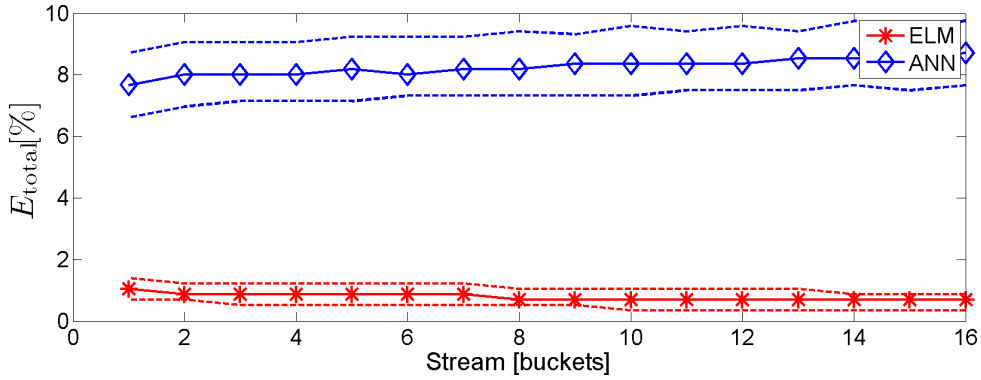


(b) F_{new}: Gesture data



(c) M_{new}: Gesture data

Figure 6.1: Accuracy metrics the incremental ELM and the AANN classifiers: (a) The update time required for each bucket T_{update}, i.e. classification and retraining. (b)The rate of falsely detected novelties F_{new}. (c) The rate of missed novelties M_{new}. (d) The total error rate E_{total}



(d) E_{total}: Gesture data

Figure 6.1: (Continued)

6.2 Auto-encoder Extreme Learning Machine

Another implementation of the ELM is proposed here. The system uses two different ELM classifiers (Huang et al., 2006b). The first model is trained by using the training data with their target labels. This model is used to assign the test samples. It is, however, not able to detect the uncommon data without additional heuristics or methods. To detect uncommon data, we apply a novelty detection based on an extreme learning machine trained as an auto-encoder (AE-ELM), i.e. the second ELM is not a classifier but a regression-based ELM that tries to reproduce its input. Based on the training data, we estimate the distribution of the residuals and test new data based on the hypothesis that it belongs to a different distribution. The AE-ELM and the corresponding novelty detection method are described Section 6.2.1 and 6.2.2, respectively.

Since the training and testing streamed data may be observed in different environments and thus the noise levels and the residual artefacts of the features may be different for each observation. Consequently, both classifiers are required to adapt themselves to the new distribution. This is achieved in a fully autonomous manner using the incremental learning techniques.

6.2.1 The auto-associative extreme learning machine

Auto-associative neural networks (AANN) (Cambria et al., 2013) are neural networks that are trained to reproduce the training data, i.e. the inputs and the targets are identical. In general, they minimize the sum of squared norms

$$\sum_i \left\| \boldsymbol{H}_{\text{E}}(\vec{x}_i) \cdot \boldsymbol{\beta}_{\text{AE}} - \vec{x}_i \right\|^2, \tag{6.17}$$

where $\vec{x}_i$ is the vector containing the ith sample's input features, $\boldsymbol{H}_{\mathrm{E}}(\vec{x}_i)$ is the matrix containing the activation of the networks output layer, and $\boldsymbol{\beta}_{\mathrm{AE}}$ is the vector containing the output weights of the neural network. Consequently, they learn the data representation (Hertz et al., 1991; Baldi and Hornik, 1989), i.e. the input pattern is reconstructed at the AANN's output layer with minimum error. Here, we use the ELM, which is detailed in 6.1 because it is easily adapted to new data (Al-Behadili et al., 2015a). One hidden layer with n hidden units is used in the architecture of this auto-encoder. The sigmoid activation function is used for the hidden neurons. Since the parameters of the hidden neurons are fixed for the ELM, only the output weights are adapted and Eq. (6.17) may be solved using linear least-squares techniques.

Using the auto-encoder ELM, we avoid the problems of the conventional AANN, which also optimize the hidden neurons and thus require additional parameters of the non-linear optimization procedure, e.g. a learning rate, and stopping criteria, from an expert user. Commonly, the number of the neurons in the hidden layer of AANN is less than the number of neurons in the input layer. Hence, we tried a different number of neurons and we found that with an increasing number of neurons the amount of detected novel samples are increasing as well. Due to the model's increasing degree of freedom, it is able to reconstruct even noisy samples. This leads to a high number of noisy samples being detected as a novelty. The best numbers of the AE-ELM's hidden neurons (based on (8–10)-dimensional input data) that reduce the noise are in the range of 3–14.

6.2.2 Novelty detection

Since linear least-squares estimators are bias-free, the average overall residuals of the AANN is zero. Furthermore, the residuals follow a univariate normal distribution and the variance is the sum of squared residuals, i.e. the residual error. Unfortunately, the residuals of the AANN do not distinguish between different samples, i.e. each feature of one sample is one residual, respectively. We thus assume, that the residual vectors $\vec{r}_i$ are drawn from a multivariate normal distribution, i.e.

$$\vec{r}_i = \boldsymbol{H}_{\mathrm{E}}(\vec{x}_i) \cdot \boldsymbol{\beta}_{\mathrm{AE}} - \vec{x}_i \sim \mathcal{N}\left(\vec{\mu}_{\mathrm{res}}, \boldsymbol{\Sigma}_{\mathrm{res}}\right). \tag{6.18}$$

where $\vec{\mu}_{\mathrm{res}}$ and $\boldsymbol{\Sigma}_{\mathrm{res}}$ are the mean vector and the covariance matrix of the multivariate normal distribution $\mathcal{N}$, respectively. The novel sample is supposed to reconstructed with larger residuals than the usual samples, i.e. the Mahalanobis distance is greatly increased. The Mahalanobis distance d_{Mahal} of an d-dimensional multivariate normal distribution follows the chi-squared distribution χ^2_d (Gatignon, 2010), where d is the number of features per one sample. An ideal threshold would thus be given by the critical value of the chi-squared distribution given a target percentage. However, we do not know the true values of $\vec{\mu}_{\mathrm{res}}$ and $\boldsymbol{\Sigma}_{\mathrm{res}}$. Consequently, we estimate them from the N training samples. To reflect

this uncertainty, χ^2_d is replaced by Hotelling's T-squared distribution $T_{d,N}$ (Gatignon, 2010). The threshold N_{th} is then computed using

$$N_{\text{th}} = \theta_{\text{drift}} \cdot T^{-1}_{d,N}(\theta_{\text{prob}}) \tag{6.19}$$

where $T^{-1}_{d,N(\theta_{\text{prob}})}$ is the critical value of Hotelling's T-squared distribution with respect to the probability θ_{prob}. Furthermore, we introduce the factor θ_{drift} to cover the concept's movement in the feature space. Although the same effect could be achieved by increasing θ_{prob}, we prefer the two parameters representation to preserve the statistical meaning of θ_{prob} and handle the concept drift using θ_{drift}. For numerical reasons, we do not directly compute Hotelling's T-squared distribution but rather compute the F-distribution, which is commonly tabulated in numerical programming languages, and use the relation

$$\frac{N-d}{(N-1)\,d} T_{d,N} = F_{d,N-d} \tag{6.20}$$

Formally, the same effect could be modelled using the extreme value theory (cf. Section 6.1). Clifton et al. (2011) showed that the extreme valued probability follows equiprobable contours of the generating distribution, which correspond to equidistant contours of the Mahalanobis distance. Consequently, it is possible to derive an extreme value probability, i.e. a probability that the considered sample is an extreme value of the generating distribution, that correspond to our novelty detection threshold. However, we increased the threshold by a small factor θ_{drift} to follow the concept drift and thus loses its statistical meaning of a probability, therefore we prefer the distance based threshold representation.

6.2.3 Incremental learning of ELM

Since the classifier has two models of the ELM, they are updated separately. The normal ELM is updated as same as the updating of the normal ELM that is explained in Section 6.1.1. The second model is similar to the first model except the $\boldsymbol{P}_{\text{E}}$ in Eq. (6.9) is changed to

$$\boldsymbol{P}_{\text{AE}} = {\boldsymbol{H}_{\text{E}}}^T \boldsymbol{X}. \tag{6.21}$$

thus,

$$\boldsymbol{\beta}_{\text{AE}} = {\boldsymbol{M}_{\text{E}}}^{-1} \boldsymbol{P}_{\text{AE}}. \tag{6.22}$$

The dimension of the matrix $\boldsymbol{P}_{\text{AE}}$ is $n \times d$. Since the dimension of these matrices only depends on the number hidden neurons n and the number of known classes d, they are independent on the number of samples used to train the classifier. Incrementing these classifiers does not require all trained samples and thus does not require a huge amount of

computer memory if the number of samples, which may be up to several millions in some cases, increases. The incremental updating of these parameters is similar to the normal ELM also (cf. Section 6.1.1).

Suppose that we have $N_{(0)}$ labelled samples for initial training. Then the initial mean vector $\vec{\mu}_{\text{res}(0)}$ and the initial covariance matrix $\mathbf{\Sigma}_{\text{res}(0)}$ of the residuals are estimated according to

$$\vec{\mu}_{\text{res}(0)} = \frac{1}{N_{(0)}} \sum_{i=1}^{N_{(0)}} \vec{r}_{i,(0)}, \text{ and} \tag{6.23}$$

$$\mathbf{\Sigma}_{\text{res}\,\vec{r},(0)} = \frac{1}{N_{(0)}-1} \sum_{i=1}^{N_{(0)}} \left(\vec{r}_{i,(0)} - \vec{\mu}_{\text{res}(0)}\right)\left(\vec{r}_{i,(0)} - \vec{\mu}_{\text{res}(0)}\right)^T, \tag{6.24}$$

where $\vec{r}_{i,(0)}$ is the auto-encoder's residual of the ith sample in the initial training set. Incremental learning is then achieved by adding chunks of samples to the training set. If the number of samples $\vec{x}'$ in the chunk $k+1$ is N' then the parameters of the normal distribution, which models the residuals are adapted according to

$$N_{(k+1)} = N_{(k)} + N', \tag{6.25}$$

$$\vec{\mu}_{\text{res}(k+1)} = \frac{N_{(k)}}{N_{(k)}+N'} \vec{\mu}_{\text{res}(k)} + \frac{N'}{N_{(k)}+N'} \vec{\mu}_{\text{res}}{}', \text{ and} \tag{6.26}$$

$$\mathbf{\Sigma}_{\text{res}(k+1)} = \frac{1}{N_{(k)}+N'-1} \Bigg[\left(N_{(k)}-1\right) \mathbf{\Sigma}_{\text{res}(k)} + \left(N'-1\right) \mathbf{\Sigma}_{\text{res}}{}' + \frac{N_{(k)}N'}{N'_{(k)}+N'} \left(\vec{\mu}_{\text{res}(k)} - \vec{\mu}_{\text{res}}{}'\right)\left(\vec{\mu}_{\text{res}(k)} - \vec{\mu}_{\text{res}}{}'\right)^T \Bigg], \tag{6.27}$$

where $\vec{\mu}_{\text{res}}{}'$ and $\mathbf{\Sigma}_{\text{res}}{}'$ are the mean vector and the covariance matrix estimates based on the new chunk of data. The Eq. (6.27) will be discussed in more details in Section 7.1.

To avoid falsely labelled samples in the new chunk of data, the new data must match two requirements. First, the samples need to pass the novelty detection check. Second, the sample must be classified with good confidence. The first condition is checked for the Mahalanobis based distance threshold of the ELM auto-encoder (see Section 6.2.2). The second condition requires the ELM classifier to assign a believability label to the new sample. We consider a label trustworthy if the outputs of the winning class and the second largest output increase a threshold of $N_{\text{th}b}$. All samples that pass the two requirements are selected and both classifiers, i.e. the ELM auto-encoder and the ELM classifier are updated.

6.2.4 Results and performance study

This method is evaluated by using two databases. The gesture database is used to evaluate the incremental performance and the novelty detection accuracy for gesture data. The accuracy of detecting a small number of novel samples within a large number of data is evaluated by using the Lunar database (cf. Section 4.4).

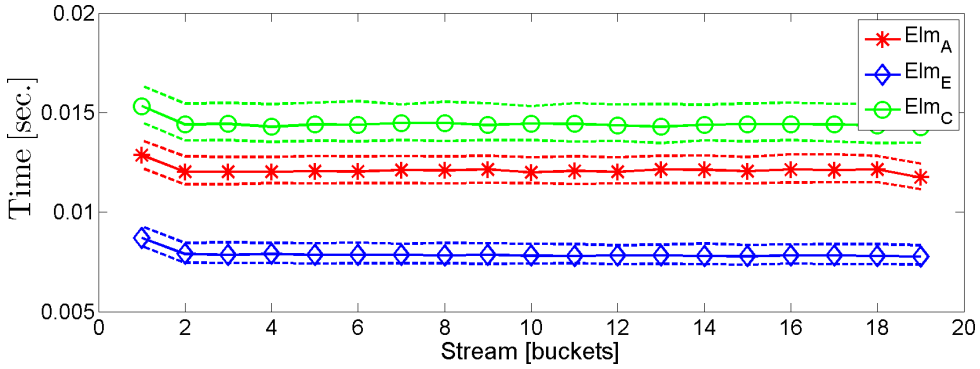
Auto-encoder ELM evaluation in gesture database

The AE-ELM (Elm_A) is compared with other two versions of the previously explained ELM. The first version of the ELM (Elm_E) uses the EVT only for the novelty detection, and the other version (Elm_C) uses the EVT and the confidence band method for the novelty detection. Since all the classifiers can learn incrementally, the data set is divided into training, learning and testing sets only with a fraction of 10%, 60% and 30%, respectively. The rest of this experiment is same as the normal procedure.

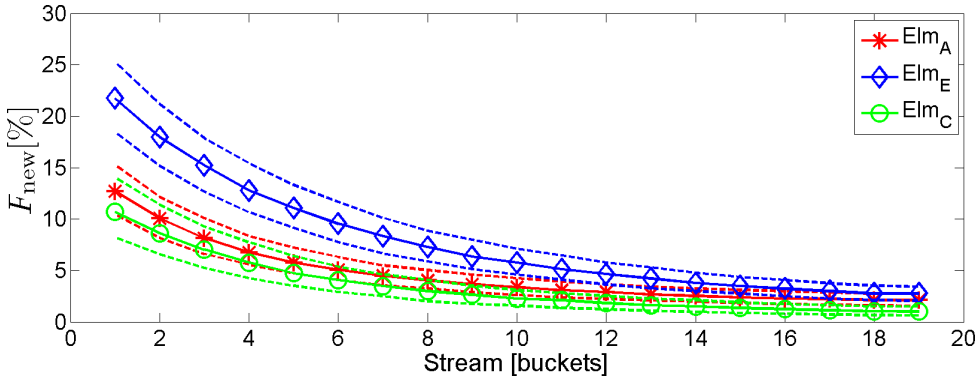
Since all classifiers in the experiment are incrementally learned, the running time of all classifiers is negligible and mostly constant. The AE-ELM is faster than the classifier that computes both the confidence bands and the EVT, but it is slower than the classifier that uses only the EVT (Fig. 6.2(a)). This processing time is expected since the AE-ELM approximately uses two ELM. Although, the median of missed novel samples of the AE-ELM classifier is zero in Fig. 6.2(c), the 75% quantile shows that the missed novel samples increase slightly with the continuous updating. In the same figure, the median of M_{new} of the ELM with the EVT and confidence band novelty detection is zero, while the 75% quantile is also increased with the continuous updating but it still less than the 75% quantile of the AE-ELM. This means the number of outliers that are accepted in the successive training, in the AE-ELM is larger than the other classifiers. Making the condition of the believability more difficult to compensate this increase in the 75% quantile leads to rejecting more normal samples, which affect the total accuracy. Fig. 6.2(b) shows the F_{new} of the three classifiers. The accuracy of indicating the samples that belong to known classes of the AE-ELM is in between the accuracy of the other classifiers. Thus the total error E_{total} (Fig. 6.2(d)) of the AE-ELM is also in the middle of the total error of other two classifiers. The best classifier is the ELM that uses both techniques of novelty detection, EVT and confidence band; while the worst is the ELM that uses only the EVT.

ELM and AE-ELM evaluation in lunar database

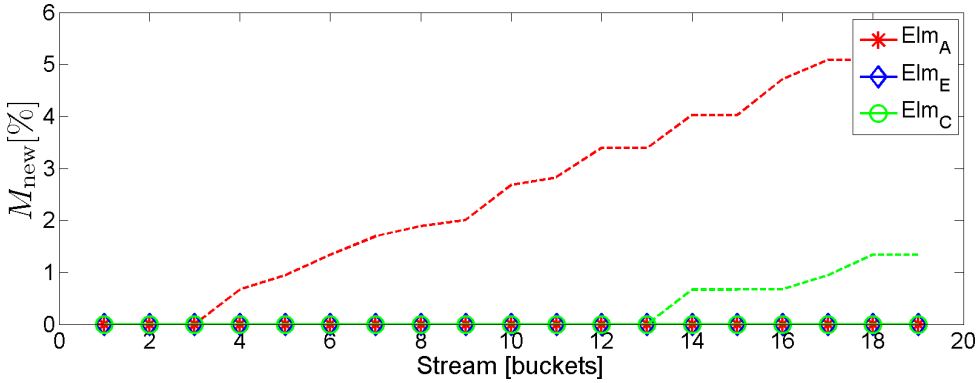
We apply the proposed framework to the western Moscoviense basin, which is described in Section 4.4.2. The output is compared to the result of Pieters et al. (2011). Since the output of the ELM depends on the randomly placed hidden neurons, we train 100 pairs of ELM and AE-ELM. While the ELM assigns a class label to each pixel, the AE-ELM provides a novelty flag. We assume that a pixel is truly a novelty, i.e. an uncommon lunar spectrum, if at least 99 %, i.e. 99 AE-ELMs, independently set the novelty flag. Similarly,



(a) T_{update}: Gesture data

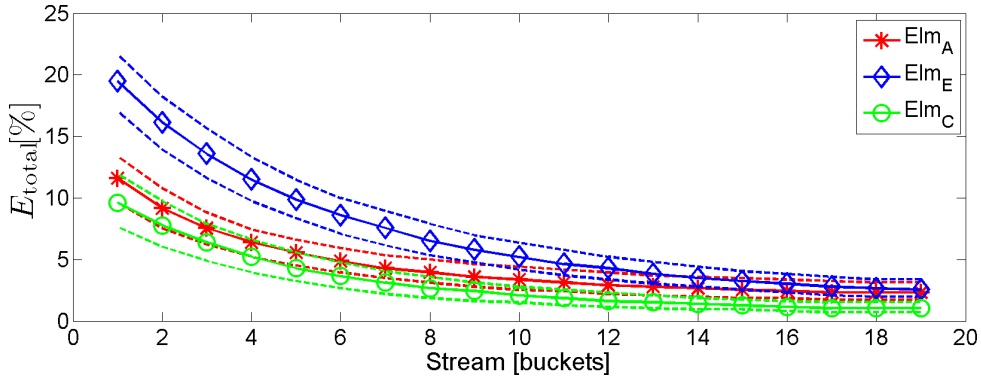


(b) F_{new}: Gesture data



(c) M_{new}: Gesture data

Figure 6.2: Accuracy metrics of the Elm_A, the Elm_E and the Elm_C classifiers: (a) The time required for each bucket T_{update}, i.e. classification and retraining. (b) The rate of falsely detected novelties F_{new}. (c) The rate of missed novelties M_{new}. (d) The total error rate E_{total}.



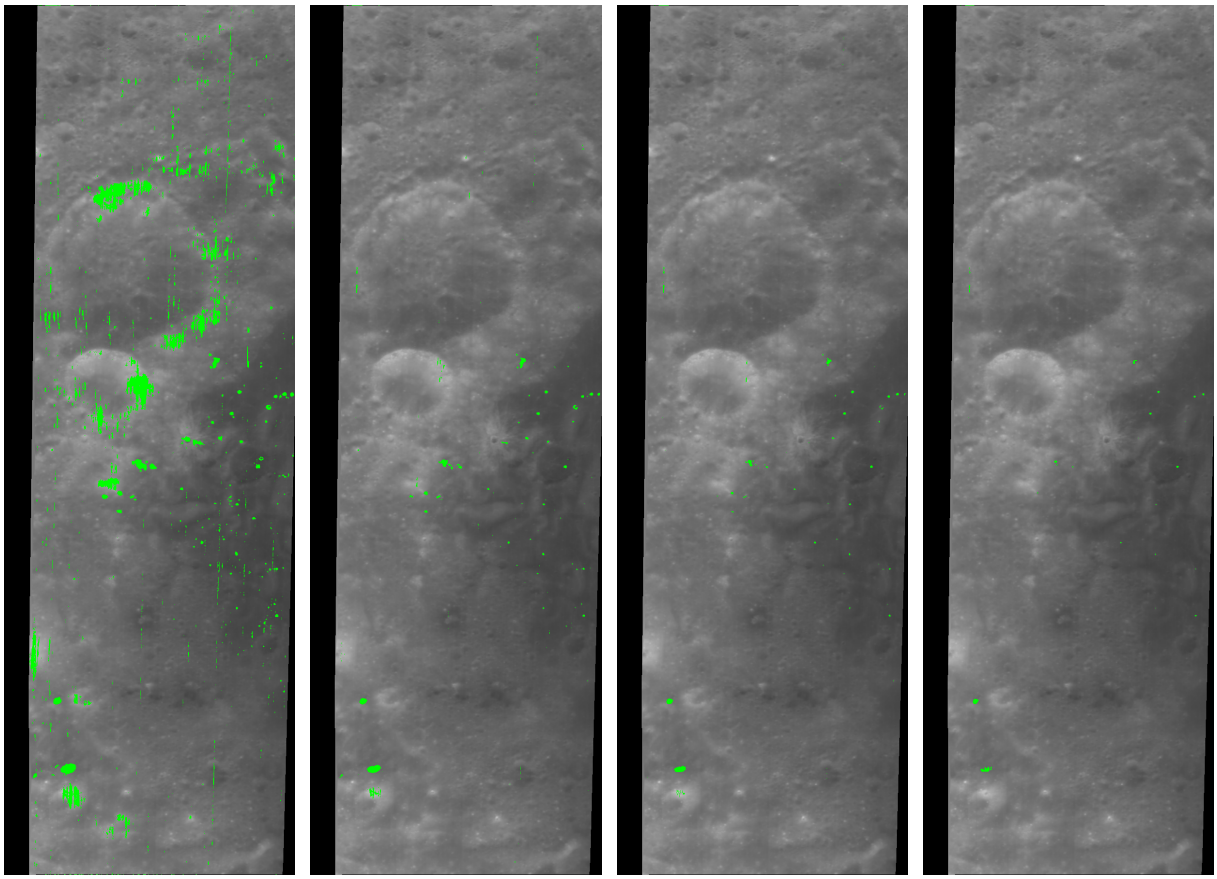
(d) E_{total}: Gesture data

Figure 6.2: (Continued)

we assign a class label based on a majority vote, i.e. we selected the most frequent label. Fig. 6.3 shows the influence of the the novelty detection parameter θ_{drift}. If the threshold is too small, the classifiers are sensitive to noise and detect too many uncommon spectra. In contrast, the classifiers select no novelties if the threshold is set too high. Consequently, the number of novelties may be gradually reduced by increasing θ_{drift}. We set $\theta_{\text{drift}} = 3$ for the further analysis.

The semi-supervised learning process is illustrated in Fig. 6.4. After each semi-supervised increment, the class labels and the novelty flags were computed. While the initial class labels appear rather noisy, the class labels become more homogeneous even after each increment and yield a homogeneous separation of the mare and highland regions. The detected novelties, however, increase with each increment. This results behaviour indicates that, although some classifiers initially miss the novelties, they adapt to the data and thus are able to detect novelties which they initially did not detect. These newly detected novelties are mainly concentrated in the mare region.

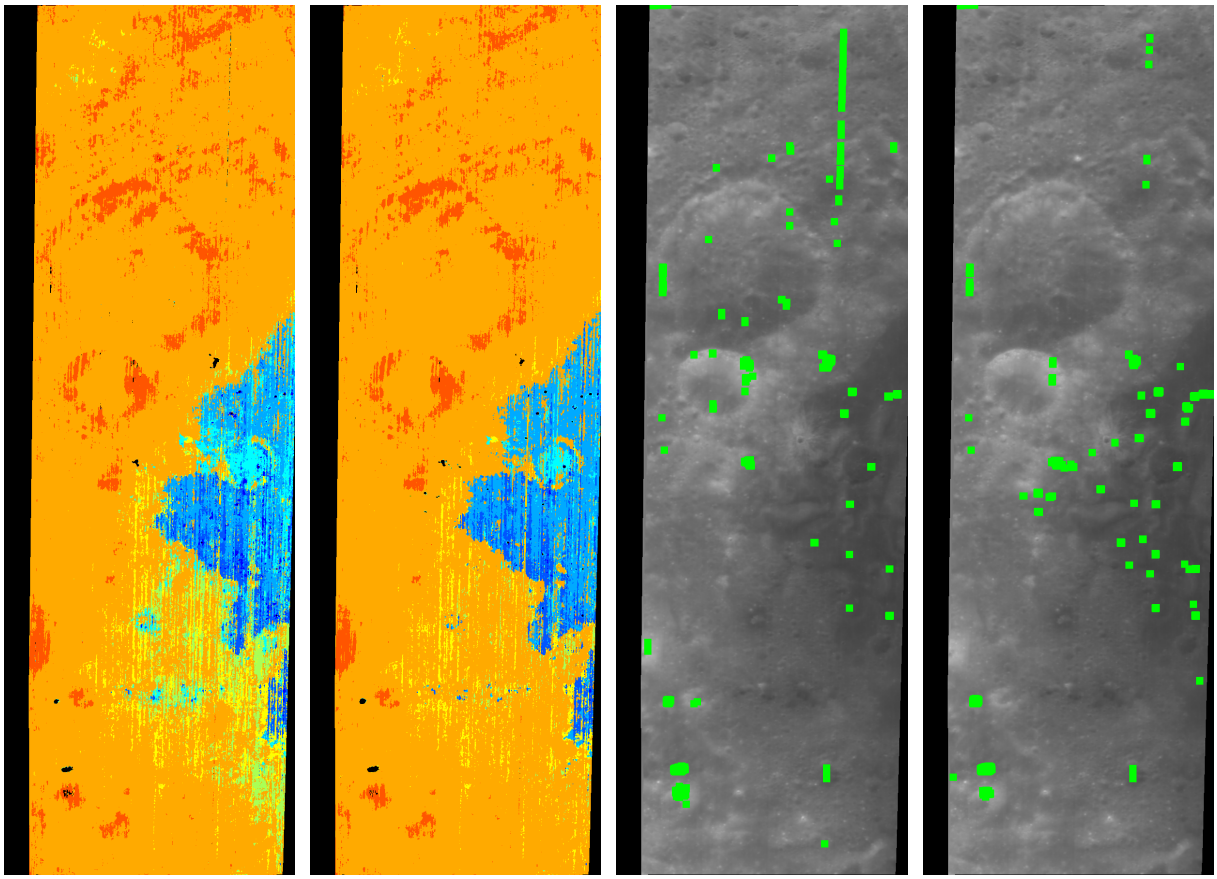
The detected novelties processed by an automatic clustering algorithm. The unsupervised mean-shift algorithm (Comaniciu and Meer, 2002) is applied to the detected novelties. Additionally, the clusters that have similar continuum removed spectrum (CR-spectra) are combined. This approach yields the results shown in Fig. 6.5(a). Notably, the order of the clusters obtained by the algorithm is not fixed and the colours have been selected to match the result of (Pieters et al., 2011), which is shown in Fig. 6.5(b). In addition to the uncommon spectra by Pieters et al. (2011) (Clusters B–D), the algorithm selects several locations on the floor of the basin (Cluster D). The CR-spectra of clusters A–D indicates olivine content, orthopyroxenes, spinel and pure olivine, respectively. The clusters E–I appear to be calibration artefacts that are still different from the training data, i.e. extremely high spectral reflectance and continuum slopes and/or missing or distorted absorption features. Furthermore, the number of spectra within the clusters A–I is



(a) $\theta_{\text{drift}} = 1.$ (b) $\theta_{\text{drift}} = 2.$ (c) $\theta_{\text{drift}} = 3.$ (d) $\theta_{\text{drift}} = 5.$

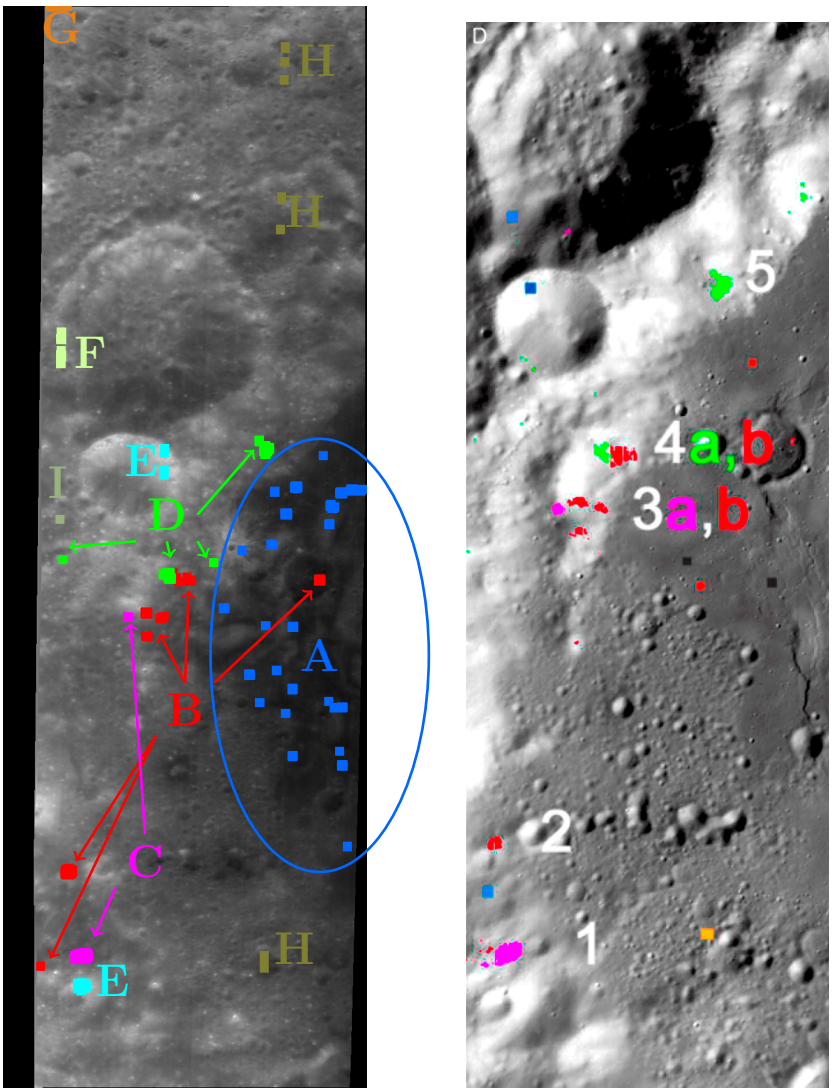
Figure 6.3: Influence of the novelty detection parameter θ_{drift}. (a)–(d) Detected uncommon spectra for various values of θ_{drift}. A low value of θ_{drift} detects too many novelties, while a high value of θ_{drift} makes the auto-encoder insensitive to uncommon spectra.

238, 234, 216, 180, 85, 63, 23, 13, and 1, respectively. It may thus be possible to remove rare calibration artefacts by removing small clusters.



(a) Initial labels. (b) Final labels. (c) Initial novelties (d) Final novelties

Figure 6.4: Influence of the semi-supervised learning. (a) The initial cluster labels mainly separate the region into highland and mare areas. There are, however, a few intermediate class labels in the southern half of the image. (d) The semi-supervised learning reduces the amount of detected intermediate classes and yields a more homogeneous highland and mare separation. The remaining stripe like class labels are artefacts of the remapping and data calibration process. (c) The initially detected novelties are mainly concentrated at bright crater rims. (d) The final novelties resulting from the semi-supervised learning increase mainly in the mare region and the orthopyroxene rich regions. The novelty maps have been dilated for readability, i.e. each novel pixel is indicated by highlighting a fifteen by fifteen pixels neighbourhood centred at the novel pixel. Refer to Fig. 6.3 for the original scale.



(a) Groups of uncommon spectra.

(b) Uncommon spectra mapped by Pieters et al. (2011).

Figure 6.5: The results of the automatic clustering algorithm. (a) The clusters of uncommon spectra that were determined by the automatic clustering algorithm. The novelty map has been dilated for readability, i.e. each novel pixel is indicated by highlighting a fifteen by fifteen pixels neighbourhood centred at the novel pixel. The detected regions detect the spinel, orthopyroxene and olivine bearing regions that were mapped by Pieters et al. (2011). (b) The original figure of Pieters et al. (2011). Red corresponds to orthopyroxene, green corresponds to olivine and magenta corresponds to spinel. Reproduced with kind permission of John Wiley & Sons, Inc. ©2011 American Geophysical Union

6.3 Semi-supervised Polynomial Classifier

The polynomial classifier (PC) is one of the most known types of interpolation functions (Theodoridis and Koutroumbas, 2009). According to Schürmann (1996), the matrix form of the discriminant function of the polynomial classifier is

$$\boldsymbol{H}_{\mathrm{P}} \cdot \boldsymbol{\beta}_{\mathrm{P}} = \boldsymbol{T}_{\mathrm{P}} \tag{6.28}$$

where $\boldsymbol{\beta}_{\mathrm{P}}$ and $\boldsymbol{T}_{\mathrm{P}}$ are the model parameters matrix and the target output of the classifier, respectively. $\boldsymbol{H}_{\mathrm{P}}$ is the vector of polynomial basis features which is computed from the input feature vectors $\vec{x}$ and depends on the polynomial degree n_{p} (see Schürmann (1996) for details). One possibility to obtain the solution is to minimise the residual norm of (Eq. (6.28)), i.e. $\|\boldsymbol{H}_{\mathrm{P}} \cdot \boldsymbol{\beta}_{\mathrm{P}} - \boldsymbol{T}_{\mathrm{P}}\|$. Since (Eq. (6.28)) is a linear system of equations for the output weights (similar to the equation of the extreme learning machine Huang et al. (2006b)), an estimation of the model parameters is possible using the pseudo inverse of the polynomial basis matrix according to $\boldsymbol{H}_{\mathrm{P}}{}^{\dagger} = (\boldsymbol{H}_{\mathrm{P}}{}^{T}\boldsymbol{H}_{\mathrm{P}})^{-1}\boldsymbol{H}_{\mathrm{P}}{}^{T}$ (Rao and Mitra, 1971):

$$\hat{\boldsymbol{\beta}}_{\mathrm{P}} = \boldsymbol{H}_{\mathrm{P}}{}^{\dagger} \cdot \boldsymbol{T}_{\mathrm{P}} \tag{6.29}$$

The pseudo-inverse $\boldsymbol{H}_{\mathrm{P}}{}^{\dagger}$ can be computed using the singular value decomposition (SVD) (Rao and Mitra, 1971). The calculated values of $\hat{\boldsymbol{\beta}}_{\mathrm{P}}$ and $\boldsymbol{H}_{\mathrm{P}}{}^{\dagger}$ are used in (6.28) to estimate the labels of the new samples.

The proposed algorithm is implemented to utilise two activities, which are incremental learning and novelty detection. Additionally, the novelty detection is achieved by using the EVT and the confidence band interval techniques.

6.3.1 Incremental learning phase

The incremental training of the polynomial classifier as described in Schürmann (1996) (which is similar to the analytical form of updating the extreme learning machine developed in Lan et al. (2009)) is adopted here, where

$$\boldsymbol{M}_{\mathrm{P}} = \boldsymbol{H}_{\mathrm{P}}{}^{T}\boldsymbol{H}_{\mathrm{P}} \text{ and } \boldsymbol{P}_{\mathrm{P}} = \boldsymbol{H}_{\mathrm{P}}{}^{T}\boldsymbol{T}_{\mathrm{P}}. \tag{6.30}$$

$$\text{and hence} \qquad \boldsymbol{\beta}_{\mathrm{P}} = \boldsymbol{M}_{\mathrm{P}}{}^{-1}\boldsymbol{P}_{\mathrm{P}}. \tag{6.31}$$

The dimension of matrix $\boldsymbol{M}_{\mathrm{P}}$ is $L \times L$ and the dimension of the $\boldsymbol{P}_{\mathrm{P}}$ is $L \times N_{\mathrm{class}}$, where L denotes the number of variables in the polynomial basis vector and N_{class} the number of classes. Assume that $N_{(0)}$ is the number of labelled samples for initial training. Then $\boldsymbol{M}_{\mathrm{P}(0)} = \boldsymbol{H}_{\mathrm{P}(0)}^{T}\boldsymbol{H}_{\mathrm{P}(0)}$ and $\boldsymbol{P}_{\mathrm{P}(0)} = \boldsymbol{H}_{\mathrm{P}(0)}^{T}\boldsymbol{T}_{\mathrm{P}(0)}$ are computed according to (Eq. (6.30)). It is thus $\boldsymbol{\beta}_{\mathrm{P}(0)} = \boldsymbol{M}_{\mathrm{P}(0)}^{-1}\boldsymbol{P}_{\mathrm{P}(0)}$. The polynomial classifier can be incremented using either

sample by sample or chunk by chunk. If the chunk $k+1$ has N' samples $\vec{x}'$ then the polynomial basis matrix $\boldsymbol{H}'_{\mathrm{P}}$ corresponding to this chunk of data is computed (Schürmann, 1996). Using the computed $\boldsymbol{H}'_{\mathrm{P}}$ in (Eq. (6.30)) yields $\boldsymbol{M}'_{\mathrm{P}} = {\boldsymbol{H}'_{\mathrm{P}}}^T \boldsymbol{H}'_{\mathrm{P}}$ and $\boldsymbol{P}'_{\mathrm{P}} = {\boldsymbol{H}'_{\mathrm{P}}}^T \boldsymbol{T}'_{\mathrm{P}}$ for the new data, leading to

$$\boldsymbol{M}_{\mathrm{P}(k+1)} = \boldsymbol{M}_{\mathrm{P}(k)} + \boldsymbol{M}'_{\mathrm{P}} \text{ and } \boldsymbol{P}_{\mathrm{P}(k+1)} = \boldsymbol{P}_{\mathrm{P}(k)} + \boldsymbol{P}'_{\mathrm{P}}, \tag{6.32}$$

and following (Eq. (6.31)) the incremented model becomes $\boldsymbol{\beta}_{\mathrm{P}(k+1)} = \boldsymbol{M}^{-1}_{\mathrm{P}(k+1)} \boldsymbol{P}_{\mathrm{P}(k+1)}$ (Schürmann, 1996).

6.3.2 Novelty detection phase

Novelty detection using EVT

In the training phase, the labels of the samples are supposed to be perfectly known, i.e. the probability that a sample belongs to another class than the labelled one is zero and the probability of belonging to the correct class is one. The probabilities of all classes are collected in the label vector. Hence, the ideal output vector of the polynomial classifier should follow the same values, i.e. all values are close to zero except the place of the class that the sample may belong to, which is around one. The trustiness of the classifier output decreases with an increasing deviation of the output values from the desired target output vector. Additionally, the absolute difference between the polynomial classifier output and the desired output is generated from a one-sided normal distribution with a mean of zero since the least squared estimate is bias-free. Moreover, we divide these values by the standard deviations of the residuals in the training phase, which corresponds to the square root of the mean squared residual. This leads to a one-sided normal distribution $\mathcal{N}(0,1)$ of zero mean and a variance of 1.

The predicted labels of the training data set $\hat{\boldsymbol{T}}_{\mathrm{P}}$ can be estimated by using (Eq. (6.28)) and substituting (Eq. (6.29))

$$\hat{\boldsymbol{T}}_{\mathrm{P}} = \boldsymbol{H}_{\mathrm{P}} ({\boldsymbol{H}_{\mathrm{P}}}^T \boldsymbol{H}_{\mathrm{P}})^{-1} {\boldsymbol{H}_{\mathrm{P}}}^T \boldsymbol{T}_{\mathrm{P}}. \tag{6.33}$$

The predicted value for each class is given by a column of $\hat{\boldsymbol{T}}_{\mathrm{P}}$, i.e. the number of columns in $\hat{\boldsymbol{T}}_{\mathrm{P}}$ corresponds to the number of classes. If $\vec{\hat{t}}_{\,c}$ denotes the vector of the predicted values and $\vec{t}_{\,c}$ the vector of the target values of class c, then the sum of the squared residuals is given by

$$r = \left[\vec{\hat{t}}_{\,Pc} - \vec{t}_{\,Pc}\right]^T \left[\vec{\hat{t}}_{\,Pc} - \vec{t}_{\,Pc}\right] \tag{6.34}$$

$$= \vec{t}_{\,Pc}^{\,T} \boldsymbol{H}_{\mathrm{P}} ({\boldsymbol{H}_{\mathrm{P}}}^T \boldsymbol{H}_{\mathrm{P}})^{-1} {\boldsymbol{H}_{\mathrm{P}}}^T \vec{t}_{\,Pc} + \vec{t}_{\,Pc}^{\,T} \vec{t}_{\,Pc}. \tag{6.35}$$

Noticeably, ${\boldsymbol{H}_{\mathrm{P}}}^T \boldsymbol{H}_{\mathrm{P}} = \boldsymbol{M}_{\mathrm{P}}$, and ${\boldsymbol{H}_{\mathrm{P}}}^T \vec{t}_{Pc}$ is the cth column of $\boldsymbol{P}_{\mathrm{P}}$. Therefore, the first summand on the right side of (Eq. (6.35)) can be incremented using (Eq. (6.32)). Consequently, the term $\vec{t}_{Pc}^{\,T} \vec{t}_{Pc}$ is the sum of the squared target values, and it can be incremented simply by adding the squared target values of the new samples. The standard deviation of these residual is the square root of r after dividing it by the number of samples. The result will be a class-wise standard deviation of the different models that is produced by the polynomial classifier.

As mentioned above, the absolute difference of the polynomial classifier output in the test phase and the target ("ideal") values originate from a zero mean one-sided normal distribution and will be of unit variance if it is divided by the class-wise standard deviation. Thus, it will fit the Gumbel distribution of the EVT (Clifton et al., 2008). Consequently, the highly believable samples should have $P_{\mathrm{ev}} = 0$ and the ideal novel sample should get $P_{\mathrm{ev}} = 1$. A threshold of $P_{\mathrm{th}} = 0.9$ is defined, and the sample is indicated as novel if at least one class detects a novelty.

Novelty Detection Using Confidence Band

Since the polynomial classifier output is a linear combination of the polynomial basis features, the confidence band intervals of the polynomial classifier output decision can be estimated. The confidence band intervals of the output decision can thus be estimated similar to the estimation in the ELM classifier. All the equations and the derivative is the same (as in cf. Section 6.1.2) except the value z is set to 180. Notably, the number of free parameters N_{p} corresponds to the number of polynomial bases features L.

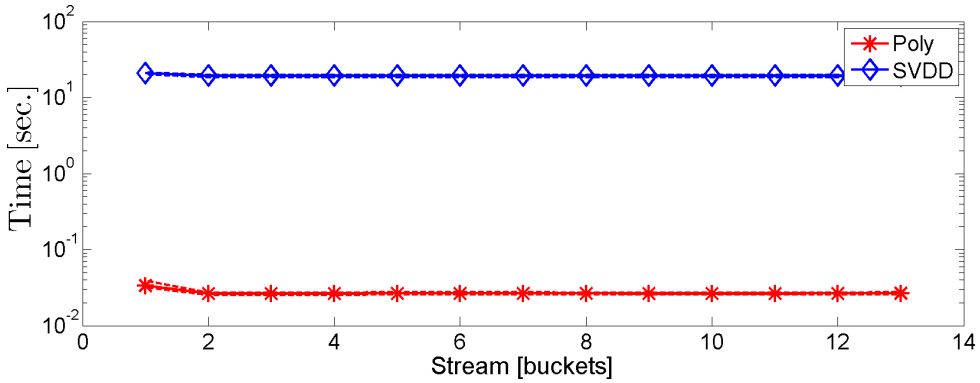
Finally, the sample $\vec{x}$ is considered as a novel only if both conditions, i.e. the EVT and confidence band, indicate it as a novel.

6.3.3 Believability Conditions

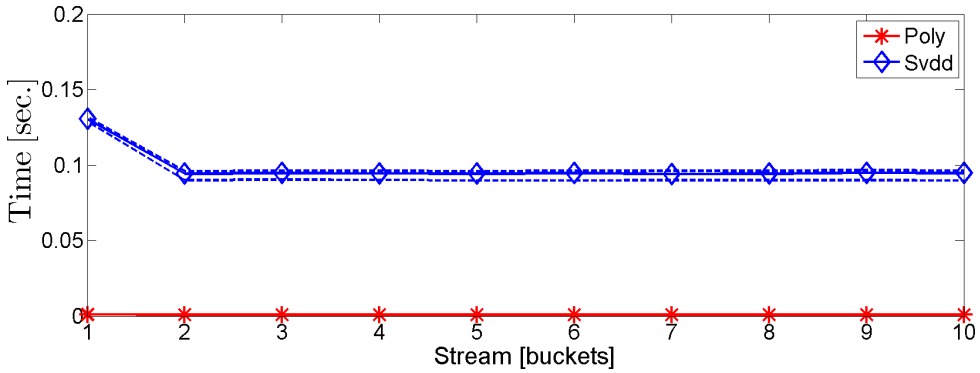
Additional conditions should be fulfilled to use the sample in the next training set. The label is considered untrustworthy if any class detects the novelty. Moreover, the ideal difference between the largest and the second largest output is one. Hence, a second threshold is set to this difference, and the believability flag is set if this difference is greater than 0.5. All untrustworthy samples are excluded from the next training set.

6.3.4 Experiments and Results

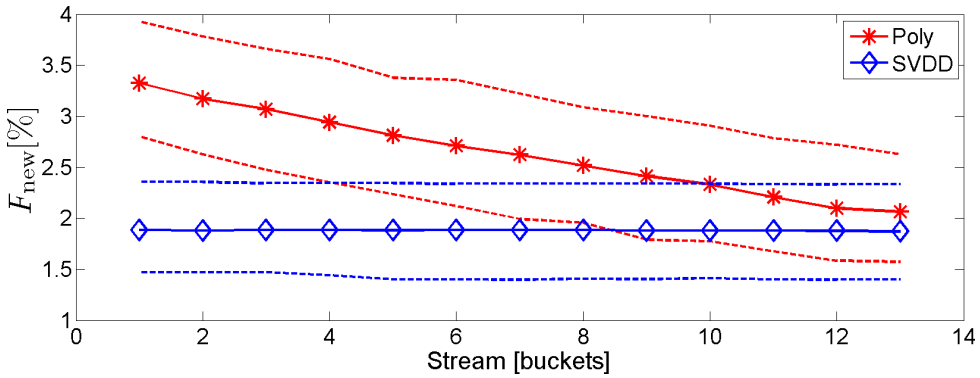
Since the SVDD algorithm (cf. Section A.1) is well-known within novelty detection and incremental learning, it is used here for comparison with the results of the proposed polynomial algorithm. We used the incremental version of the SVDD. Both classifiers have the ability to detect outliers and are updated incrementally. Hence, we used them in the semi-supervised process to compare their results. The experiment set-up in (Section 5.2)



(a) T_{update}: Gesture data

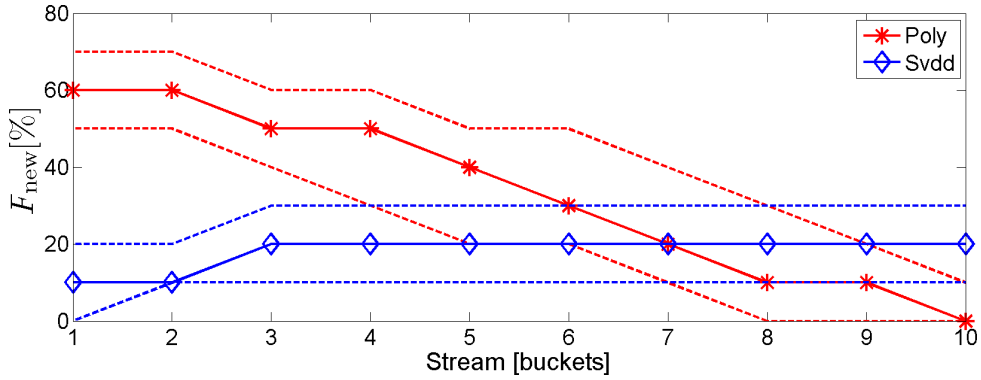


(b) T_{update}: Iris data

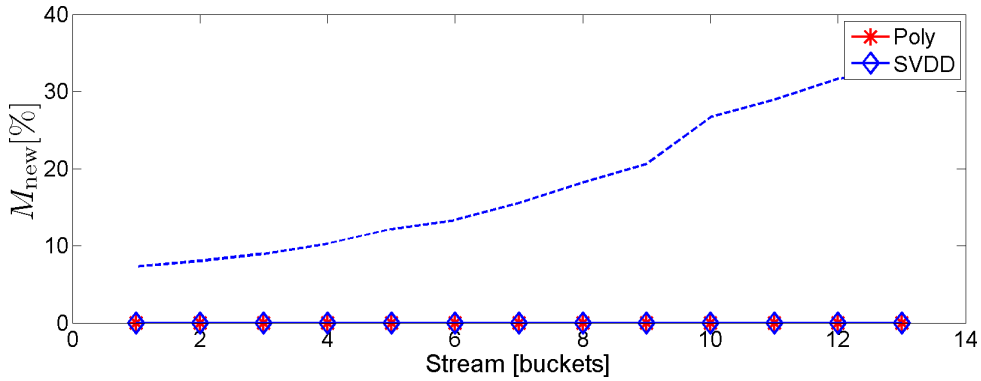


(c) F_{new}: Gesture data

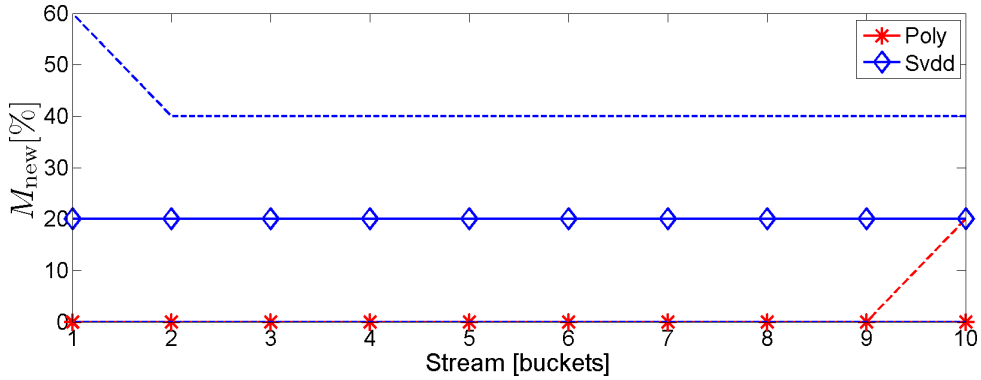
Figure 6.6: Accuracy metrics of the classifiers: (a)–(b) Time consumed by each classifier for the gesture and the Iris data set, respectively. (e)–(f) The rate of missed novelties M_{new} for the gesture and the Iris data set, respectively. (c)–(d) The rate of false detections F_{new} for the gesture and the Iris data set, respectively. (g)–(h) The total error E_{total} for the gesture and the Iris data set, respectively.



(d) F_{new}: Iris data

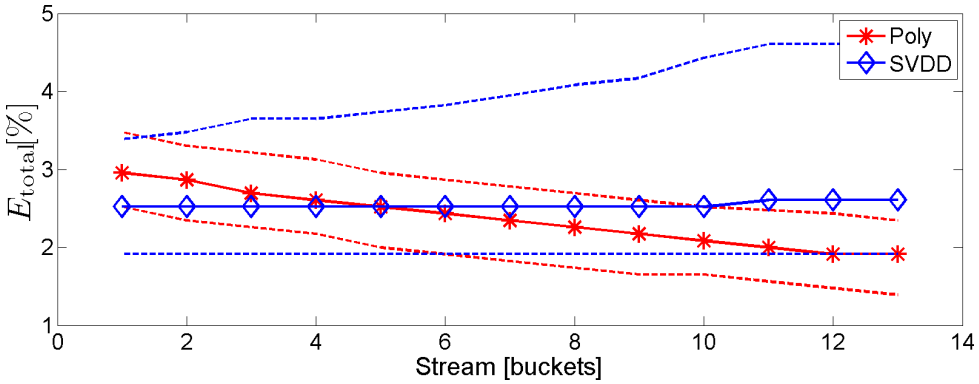


(e) M_{new}: Gesture data

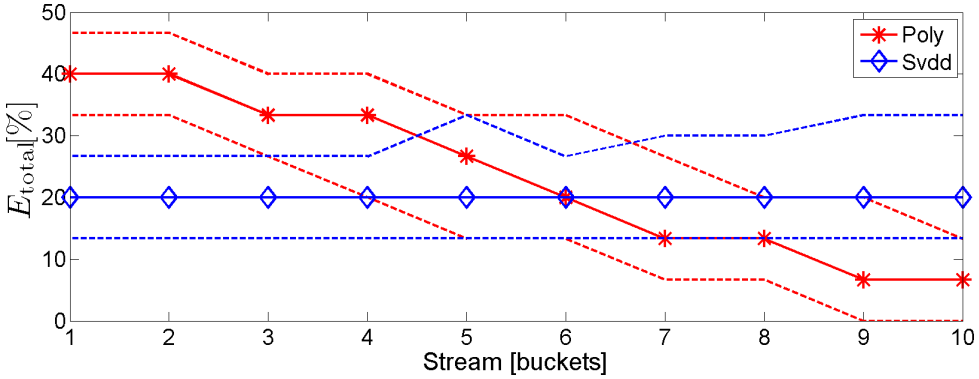


(f) M_{new}: Iris data

Figure 6.6: (Continued)



(g) E_{total}: Gesture data



(h) E_{total}: Iris data

Figure 6.6: (Continued)

is used to evaluate the proposed algorithm for two data sets; Gesture and Iris datasets. The gesture data set is divided into fractions of 25%, 50%, and 25% as an initial training set, learning set and test set, respectively. Whereas, the Iris data set (150 samples) is divided into fractions of 40%, 30%, and 30%, respectively. To emulate the data streaming, the learning data set is subdivided into "buckets", each of which consists of 100 samples in case of the gesture data set and 5 samples in the case of the Iris data set. The novelty flag is set if both conditions of novelty, i.e. the EVT output and the confidence band condition, indicate a sample as novel, whereas the believability flag is set if it is not indicated as novel by any of the novelty conditions and the difference between the winner class and the second class exceeds 0.5.

A very important factor is the incremental updating, which it is evaluated by computing the processing time (total time required for training and testing) of each classifier. The runtime of the proposed polynomial classifier is computed using the explained experiments and compared with the runtime of the SVDD classifier. The difference between

the processing times is quite clearly apparent from Fig. 6.6(a) and Fig. 6.6(b) (the dashed lines denote the 25% and 75% quantiles in all figures). It requires just some milliseconds to update itself for one bucket, and thus it is suitable for incremental learning. To implement the autonomous updating successfully, the classifier should be able to indicate the samples which are dissimilar to the concept learned by the classification system. The missed indication of these samples is summarised by the value M_{new}, which means that it should be as small as possible to reject all outliers from the training set. It is exactly equal to zero for the proposed algorithm which means that all outliers in both datasets are detected. Although the median of M_{new} of the SVDD classifier is zero, the 75% quantile exceeds 30% of the total number of outliers in the case of the gesture data as shown in the Fig. 6.6(e) and it corresponds to 20% with a 75% quantile of more than 40% for the Iris data (Fig. 6.6(f)).

In contrast, rejecting a large number of samples leads to a small amount of new data, which means that not much new information will be added to the classifiers. The error of rejecting samples belonging to the known concept is evaluated by computing F_{new}. It affects only the size of selected data for the next training period, i.e. it is not critical if some samples belonging to known classes are rejected. This behaviour may occur because such samples lie on the border of the class. Fig. 6.6(c) and Fig. 6.6(d) show the value of F_{new} of the proposed algorithm, which starts higher than that of the SVDD but decreases with successive updating and becomes lower than that of the SVDD. This means that additional information is added to the classifier by the semi-supervised learning. The other important measure in semi-supervised learning, the total error rate E_{total}, which contains the above two types of error in addition to the error of misclassifying a sample of a known class as belonging to the wrong known class. It should decrease with increasing amount of training data, which is shown in Fig. 6.6(g) and Fig. 6.6(h) for the proposed algorithm. This means that the classifier extracts new information from the new unlabelled data. In contrast, the total error rate of the SVDD classifier slightly increases with increasing amount of data, which may be due to the continued acceptance of some outliers.

7

Contribution: Semi-Supervised Methods Based on Metric Learning

Metric learning, sometimes called similarity learning, predicts the class of different objects depending on a measured distance between the object and another object, e.g. the Euclidean distance or the Mahalanobis distance. Similar objects are assumed to be closer to each other, and thus the same class label may be assigned to close-by objects. Commonly, a class is represented by its centroid, i.e. the expected or average instance. In this context, adaptive distance metrics are e.g. discussed by Schneider et al. (2009). It is straightforward to add new classes to the trained model by adding the centroids of the new classes to the set of class centroids. These classifiers are termed "nearest class mean" classifiers (NCM) (e.g. Kuncheva and Bezdek, 1998; Webb, 2003; Zhou et al., 2008) and mostly use the Mahalanobis distance to measure the distance between the instance to the mean of the classes. These classifiers, however, need all the training data to retrain the classifier. Additionally, they need additional labelled data for threshold setting. The k-nearest neighbour classifier k-NN (e.g. Altman, 1992; Deng et al., 2009; Weston et al., 2011) is a non-parametric classifier, has good performance in non-linear distributions but the nearest neighbour search is computationally expensive for high-dimensional and/or large databases.

To overcome the computational expense of high-dimensional data, possibly linear subspace projections have been proposed. The well-known Fisher linear discriminant analysis (FDA) (e.g. Fisher, 1936; Rao, 1948; Webb, 2003; Baudat and Anouar, 2000; Mika et al., 1999) determines a projection matrix that maximises the ratio of the between-class scatter and the within-class scatter. The FDA, however, reduces to a principal component analysis on the class means if the covariance of each class is the identity matrix (Mensink et al., 2013).

Mensink et al. (2013) propose an alternative subspace projection by sharing one low-rank Mahalanobis distance across all classes. Their nearest class mean of multi-class

logistic discrimination (NCMML) reduces the dimensionality of the data using a linear subspace projection and gives the ability to adapt to new classes. Unfortunately, its performance in the non-linear distributions of data is lack and also its accuracy proportionally decreases as the dimensionality of the data reduces. A comparison between NCMML and FDA is presented by Mensink et al. (2013).

In this chapter, we have used the class-specific Mahalanobis distance to compute the distance between a test sample or unlabelled sample and each class, similar to the NCM framework. In contrast to the NCM, the required parameters to compute the distance are updated incrementally. Additionally, the EVT is used instead of conventional novelty detection.

The NCMML classifier proposed by Mensink et al. (2013) uses a linear subspace projection. Most gesture datasets, however, are highly non-linear. To extend the NCMML towards non-linear datasets, we apply the kernel trick to improve the performance of the NCMML in such environments and make it less sensitive to the dimensionality of the data. The kernel function is applied after the linear subspace projection to avoid an increased runtime. The proposed algorithm shows an increased accuracy in both linear and non-linear system as well as a reduced time of processing in most cases.

This chapter has been adapted and/or adopted from: (Al-Behadili et al., 2015b,c,d, 2016e)

7.1 Incremental Update of Mahalanobis Distance Parameters

The Mahalanobis distance may be interpreted as the Euclidean distance in a space with translated, scaled and/or rotated coordinate axes (Marsland, 2009). Consequently, all equidistant points reside on a hyper-ellipsoid. The Mahalanobis distance to one class is defined by the centre of the hyper-ellipsoid, i.e. the mean instance of a class, and a rotation and scaling, i.e. the covariance matrix of a class. The Mahalanobis distance is defined by its centre, i.e. the d-dimensional class mean $\vec{\mu}$, and a set of scaled and rotated coordinate axes, i.e. $d \times d$ dimensional covariance matrix $\boldsymbol{\Sigma}$. To discard the training data and thus reduce the memory requirements, we apply an incremental update of the parameters that does not require the previous training data. Let $\vec{\mu}$, $\boldsymbol{\Sigma}$, $\vec{\mu}'$, $\boldsymbol{\Sigma}'$, and $\hat{\vec{\mu}}$, $\hat{\boldsymbol{\Sigma}}$ be the mean vector and the covariance matrices estimated from the initial data, the added data and the combined data, respectively. The previous set of parameters, i.e. $\vec{\mu}$, $\boldsymbol{\Sigma}$, was estimated from N samples and N' denotes the number of samples that is added to the training dataset resulting in a dataset size of $N_{\text{total}} = N + N'$. The updated mean vector is then given by

$$\vec{\hat{\mu}} = \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} \vec{x}_i = \frac{1}{N_{\text{total}}} \left[\sum_{i=1}^{N} \vec{x}_i + \sum_{i=N+1}^{N_{\text{total}}} \vec{x}_i \right] = \frac{1}{N_{\text{total}}} (N \cdot \vec{\mu} + N' \cdot \vec{\mu}') \tag{7.1}$$

as shown by Schürmann (1996), where $\vec{x}_i$ denotes the ith sample vector in the ordered combined dataset, i.e. the last N' samples are the added samples. For the covariance matrix, one obtains

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N_{\text{total}}} \left[\sum_{i=1}^{N} (\vec{x}_i - \vec{\hat{\mu}})(\vec{x}_i - \vec{\hat{\mu}})^T + \sum_{i=N+1}^{N_{\text{total}}} (\vec{x}_i - \vec{\hat{\mu}})(\vec{x}_i - \vec{\hat{\mu}})^T \right]. \tag{7.2}$$

Some simplifications yield

$$\hat{\boldsymbol{\Sigma}} = \frac{N}{N_{\text{total}}} \boldsymbol{\Sigma} + \frac{N'}{N_{\text{total}}} \boldsymbol{\Sigma}' + \frac{NN'}{{N_{\text{total}}}^2} (\vec{\mu} - \vec{\mu}')(\vec{\mu} - \vec{\mu}')^T. \tag{7.3}$$

When one single sample is added at a time, Eq. (7.3) further reduces to

$$\hat{\boldsymbol{\Sigma}} = \frac{N}{N+1} \boldsymbol{\Sigma} + \frac{N}{(N+1)^2} (\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T. \tag{7.4}$$

In case of using the factor $1/(N-1)$, which it is used in this thesis, instead of $1/N$ to compute the covariance, Eq. (7.3) is changed to

$$\hat{\boldsymbol{\Sigma}} = \frac{N-1}{N_{\text{total}}-1} \boldsymbol{\Sigma} + \frac{N'-1}{N_{\text{total}}-1} \boldsymbol{\Sigma}' + \frac{NN'}{(N_{\text{total}}-1)(N_{\text{total}})} (\vec{\mu} - \vec{\mu}')(\vec{\mu} - \vec{\mu}')^T. \tag{7.5}$$

Based on Eq. (7.1) and Eq. (7.4), the mean vector and covariance matrix are updated each time a new sample is added to the training data. The scheme of Eq. (7.2)–Eq. (7.4) is related to but different from Schürmann (1996), where a direct iterative estimation of the inverse covariance matrix is performed. The mean vector, the covariance matrix and the number of samples used for the estimation are stored, and the old training data may be discarded.

7.1.1 Novelty detection using Mahalanobis distance

Commonly, metric learning is used for conventional novelty detection (e.g. Pedregosa et al., 2011). Most of the existing methods rely on the validation set to set a threshold, i.e. a bounding hyper-ellipsoid specified by a maximum Mahalanobis distance, for each class, respectively. Instances outside this Mahalanobis radius are supposed to be outliers for this class while instances inside this radius may belong to that specific class. In a multi-class system, the instance will be assigned to the class with the smallest Mahalanobis distance and novelty is detected if the instance resides outside the bounding hyper-ellipsoid of all classes.

Here, we present an incremental classifier that uses the extreme value theory and the Mahalanobis distance (IncEVT). It can either assign a new instance to one of the existing classes or indicate it as an unknown. The results show that it works efficiently even in non-linear multi-class systems. The presented work differs from existing approaches as follows. First, it is a single model classifier that is incremented by simply updating the mean and the covariance of the seen classes without requiring complex computations or operations or the full dataset. Consequently, extra hardware resources or time of processing is not required. Secondly, there is one constant threshold for the whole system due to the EVT. Thus, the validation dataset is not required contrary to the conventional methods. Finally, it is possible to work efficiently in the non-linear multi-class systems without the application of kernels. Additionally, dividing the non-linearly separable classes to several sub-classes may further improve the accuracy of the classification.

The sample is assumed to be novel if all the outputs of the EVT are greater than the threshold ($P_{\text{th}} = 0.995$). The believability flag is set if the output of the EVT that correspond to the estimated label is equal or less than the confidence threshold (0.5).

7.1.2 Experiments and results

The accuracy of the algorithm is evaluated based on three datasets; Gesture dataset (Section 4.1), Artificial dataset (Section 4.2) and Iris dataset (Section 4.3). The well-known but rather a small Iris dataset is used to evaluate the accuracy in the non-linear Euclidean space, a gesture dataset is used to evaluate the ability to detect novel classes in a possible application scenario, and a large normally distributed artificial dataset will demonstrate the runtime of the algorithm.

The proposed algorithm has the ability of detecting the novelties in linear and non-linear multi-class problems without the application of kernel functions. Moreover, it is an incremental algorithm, i.e. it updates itself within a small amount of time. To show these properties, we conduct three experiments using the mentioned datasets.

In each experiment, we compare the result of the IncEVT with the result of a classifier that uses threshold based novelty detection (normal Mahalanobis distance based classifier). In the case of the IncEVT, only three sets of data are required: an initial training

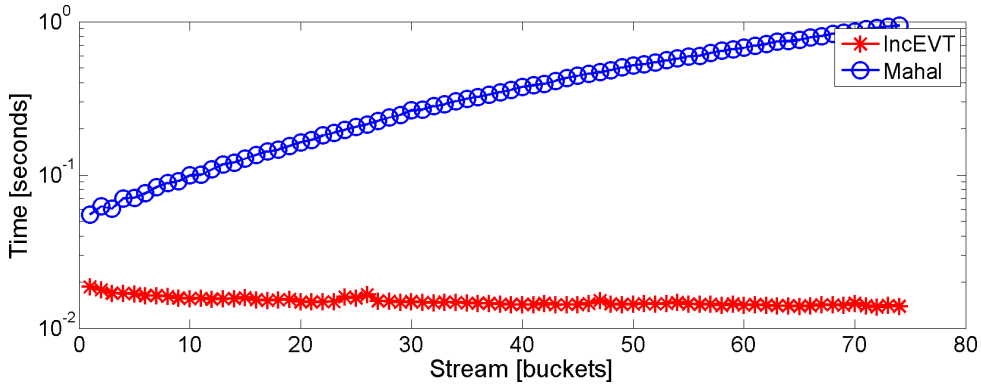


Figure 7.1: A comparison between the classifiers regarding the runtime using artificial data. The 50% quantile over the 100 runs is used to draw the approximated runtime.

set, a learning set and a test set. In the case of the threshold-based classifier, however, one additional set for the threshold validation is required. Hence, we divide the original data to initial training, learning, test, and validation sets, and we use just the first three for IncEVT. The learning set is partitioned into equally sized buckets, and one bucket per iteration is classified before the classifiers are updated with these new data.

The runtime and the accuracy are evaluated in different experiments. As previously explained each experiment is repeated 100 times with different random permutations of the dataset to compute the expectation of the result since it depends on the random partition of the data. Both classifiers use the same random permutation for each experiment during these 100 experiments.

Comparing runtime experiment

The runtime of the classifiers is evaluated based on the artificial dataset. Here, we assume that no novelties or unknown classes appear. The test set comprises 10% of the data. The remaining data is partitioned into a training set (10%) and learning set (80%). The learning set is divided into buckets using a bucket size of 100 samples. One iteration consists of training and a classification phase. The total time T_{update} consumed by one iteration is recorded for each classifier, respectively.

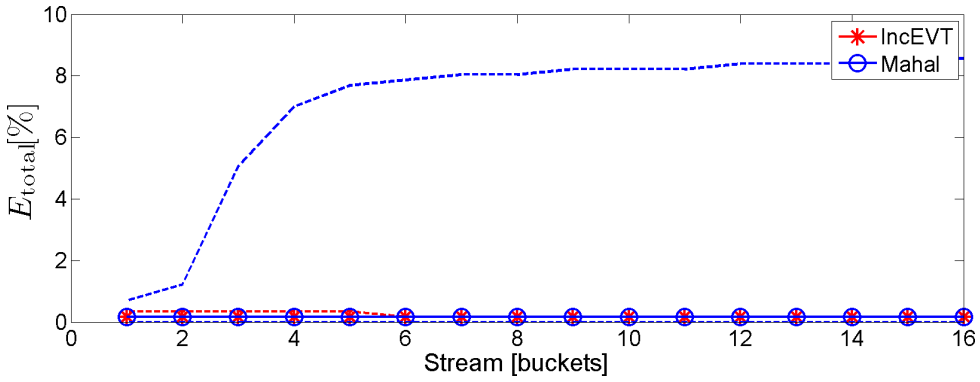
Fig. 7.1 shows the time T_{update} consumed by the classifiers for one bucket of the artificial dataset, i.e. adapting to the previous bucket and classifying the new bucket. Initially, the consumed time is nearly identical, while the difference increases with each new bucket. Notably, IncEVT consumes an almost constant amount of time while the computational burden of the non-incremental Mahalanobis distance classifier is strongly increased with the increasing the amount of data in the training set.

Novelty detection evaluation

These experiments address the accuracy and the ability to detect novelties. The experiments are conducted using the gesture dataset and the Iris dataset, respectively. The datasets are partitioned into a training set, a learning set, a testing set and a validation set. For the gesture dataset, the fractions are 30%, 30%, 20%, 20%, respectively and for the Iris dataset the fractions are 50%, 25%, 10%, 15%, respectively. Again, the learning set is divided into buckets of size 100 or 10 samples for the gesture dataset and the Iris dataset, respectively. The validation set is used only by the threshold-based classifier to determine the threshold and is ignored by the IncEVT. The novelty threshold of IncEVT set to 0.995 and the believability threshold set to 0.5.

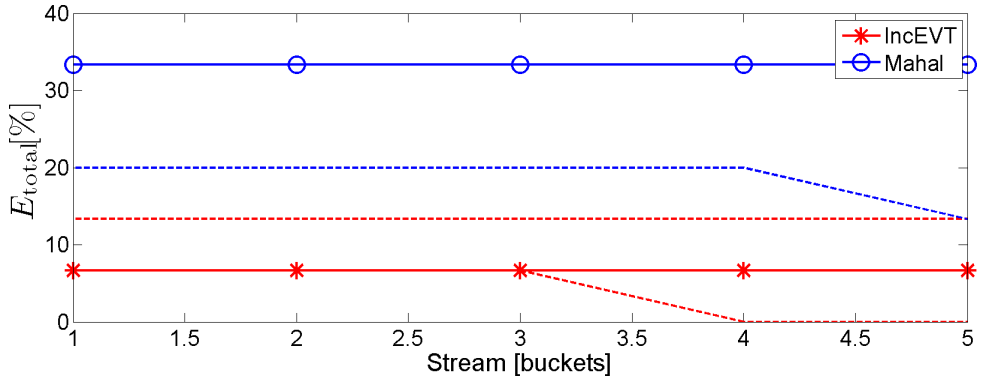
After being adapted to the initial training set, the classifiers assign labels to the samples in the first bucket of the learning set. Notably, the algorithms may either assign the sample to an existing class or detect a novelty. The trusty assigned labels are added to the training set of the classifier, and the classifiers are then retrained/incremented. This procedure is repeated for each bucket, respectively. In this experiment, each classifier may be updated with a different number of samples depending on the classifier output.

The total error E_{total} for the gesture and the Iris datasets are shown in Fig. 7.2(a) and Fig. 7.2(b), respectively. The threshold-based classifier indicates a value of 33% for the Iris dataset. This is due to the misclassification of all data in the linearly non-separable class, which might be related to the problem of estimating the correct threshold. In contrast, the total error of IncEVT is only 6.8%. F_{new} is shown in Fig. 7.2(c) and Fig. 7.2(d) for

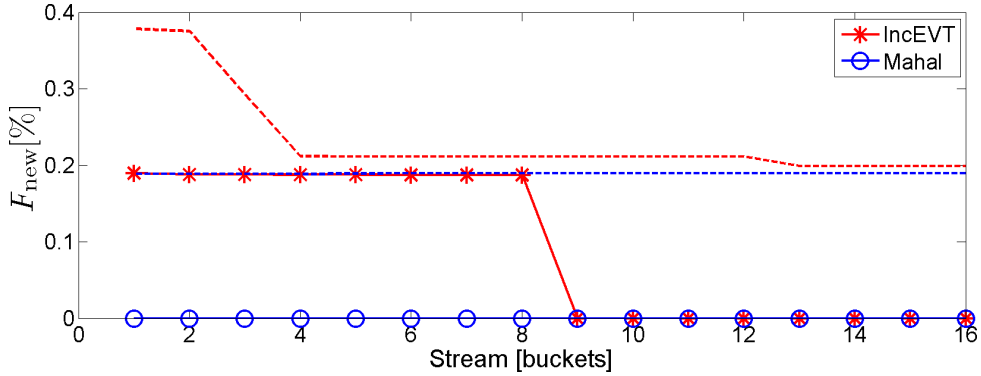


(a) E_{total}: Gesture data

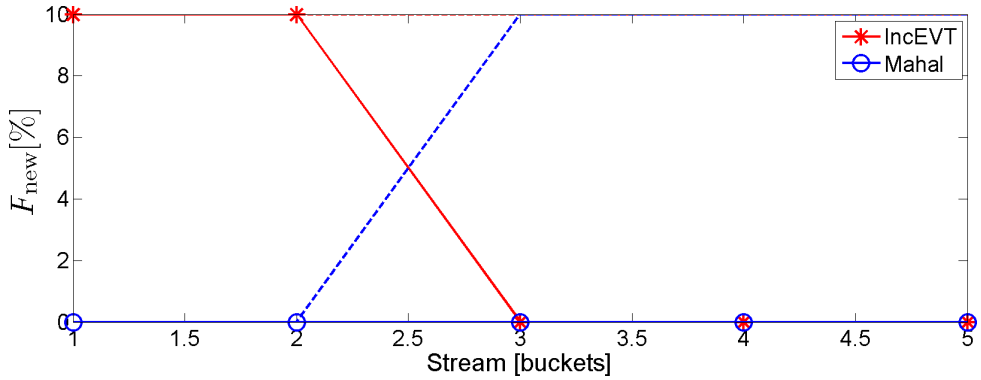
Figure 7.2: Accuracy metrics of the classifiers: (a)–(b) The total error E_{total} for the gesture and the Iris dataset, respectively. The dashed lines correspond to the 25% and 75% quantiles and represent the spread over 100 repetitions. (c)–Fig. 7.2(d) The rate if false detections F_{new} for the gesture and the Iris dataset, respectively. (e)–(f) The rate of missed novelties M_{new} for the gesture and the Iris dataset, respectively.



(b) E_{total}: Iris data

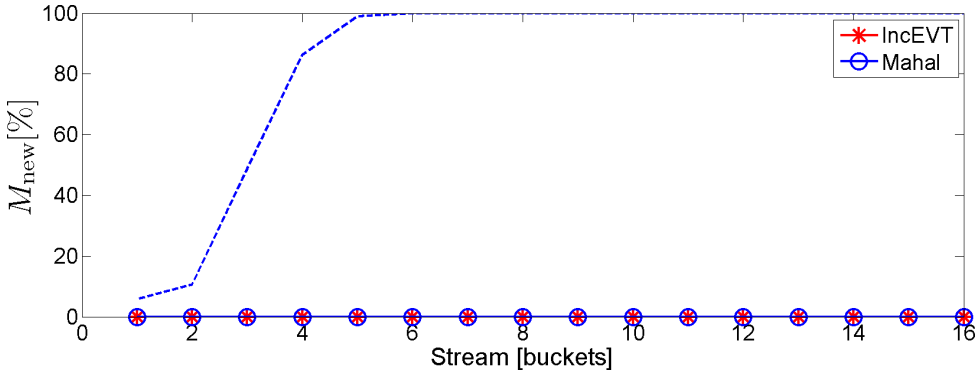


(c) F_{new}: Gesture data

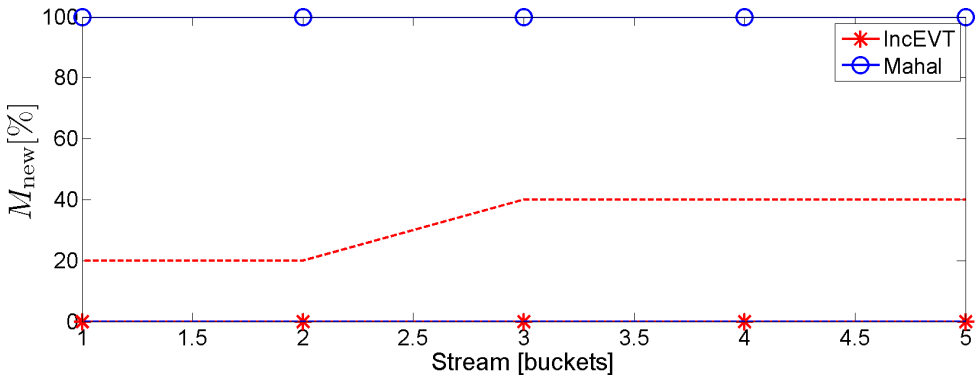


(d) F_{new}: Iris data

Figure 7.2: (Continued)



(e) M_{new}: Gesture data



(f) M_{new}: Iris data

Figure 7.2: (Continued)

the gesture and the Iris dataset, respectively. These figures show that the IncEVT yields a smaller number of false positives if the number of training samples is sufficient. This may be due to the Gumbel distribution being inadequate for small training sets (Clifton et al., 2011). Fig. 7.2(e) and Fig. 7.2(f) show M_{new} for the gesture and the Iris dataset, respectively. For the Iris data, the IncEVT detects all novelties, and the threshold-based classifier misses all novelties. Again, this behaviour may be related to the data based threshold estimation.

7.2 Semi-supervised Kernel Nearest Class Mean

The performance of the classifier in semi-supervised learning scenarios is very sensitive to the classifier accuracy in the previous classification process. Hence, we need to build a classifier that is robust against outliers, has high accuracy, and is quickly retrained in near constant time. The NCMML classifier proposed by Mensink et al. (2013) uses a linear

subspace projection approximating the covariance matrix to reduce the computational complexity of modern high-dimensional classification problems. Most gesture datasets, however, are not linearly separable. Mensink et al. (2013) divided the non-linearly separable classes to several sub-classes by applying a clustering algorithm such as K-means. This method is called nearest class mean with multiple class centroid (NCMC). They proved that the classification results are improved by using the NCMC. Unfortunately, the results of applying the NCMC to gesture database need further improvements. Hence, We extend the NCMC towards non-linearly inseparable datasets by applying the kernel trick to improve the performance of the NCMC in such environments and make it less sensitive to the dimensionality of the data. To avoid an increased runtime, the kernel function is applied after the linear subspace projection. The proposed algorithm shows an increased accuracy in both linear and non-linear system as well as a reduced time of processing in most cases. This method is quite helpful for large dataset classification but unfortunately; it is not incremental due to the non-linear optimisation of computing the projection matrix.

7.2.1 Non-linear NCM with multiple class centroid (NCMC)

The nearest class mean classifier implemented by Mensink et al. (2013) looks for the closest centroid $\vec{\mu}_c$ of class c to assign the corresponding class label to the instance with feature vector $\vec{x}$ of dimensionality d. If we have a new sample $\vec{x}'$, the distance between this sample and the centroid $\vec{\mu}_c$ of class c is $d_{\text{Eucl}}(\vec{x}', \vec{\mu}_c)$. This new sample is labelled by $\hat{c}$, corresponding to the class with the minimum distance among the distances from N_{class} classes to this sample:

$$\hat{c} = \underset{c \in \{1,\ldots,N_{\text{class}}\}}{\operatorname{argmin}} d_{\text{Eucl}}(\vec{x}', \vec{\mu}_c). \tag{7.6}$$

The centroid of class c is the mean of the N_{c} instances $\vec{x}_i$ of class c:

$$\vec{\mu}_c = \frac{1}{N_{\text{c}}} \sum_{i=1}^{N_{\text{c}}} \vec{x}_i. \tag{7.7}$$

The squared Mahalanobis distance specified by the covariance matrix $\boldsymbol{\Sigma}$, i.e.

$$d_{\text{Mahal}}(\vec{x}, \vec{\mu}_c) = (\vec{x} - \vec{\mu}_c)^T \boldsymbol{\Sigma}^{-1} (\vec{x} - \vec{\mu}_c), \tag{7.8}$$

was applied by Mensink et al. (2013). Furthermore, it is assumed by Mensink et al. (2013) that $\boldsymbol{\Sigma} = \boldsymbol{W}^T \boldsymbol{W}$, since $\boldsymbol{\Sigma}$ is a positive semi-definite matrix. The matrix $\boldsymbol{W} \in \mathbb{R}^{d' \times d}$ is a low-rank metric and $d' \leq d$ is the effective dimension of the subspace projection. Consequently,

$$d_{\mathrm{W}}(\vec{x}, \vec{\mu}_c) = (\vec{x} - \vec{\mu}_c)^T \boldsymbol{W}^T \boldsymbol{W} (\vec{x} - \vec{\mu}_c) = \|\boldsymbol{W}\vec{x} - \boldsymbol{W}\vec{\mu}_c\|_2^2 . \tag{7.9}$$

The posterior of the class c given an instance $\vec{x}$ is defined by Mensink et al. (2013) as

$$p(c|\vec{x}) = \frac{\exp\left(-\frac{1}{2} d_{\mathrm{W}}(\vec{x}, \vec{\mu}_c)\right)}{\sum_{\tilde{c}=1}^{N_{\mathrm{c}}} \exp\left(-\frac{1}{2} d_{\mathrm{W}}(\vec{x}, \vec{\mu}_{\tilde{c}})\right)} \tag{7.10}$$

assuming uniformly distributed classes and a normally distributed likelihood, i.e. $p(\vec{x}|c) = \mathcal{N}(\vec{\mu}_c, \boldsymbol{W}^T\boldsymbol{W})$. The covariance $\boldsymbol{W}^T\boldsymbol{W}$ is shared across all classes. The log-posterior of the correct prediction is maximized using a stochastic gradient descent to obtain the optimal projection matrix $\boldsymbol{W}$ (Mensink et al., 2013).

For non-linear spaces, Mensink et al. (2013) replace the single centroid of each class by multiple prototypes, which are obtained using the k-means algorithm. The query samples are then assigned to the class of the nearest centroid. Assuming an N_{cent} mixture of normal distributions centred at the centroids $\vec{\mu}_{c_j}$, $j \in \{1, \ldots, N_{\mathrm{cent}}\}$ for each class c, the posterior probability of class c is defined by Mensink et al. (2013) as

$$p(c|\vec{x}) = \sum_{j=1}^{N_{\mathrm{cent}}} p(\vec{\mu}_{c_j}|\vec{x}), \text{ with} \tag{7.11}$$

$$p(\vec{\mu}_{c_j}|\vec{x}) = \frac{\exp\left(-\frac{1}{2} d_{\mathrm{W}}(\vec{x}, \vec{\mu}_{c_j})\right)}{\sum_{c=1}^{N_{\mathrm{class}}} \sum_{\tilde{j}=1}^{N_{\mathrm{cent}}} \exp\left(-\frac{1}{2} d_{\mathrm{W}}(\vec{x}, \vec{\mu}_{c_{\tilde{j}}})\right)}. \tag{7.12}$$

7.2.2 Kernel based metrics

Kernels are proposed to solve non-linear separation problems in different types of machine learning algorithms. As described in detail by Theodoridis and Koutroumbas (2009), the transformation $\vec{\varphi} : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^{d'}$ is a generally non-linear transformation of the feature space $\mathbb{R}^d$ to a space $\mathbb{R}^{d'}$ of increased dimension d'. Due to the transformation $\vec{\varphi}$, a linear separation, i.e. a separating hyperplane, in $\mathbb{R}^{d'}$ becomes a non-linear separating function when projected back onto the original feature space. Consequently, the classes may become linearly separable after the transformation into the space of increased dimension. An example is shown in Fig. 7.3.

Commonly, it is sufficient to compute inner products in the higher dimensional space, e.g. computing the Euclidean distance of samples from the separating hyperplane. The so-called "kernel trick" utilises this fact. As shown by Theodoridis and Koutroumbas (2009),

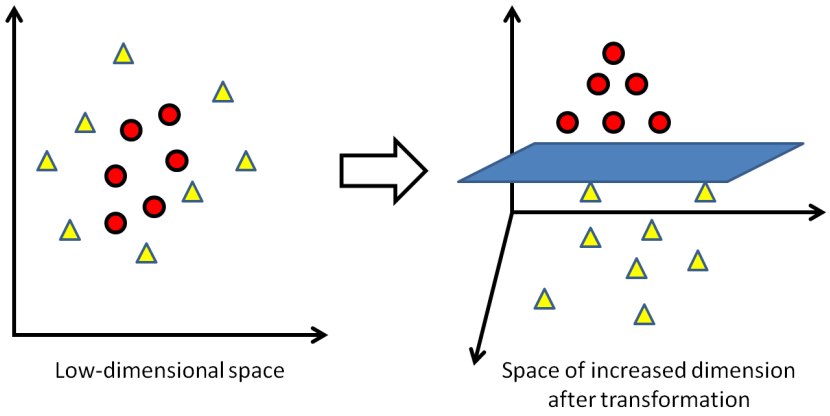


Figure 7.3: Kernel trick. By transforming the original space (left) into a space of increased dimension (right) the two classes "circle" and "square" become linearly separable. Adapted from Elmezain et al. (2009)

.

if the mapping of the vectors $\vec{x}_i$ and $\vec{x}_j$ satisfies Mercer's theorem it is not required to evaluate or know the mapping $\vec{\varphi}$, and the inner product may be replaced by the kernel function

$$K(\vec{x}_i, \vec{x}_j) = \langle \vec{\varphi}(\vec{x}_i), \vec{\varphi}(\vec{x}_j) \rangle_{d'} = \vec{\varphi}(\vec{x}_i)^T \vec{\varphi}(\vec{x}_j). \tag{7.13}$$

Consequently, it is possible to evaluate the inner product in the high dimensional space by applying a possibly non-linear kernel function $K(\vec{x}_i, \vec{x}_j)$ to the representations $\vec{x}_i$ and $\vec{x}_j$ in the original space. Examples of common kernel functions are polynomial kernels or Gaussian kernels (Theodoridis and Koutroumbas, 2009). In this work, the kernel function is a radial basis function (RBF) of Gaussian shape given by

$$K_{\mathrm{RBF}}(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|_2^2}{2{\sigma_{\mathrm{rbf}}}^2}\right), \tag{7.14}$$

where σ_{rbf} denotes the predefined width of the kernel function. The Gaussian RBF kernel is shift-invariant. Consequently, the similarity metric learned using RBF kernels will be coordinate-independent (Kung, 2014).

7.2.3 Proposed kernel NCM (KNCM)

Recalling the distance proposed by Mensink et al. (2013) $d_{\mathrm{W}}(\vec{x}, \vec{\mu}_c) = \|\boldsymbol{W}\vec{x} - \boldsymbol{W}\vec{\mu}_c\|_2^2$ and setting $\vec{\tilde{x}} = \boldsymbol{W}\vec{x}$ and $\vec{\tilde{\mu}}_c = \boldsymbol{W}\vec{\mu}_c$, the distance becomes

$$d_{\mathrm{W}}(\vec{x}, \vec{\mu}_c) = \left\| \vec{\tilde{x}} - \vec{\tilde{\mu}}_c \right\|_2^2 = \vec{\tilde{x}}^T \vec{\tilde{x}} - \vec{\tilde{x}}^T \vec{\tilde{\mu}}_c - \vec{\tilde{\mu}}_c^T \vec{\tilde{x}} + \vec{\tilde{\mu}}_c^T \vec{\tilde{\mu}}_c. \tag{7.15}$$

Applying the transformation to a higher-dimensional space to $\vec{\tilde{x}}$ and $\vec{\tilde{\mu}}_c$, respectively, yields the kernel based distance

$$d_{\mathrm{Kernel}}(\vec{\tilde{x}}, \vec{\tilde{\mu}}_c) = K(\vec{\tilde{x}}, \vec{\tilde{x}}) - K(\vec{\tilde{x}}, \vec{\tilde{\mu}}_c) - K(\vec{\tilde{\mu}}_c, \vec{\tilde{x}}) + K(\vec{\tilde{\mu}}_c, \vec{\tilde{\mu}}_c), \tag{7.16}$$

which, in the case of a Gaussian RBF, is given by

$$d_{\mathrm{rbf}}(\vec{x}, \vec{\mu}_c) = 2 - 2 \exp\left(- \frac{\| \boldsymbol{W} \vec{x} - \boldsymbol{W} \vec{\mu}_c \|^2}{2 {\sigma_{\mathrm{rbf}}}^2} \right). \tag{7.17}$$

Adopting the approach of Mensink et al. (2013) we obtain the posterior probability

$$p(c|\vec{x}) = \frac{\exp\left(-\frac{1}{2} d_{\mathrm{rbf}}(\vec{x}, \vec{\mu}_c)\right)}{\sum_{\tilde{c}=1}^{N_{\mathrm{c}}} \exp\left(-\frac{1}{2} d_{\mathrm{rbf}}(\vec{x}, \vec{\mu}_{\tilde{c}})\right)} \tag{7.18}$$

and compute the matrix $\boldsymbol{W}$ by maximizing the log-posterior of the correct prediction using gradient ascent. Further discussion was submitted by the master thesis of Migdadi (2015), who was supervised by the author of this thesis. Fig. 7.4, shows the flowchart of the proposed method.

7.2.4 Discussion of experiments results

The proposed algorithm has the ability to solve linearly inseparable multi-class problems within a short time of processing. The evaluation process is implemented in two types of experiments; training the compared classifiers in a fully supervised and a semi-supervised manner. Both Gesture dataset and Iris dataset are used in the experiment of the fully supervised learning and only the Gesture data used in the semi-supervised learning. In each experiment, the resulting accuracy and runtime of the KNCM are compared to the original NCMC code[1] by Mensink et al. (2013).

[1]The code of the NCMC is available at https://staff.fnwi.uva.nl/t.e.j.mensink/code.php

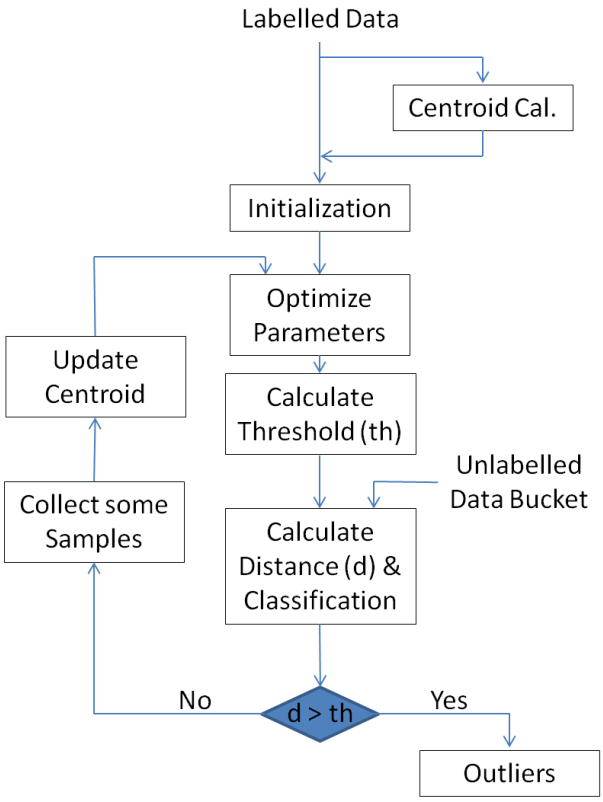


Figure 7.4: The flowchart of the proposed Kernel Nearest Class Mean algorithm

.

Fully supervised learning experiments

The fully supervised experiment focuses on the evaluation of the relation between the accuracy and the training data size. Different fractions of 10%, 20%, 30%, 40%, 50%, and 60% of the total size of the dataset are used for training, respectively. The test data size is set to 40% of the total size in all cases. The partition into training and test datasets follow a random permutation. Since the number of Iris data is low, the number of centroids per class in the NCMC method is set to 1. Each experiment is repeated 100 times with different random permutations to derive an expectation of the accuracy that is independent of the partition. Both classifiers use the same random permutation for each experiment during these 100 experiments.

The results in terms of classification accuracy and runtime is shown in Table 7.1 and Table 7.2, respectively. In most cases, the accuracy of the proposed KNCM exceeds the accuracy of the NCMC algorithm by more than five standard deviations. Furthermore, the accuracy of the KNCM shows only a weak dependence on the dimensionality of the feature vector. Consequently, the computational expense of the introduced kernel function

Table 7.1: The average accuracy of KNCM and NCMC for the different data sizes and dimensionality d' of $\boldsymbol{W}$, respectively. The deviations are the standard deviation over all 100 runs.

Dataset	d'	Classifier	Training Data Size					
			10%	20%	30%	40%	50%	60%
Gest. dataset	2	KNCM	0.98 ±0.01	0.98 ±0.08	0.98 ±0.05	0.99 ±0.01	0.98 ±0.07	0.96 ±0.12
		NCMC	0.63 ±0.02	0.63 ±0.01	0.63 ±0.01	0.63 ±0.01	0.63 ±0.01	0.63 ±0.01
	4	KNCM	0.99 ±0.01	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00	0.99 ±0.00
		NCMC	0.86 ±0.01	0.86 ±0.01	0.86 ±0.01	0.86 ±0.01	0.86 ±0.01	0.86 ±0.01
	8	KNCM	0.99 ±0.01	0.99 ±0.01	0.99 ±0.00	0.99 ±0.01	0.99 ±0.00	0.99 ±0.00
		NCMC	0.90 ±0.01	0.90 ±0.01	0.90 ±0.01	0.91 ±0.01	0.91 ±0.01	0.91 ±0.01
Iris dataset	2	KNCM	0.89 ±0.10	0.93 ±0.10	0.93 ±0.11	0.92 ±0.14	0.95 ±0.10	0.94 ±0.11
		NCMC	0.61 ±0.08	0.63 ±0.05	0.63 ±0.06	0.63 ±0.05	0.63 ±0.05	0.64 ±0.05
	4	KNCM	0.91 ±0.07	0.95 ±0.03	0.96 ±0.03	0.96 ±0.03	0.97 ±0.03	0.97 ±0.03
		NCMC	0.88 ±0.04	0.90 ±0.03	0.89 ±0.04	0.89 ±0.03	0.89 ±0.03	0.89 ±0.03

is covered by the lower dimensionality of the subspace projection. The lower subspace dimension, in addition, requires fewer parameters of the matrix $\boldsymbol{W}$ to be estimated. Both algorithms show no significant influenced by the amount of training data. The time consumed by the training of the NCMC is approximately constant for all values of d' and training datasets, respectively. This is related to the stochastic gradient descent algorithm applied by Mensink et al. (2013): A constant number of samples is drawn with replacement from the training dataset. The gradient descent is then executed for 1000 iterations. Consequently, the number of function evaluations in the optimisation procedure is almost constant. The gradient descent approach, in contrast, terminates if the relative increase in the target function is less than 10^{-6} and uses all training data. Consequently, the training time of the KNCM increases with the amount of training data. Unfortunately, this does not allow for a direct comparison without modifications of the code by Mensink et al. (2013) and is thus subject to further analysis. The time of the recall phase, however, is comparable to both methods and shows that the expected runtime of the KNCM is similar to the runtime of the NCM. Thus, the KNCM is supposed to perform as well in online or large-scale scenarios as the NCM.

Table 7.2: The average runtime of KNCM and NCMC for the different data sizes and dimensionality d' of $\boldsymbol{W}$, respectively. The deviations are the standard deviation over all 100 runs. The test or recall phase is not included in the runtime measurements and thus given in the last column.

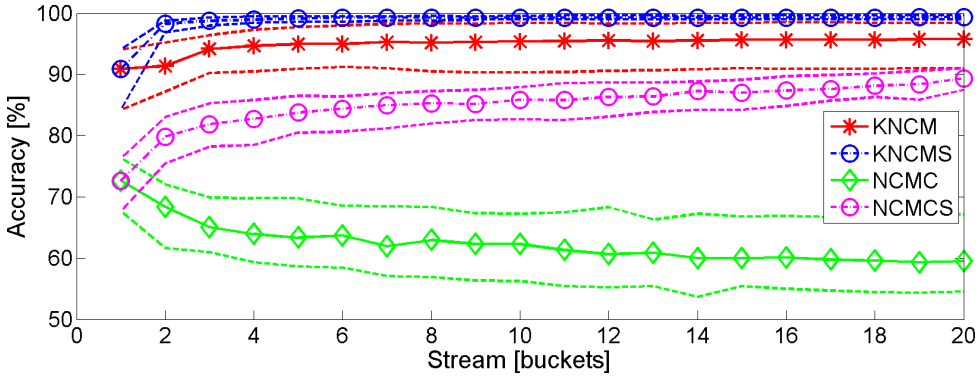
Dataset	d'	Classifier	Training Phase/Data Size						Testing Phase
			10%	20%	30%	40%	50%	60%	
Gest. dataset	2	KNCM	0.20 ±0.14	0.28 ±0.24	0.37 ±0.32	0.36 ±0.30	0.44 ±0.40	0.40 ±0.34	4.92×10^{-4} $\pm 0.31 \times 10^{-4}$
		NCMC	0.83 ±0.02	0.82 ±0.01	0.82 ±0.02	0.82 ±0.01	0.81 ±0.01	0.82 ±0.01	4.44×10^{-4} $\pm 0.14 \times 10^{-4}$
	4	KNCM	0.21 ±0.02	0.31 ±0.02	0.42 ±0.03	0.43 ±0.03	0.53 ±0.03	0.66 ±0.03	5.68×10^{-4} $\pm 0.52 \times 10^{-4}$
		NCMC	0.86 ±0.01	0.86 ±0.01	0.86 ±0.01	0.85 ±0.01	0.85 ±0.01	0.86 ±0.01	5.56×10^{-4} $\pm 0.31 \times 10^{-4}$
	8	KNCM	0.30 ±0.07	0.44 ±0.12	0.65 ±0.13	0.67 ±0.17	0.85 ±0.18	1.02 ±0.18	7.29×10^{-4} $\pm 0.83 \times 10^{-4}$
		NCMC	0.91 ±0.01	0.91 ±0.01	0.91 ±0.01	0.92 ±0.01	0.91 ±0.01	0.92 ±0.01	6.48×10^{-4} $\pm 0.25 \times 10^{-4}$
Iris dataset	2	KNCM	0.01 ±0.01	0.03 ±0.03	0.03 ±0.03	0.04 ±0.03	0.05 ±0.03	0.07 ±0.03	1.10×10^{-4} $\pm 0.11 \times 10^{-4}$
		NCMC	0.55 ±0.00	0.55 ±0.00	0.55 ±0.01	0.55 ±0.01	0.55 ±0.01	0.55 ±0.00	1.12×10^{-4} $\pm 0.10 \times 10^{-4}$
	4	KNCM	0.02 ±0.02	0.05 ±0.05	0.07 ±0.06	0.09 ±0.07	0.11 ±0.06	0.13 ±0.06	1.20×10^{-4} $\pm 0.08 \times 10^{-4}$
		NCMC	0.63 ±0.01	0.62 ±0.00	0.62 ±0.01	0.62 ±0.00	0.62 ±0.01	0.62 ±0.00	1.19×10^{-4} $\pm 0.10 \times 10^{-4}$

Semi-supervised learning experiments

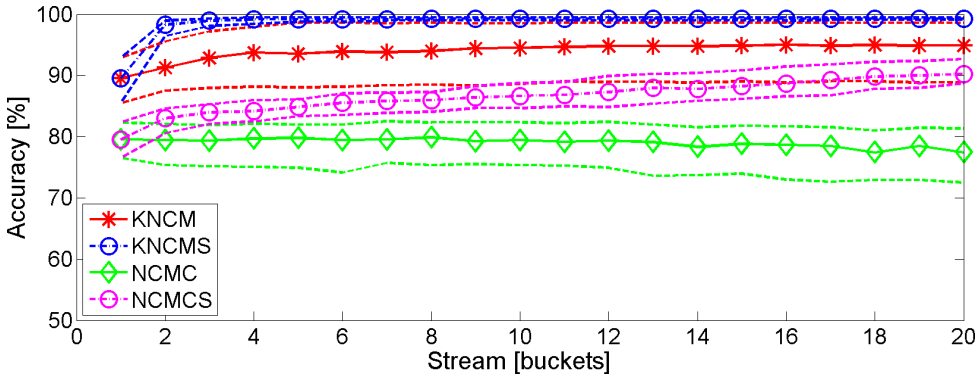
Since falsely assigned labels have a strong effect on the performance of a semi-supervised learning algorithm, we introduce a confidence threshold and reject possible outliers, i.e. samples exceeding a distance threshold. The labels of samples that do not exceed the threshold are added to the training dataset of the classifier. The threshold is based on an independent validation dataset. Consequently, the full dataset in the semi-supervised learning experiment is subdivided into four parts: a labelled initial training set, a labelled validation set, an unlabelled learning set and a labelled test set.

At the beginning of the experiment, the classifiers are adapted to the initial training set, and the confidence threshold is computed based on the validation set. The learning set is further subdivided into so-called "buckets" that represent a stream of data. The buckets are presented to the classifiers one by one. The classifiers then assign labels to the samples contained in the bucket, respectively, and add the samples that do not exceed the training threshold with the assigned labels to the training data. Then the classifiers are adapted to the extended training set, and a new confidence threshold is computed.

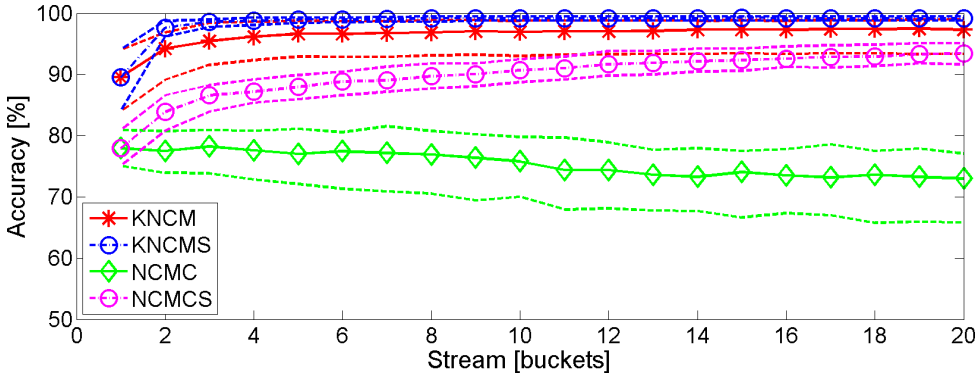
This process is repeated until the last bucket has been presented to the classifiers.



(a)

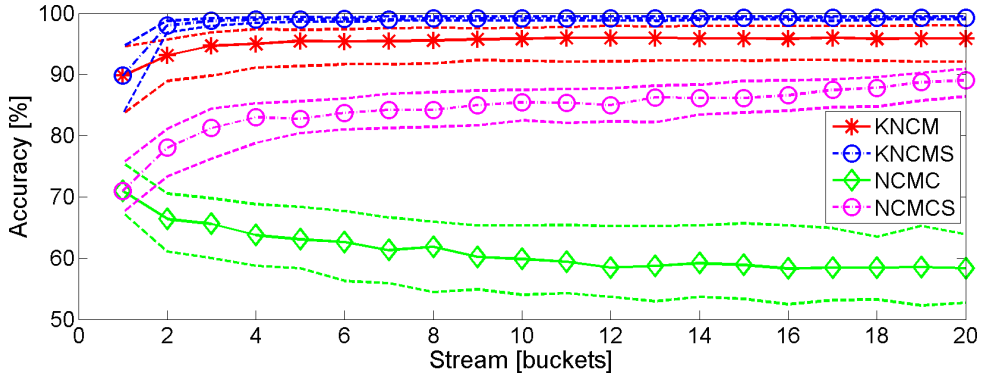


(b)



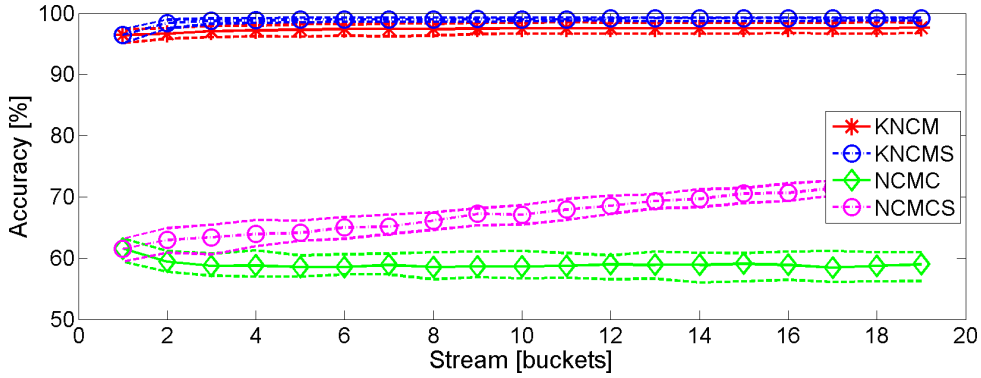
(c)

Figure 7.5: Median prediction accuracy of the classifiers for an initial training set comprising 1% of the total data. KNCMS and NCMCS denote the supervised version of the KNCM and NCMC, respectively. The dashed lines correspond to the 25% and 75% quantiles and represent the spread over 100 repetitions. These sub-figures show the accuracy of the classifiers at (a) $d' = 2$. (b) $d' = 4$. (c) $d' = 6$. (d) $d' = 8$.

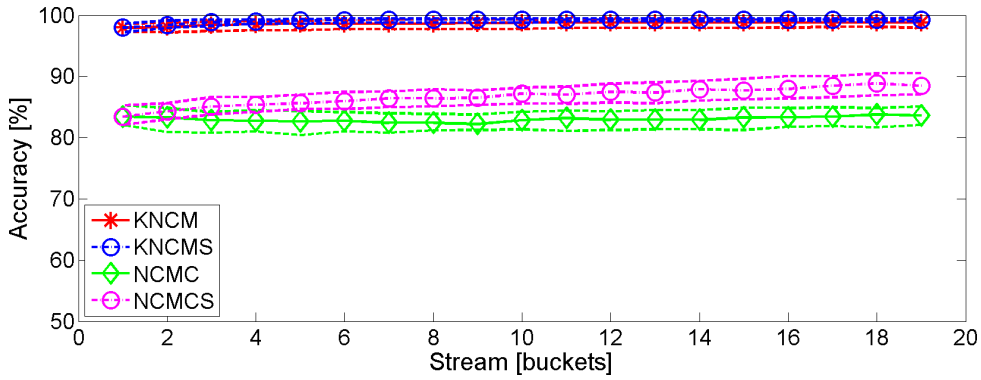


(d)

Figure 7.5: (Continued)

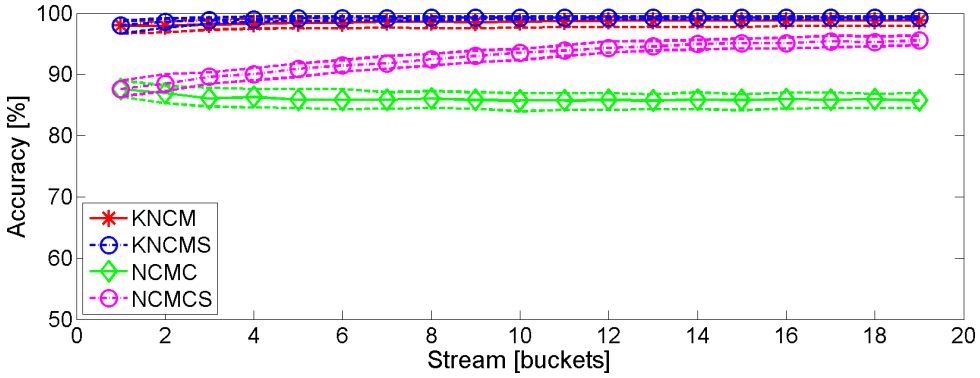


(a)

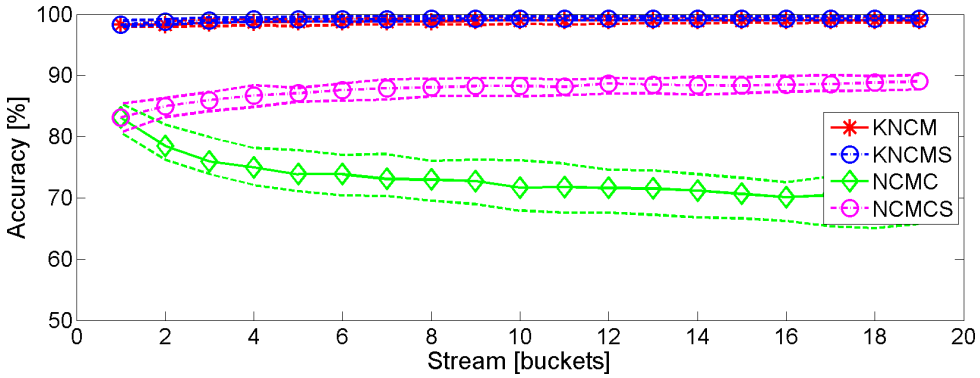


(b)

Figure 7.6: Median prediction accuracy of the classifiers for 5% of the total data used as initial training set. KNCMS and NCMCS denote the supervised version of the KNCM and NCMC, respectively. (a) $d' = 2$. (b) $d' = 4$. (c) $d' = 6$. (d) $d' = 8$.



(c)



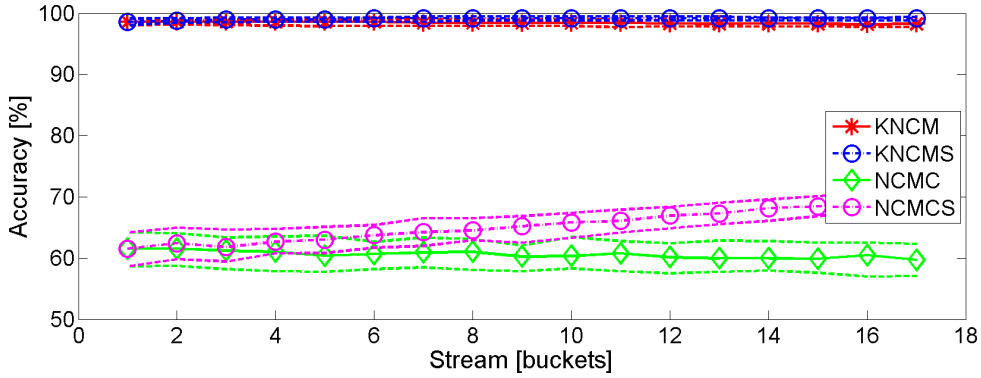
(d)

Figure 7.6: (Continued)

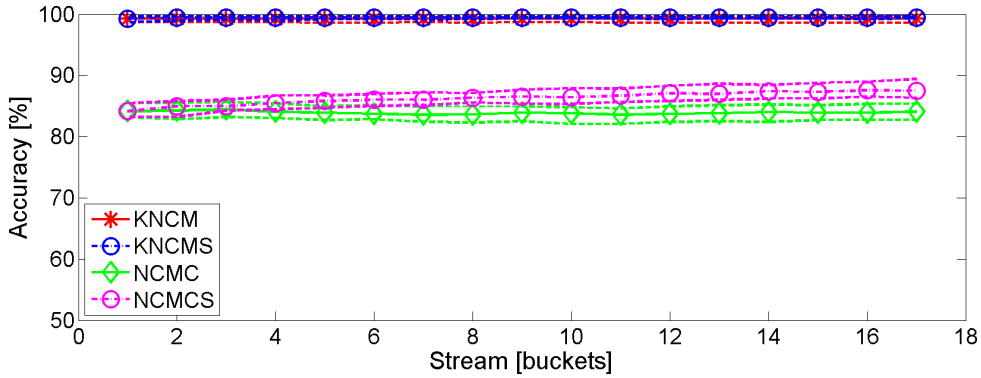
After each training process of the classifiers, we evaluate the accuracy of the classifiers based on the test set. In addition to the accuracy, we track the computation time required by both training and prediction processes, and the training set size. Additionally, we train the second version of each classifier with the correct labels to evaluate the performance of a fully supervised learning scenario in each step.

The semi-supervised learning experiment is repeated for three different sizes of the initial training set only: 1%, 5% and 10% of the total dataset. In all these experiments, the test and the validation set comprise 20% and 15% of the total dataset, respectively. The remainder is the unlabelled learning set, which is subdivided into buckets of 100 samples each. The sets are formed using a class-wise random partition. Since each of the nine gestures in the dataset is represented by a different number of samples, the number of samples in the initial training set may be as low as two samples in the case of class nine and an initial training set comprising 1% of the total data.

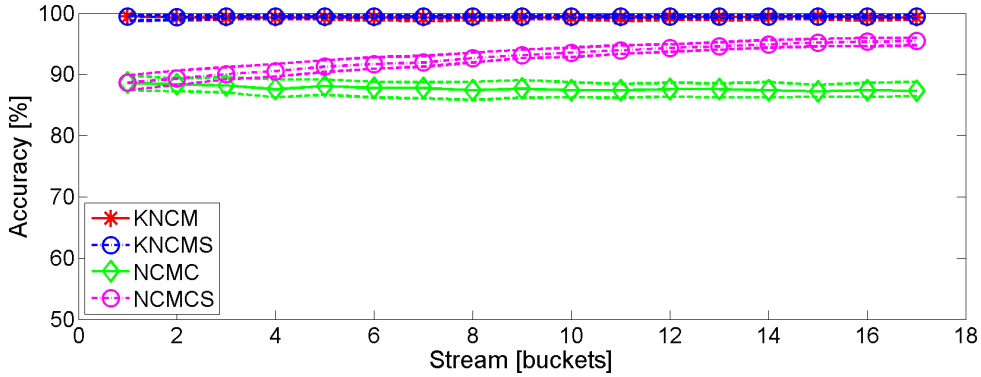
Each experiment is repeated for four different dimensions of the subspace $d' = \{2, 4, 6, 8\}$,



(a)

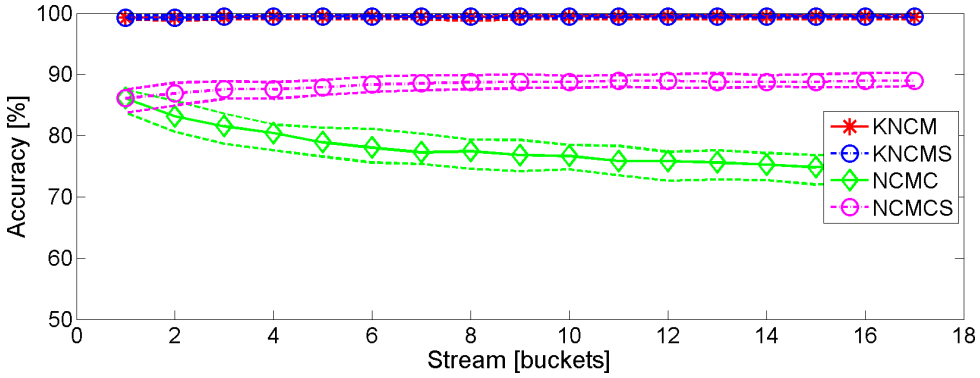


(b)



(c)

Figure 7.7: Median prediction accuracy of the classifiers for 10% of the total data used as initial training set. KNCMS and NCMCS denote the supervised version of the KNCM and NCMC, respectively. (a) The accuracy of the classifiers at $d' = 2$. (b) The accuracy of the classifiers at $d' = 4$. (c) The accuracy of the classifiers at $d' = 6$. (d) The accuracy of the classifiers at $d' = 8$.



(d)

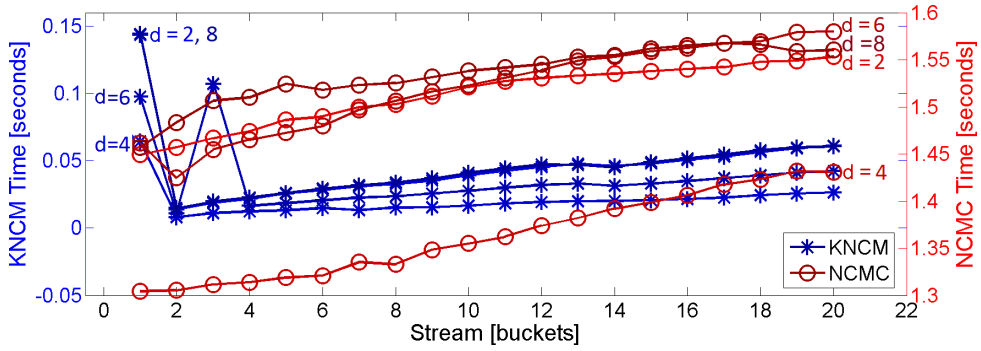
Figure 7.7: (Continued)

respectively, resulting in twelve different experiments. The experiments are repeated 100 times with a different random permutation. Both classifiers use the same random permutation for each experiment during these 100 repetitions.

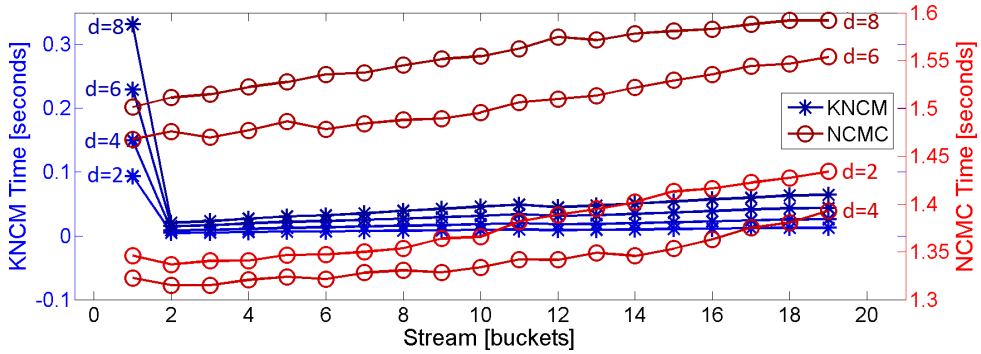
The NCMC requires the specification of the number of centroids per class N_{cent}. There is, however, a function within the utilised code package published by Mensink et al. (2013) that computes the optimal number of centroids per class. We apply this code to specify the best N_{cent} for each projection matrix dimension. The resulting values were $N_{\text{cent}} = 2$ for the four dimensions of the subspace, respectively. In case of $N_{\text{cent}} > N_{\text{c}}$, i.e. the number of class c samples in the initial training is smaller than the number of centroids, we start by setting $N_{\text{cent}} = N_{\text{c}}$ and then gradually increase N_{cent} until it equals the optimal value.

Fig. 7.5–Fig. 7.7 show the prediction accuracy of KNCM and NCMC. Notably, the first bucket corresponds to the initial training set. The remaining results thus may be directly compared to the result obtained by using the initial training set. For large training sets (see Fig. 7.7), the accuracy of the KNCM equals the accuracy of the supervised version. The high initial accuracy is kept throughout the learning process. If the size of the training set is reduced (see Fig. 7.5–Fig. 7.6) the accuracy of the semi-supervised KNCM does not reach the optimal value of the fully supervised KNCM. It, however, increases strongly over the first few buckets and approaches the fully supervised KNCM. The difference between the semi-supervised KNCM and the fully supervised KNCM increases with decreasing dimensions of the projection subspace. However, the effect of the subspace dimension on the KNCM accuracy is rather subtle and thus it is possible to use subspace projections of very low dimension.

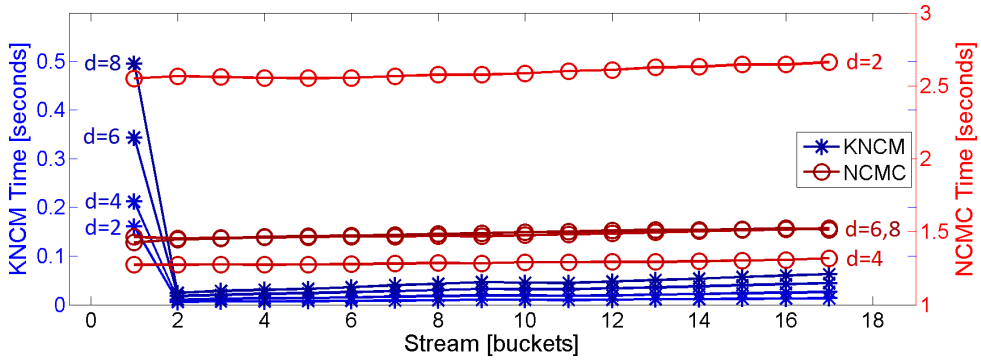
The accuracy of the NCMC, in contrast, shows a larger difference to its fully supervised counterpart if the number of presented buckets increases. While the fully supervised NCMC shows an increased accuracy in the course of the learning experiment, the accuracy of the semi-supervised NCMC seems constant in the best case and exhibits a strong



(a)

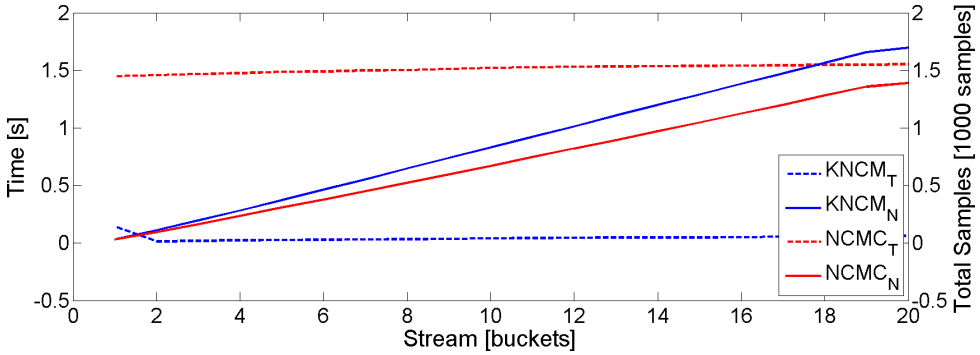


(b)



(c)

Figure 7.8: Time consumed by the classifiers. (a) Initial training set: 1% of total data. (b) Initial training set: 5% of total data. (c) Initial training set: 10% of total data. (d) Runtime (KNCM_T and NCMC_T) and samples in the training set (KNCM_N and NCMC_N) for the KNCM and the NCMC, respectively. The initial training set size is 1% of the total data, and the dimensionality is $d' = 2$.



(d)

Figure 7.8: (Continued)

decrease in some experiments. Both the semi-supervised NCMC and the fully supervised NCMC do not reach the accuracy of the KNCM in all experiments. The decreasing accuracy of the NCMC suggests that the NCMC adds false labels to the training set. This may be due to the low initial prediction accuracy. The effect is less noticeable if the size of the training set increases. Furthermore, this effect seems to be strong for both the full dimension of the data set and a very small subspace dimension. The former may be related to the estimation of many parameters while the latter may be due to in insufficient subspace dimension that does not allow for a separation of the classes.

In addition to the gain in prediction accuracy, the runtime of the KNCM is considerably lower than the runtime of the NCMC, as shown in Fig. 7.8. The decreasing accuracy of the NCMC suggests that the NCMC method results in the addition of the false labels to the training set, possibly leading to a larger total number of samples. However, the opposite is true. Fig. 7.8(d) shows the median runtime and the median of the training set size for an initial training set comprising 1% of the total data and a dimensionality of $d' = 2$. This exemplary semi-supervised learning progress is similar to the other experiments. Fig. 7.8(d) clearly shows that the runtime is independent of the training set size.

8

Contribution: Semi-supervised Learning Based on Self-adaptive Structure

In the previous chapters, we have discussed the incremental learning (INC). If the classifier is trained on some classes, then it can be updated for new data that belong to the known classes without a need of the old training data. The question here is what will happen if we get some samples belonging to new concepts or classes. This may occur in the data stream with a high probability. The known concept in the data stream may consist of more than one class, and different concepts may be detected in the course of time. Hence, classifying a new sample as known or unknown is not enough since both of them are composed of several classes. Thus, the model structure cannot be static, but it should be dynamically updated to represent new concepts.

The incremental learning is crucial to update the classifier knowledge, particularly in the smart environments such as the robotics, which experience a need to be updated continuously according to the environment of the robot. Commonly, smart devices or robotics use the gesture applications. Let us ask ourselves, is the incremental learning enough to build an intelligent environment? To answer this question let us discuss this example: if children hear a new word or see a new activity, they ask their parents about the meaning (the label) of that action. The worthy note here is that the children had already recognised and learnt the word or the activity before they asked their parents about its meaning. This learning type is the great secret of the human intelligence of learning from the environment around. Therefore, we think the machine will be more intelligent if we could implement the similar technique in the smart devices.

In this chapter, we extend the incremental learning of the proposed classifiers to be able to learn new concepts without a need for the old data of the known classes. The proposed classifiers assume an arbitrary label for the new concept; retaining its right to ask about the meaning of the new data. Commonly, the incremental class learning is implemented in supervised learning. However, we propose a method to autonomously

add the emergent classes. The traditional method converts the data (gestures) to feature vectors and the meaning of those activities (gestures) to labels (mostly integer numbers), then the feature vectors and their labels are provided to train the classifier. The proposed method is trained on the known feature vectors and their labels as the traditional methods and suggests new labels for the new concepts. Then, the classifier retrains itself on the new feature vectors and their proposed label. A meaning for the new labels still needs to be provided later at least once, exactly similar to children when asking about the meaning. This process is summarised in Fig. 8.1.

Additionally, the proposed method is able to discriminate among the samples that belong to the same or different concepts. Thus, the various novelties, i.e. samples that belong to different concepts, are added as separated classes. The outliers are not considered as a novelty, i.e. random movements or noise samples are discarded since the novelty of the concept is composed of a group of samples more than a specific number (in our experiment it is equal to 15 samples), which are cohesive and representative. Combining the incremental class learning with the ability to discriminate the novelties makes the system be learned completely autonomous. Hence, we called this learning system as "**s**elf-adaptive **s**tructure **s**emi-**s**upervised **l**earning (S4L)".

8.1 Incremental Class Learning

The incremental class learning (ICL) is a competitive innovation in the machine learning, in particular for the complex and the open-ended problems (Mańdziuk and Shastri, 2002). In addition to being essential in the application related to the data stream, it provides a practical solution for scalable learning systems, where the complex learning problem can be decomposed into learning sub-problems incrementally, one at a time. For example, the model is trained for a limited number of categories and then incremented for new categories subsequently. Thus, the ICL configures a learning method that supports the sharing of previously learned knowledge structures (Mańdziuk and Shastri, 2002). Here, the proposed classifiers are extended to be able to adapt their classes structure. Since the proposed classifiers are different in the structure, the extension to new classes and the evaluation of each classifier is discussed individually.

The gesture data are used to evaluate the incremental class learning. The same evaluation process is applied to all classifiers. Hence, the experiment is explained once as follows: The complete data are divided into two sets, training set and testing set. The training set is subdivided into two sets where the first set contains the data that belong to only three classes, and the other set includes the rest of the classes. These three classes are selected randomly from the nine classes of the gesture data. The classifier is trained on the selected three classes and evaluated on the test set. Afterwards, the second set (the rest of 6 classes) is grouped according to their classes. The data of one class are used to update the classifier knowledge on this new class and evaluate the classifier on

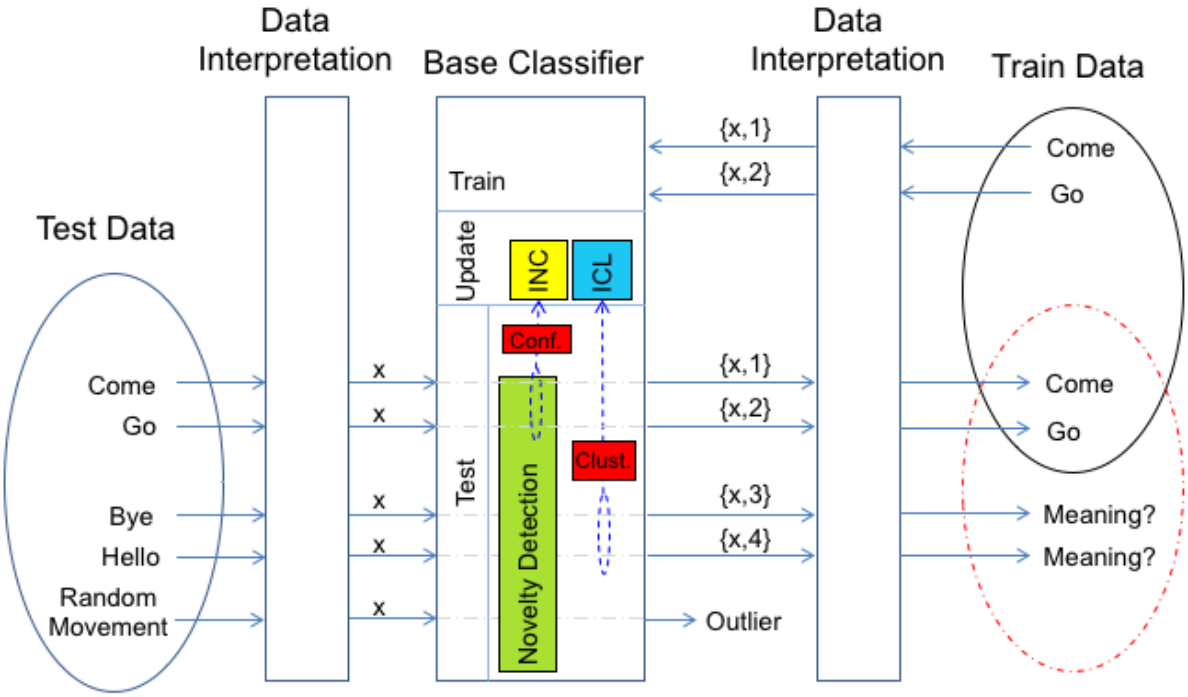


Figure 8.1: The proposed classification process. The training gesture data are interpreted as feature vectors and integer values (labels), which represents the gesture types of these vectors, then they are provided to train the classifier. Only the feature vector of the new (testing) gesture is provided to the classifier to estimate its label that corresponds to the most similar gesture type. Traditional classifiers are restricted to the training types. The standard semi-supervised learning aims to add these new samples and their estimated labels to the training data. Notably, the intersection between the dashed red and black circles on the training data side represents the aim of the semi-supervised learning. The proposed method provides additional facilities (highlighted with blue) to the standard semi-supervised techniques, which suggests new label(s) for the new concept(s) and then asks about the meaning if the definition is required. Additionally, it can detect more than one new concept, and it also can identify the random movements, e.g. "bye" and "hello" gestures, and the random movements, respectively. Finally, the initial training dataset is extended to the new data and labels (dashed red line). The new samples from the test data are used to update the classifier knowledge, where the classifier updated on (come, go) samples using the standard incremental learning (INC) and on the (bye, hello) using the incremental class learning (ICL).

the test set. The same procedure is repeated till the last class in the second set is provided. The classifier updates its knowledge on the new classes incrementally. In parallel, we used a second version of the same classifier, but it is trained in a classical way, i.e. each time is trained with initial training data in addition to the data of the added class. This experiment is repeated for 100 times since it depends on random a permutation. In each iteration, different random three classes are used for the initial training. The same metrics that were used to evaluate the proposed algorithms (M_{new}, F_{new}, E_{total}, T_{update}) are used here. However, instead of using the values itself we used the difference between the metrics of the classifiers, which are the incremental classifier and the classifier that used all data to train itself. The difference between the metrics of the classifiers is used since the evaluation of the proposed methods is already studied in the last chapters. Additionally, we focus here on maintaining the same accuracy of the classic classifier with the ability to learn new concepts incrementally.

In all result tables, the first column is the difference between the two classifiers' metrics when trained on the initial training data. The other classes are added subsequently, and the difference between the classifiers' metrics are computed in the other columns. For each metric, the quantile of 25%, the median and the quantile of 75% of the 100 runs are computed. The minus sign in the tables means that the (error) metrics of the incremental class classifier is less than the metrics of the standard classifier.

8.1.1 Extreme learning machine – EVT

The output of the ELM is computed by Eq. (6.5), where $\boldsymbol{\beta}_{\text{E}}$ is the main parameter in the equation. The parameter $\boldsymbol{P}_{\text{E}}$ in the Eq. (6.10) is the only parameter that is related to the labels of the training data ($\boldsymbol{T}_{\text{E}}$) as in Eq. (6.9). Hence, $\boldsymbol{\beta}_{\text{E}}$ can be updated to new classes by updating the parameter $\boldsymbol{P}_{\text{E}}$. The rows of $\boldsymbol{T}_{\text{E}}$ are vectors of length equal to a number of the classes, and each vector contains 1 in the place of the class where the sample belong and -1 elsewhere. That means it is simply updated to the new class by adding a new column with value -1. But we cannot store the matrix $\boldsymbol{T}_{\text{E}}$, since it is enlarged with each updating. Instead, we already store the matrix $\boldsymbol{P}_{\text{E}}$, hence we need to update the matrix $\boldsymbol{P}_{\text{E}}$.

$$\boldsymbol{P}_{\text{E}}(i,j) = \Sigma_{i,j}\boldsymbol{H}_{\text{E}}(j,i) \cdot \boldsymbol{T}_{\text{E}}(i,j) \tag{8.1}$$

Since all values of the new column in the $\boldsymbol{T}_{\text{E}}$ is equal to -1, the equation will be reduced to the negative sum of the columns of $\boldsymbol{H}_{\text{E}}$ resulting the values in the new column of the $\boldsymbol{P}_{\text{E}}$ is the sum of the corresponding neurons output of all samples i.e. a vector of length equal to the number of neurons. Therefore, to add one class to the learned classifier, we just need to add a new column to the $\boldsymbol{P}_{\text{E}}$ with values equal to the sum of the neurons' output of the old samples, which it has a length equal to the number of rows in $\boldsymbol{P}_{\text{E}}$. To add more than one class we need to add columns equal to the number of new

classes. Table 8.1 summarises the results of the experiments that explained in Section 8.1. For each metrics, we calculate the quantile of 25%, the median and the quantile of 75%. The first column represents the result of the test set when the classifiers are trained only for 3 classes. In the next columns, we added one class in each iteration and evaluate the classifiers on test data. Hence in the last column, all classes were added. As expected, the runtime of the proposed method (for 25%, 50% and 75% quantile) is less than the standard version of the classifier as shown in Table 8.1, where the object of the method is to reduce the runtime. The difference of the F_{new} between the proposed method and the standard method is exactly zero, which means it does not effected by reducing the time of the processing. Most values of the M_{new} are negative which means the error in the suggested method is less than the standard classifier. The positive values of the median are less than 0.5%. Similarly, the values of E_{total} are mostly negative and the positive values of the median, if exist, are less than 0.2%. Hence, the proposed method reduces the processing time of adding new class while maintaining the or improving the accuracy, although we expect the slight differences in the accuracy is due to the fact that the ELM randomly generates hidden neurons weights.

8.1.2 Extreme learning machine – auto encoder

Essentially, the classifier of the auto-encoder ELM has two different ELM classifiers. The first classifier is to estimate the labels only, which it uses the data and their labels in training, and the second classifier does not use the labels since it is implemented as auto-encoder to guess the novelty.

The first classifier is extended to be incremental class exactly as in Section 8.1.1 by adding new columns equal to the new classes number and consist values equal to the sum of the columns of $\boldsymbol{H}_{\text{E}}$. The second classifier used the data itself as a target hence

Table 8.1: The difference between the metrics of the incremental class ELM-EVT classifier and the classical ELM-EVT classifier.

Metric		$C1 - C3$	C4	C5	C6	C7	C8	C9
T_{update} (msec)	quantile 25%	0.00	−3.22	−4.12	−4.56	−5.58	−5.81	−7.15
	median	0.00	−2.42	−2.37	−3.05	−4.04	−4.57	−5.56
	quantile 75%	0.00	−0.80	−1.39	−1.83	−2.86	−3.19	−4.17
E_{total} [%]	quantile 25%	0.00	−0.69	−1.89	−1.89	−3.10	−2.84	0.00
	median	0.00	0.17	−0.09	−0.26	0.17	−0.09	0.00
	quantile 75%	0.00	1.12	1.20	3.10	1.72	1.81	0.00
M_{new} [%]	quantile 25%	0.00	−1.21	−3.48	−6.05	−10.74	−16.50	–
	median	0.00	0.25	−0.14	−0.69	0.50	−1.00	–
	quantile 75%	0.00	1.69	2.41	7.50	6.70	13.50	–
F_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	median	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	0.00

the dimension of the parameters will not be changed if the number of classes is changed since it is not related to the labels of the data. However, the average of the data is used in the normalisation process hence it is updated using the Eq. (7.1). The evaluation results of the experiments that explained in Section 8.1 shows that the incremental class classifier accuracy is almost same the classical classifier. The negative values in the results Table 8.2 means the error of the ICL classifier is less than the other classifier. However, these negative values are just less than 1%, and we expect that difference because the ELM is randomly generating hidden neurons weights.

Table 8.2: The difference between the metrics of the incremental class ELM-AE classifier and the classical ELM-AE classifier.

Metric		$C1 - C3$	C4	C5	C6	C7	C8	C9
T_{update} (msec)	quantile 25%	0.00	−3.76	−4.14	−5.03	−5.45	−6.24	−7.31
	median	0.00	−2.38	−2.96	−3.80	−4.50	−5.46	−6.20
	quantile 75%	0.00	−1.30	−2.17	−3.01	−3.35	−4.26	−4.99
E_{total} [%]	quantile 25%	0.00	−0.52	−0.69	−1.55	−1.72	−0.26	0.00
	median	0.00	0.00	0.17	−0.17	0.17	0.34	0.34
	quantile 75%	0.00	0.26	0.69	0.52	0.52	0.60	0.69
M_{new} [%]	quantile 25%	0.00	−1.03	−1.83	−4.15	−6.72	−1.00	–
	median	0.00	0.00	0.00	−1.25	−0.81	0.00	–
	quantile 75%	0.00	0.00	0.37	0.44	0.00	1.00	–
F_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	0.17
	median	0.00	0.40	0.65	0.54	0.47	0.39	0.34
	quantile 75%	0.00	0.94	1.14	0.99	0.79	0.62	0.69

8.1.3 Metric learning – Mahalanobis distance

Since the Mahalanobis distance classifier parameters are only the mean and the covariance matrix, it can be extended to new classes by computing the mean and the covariance matrix of the new classes. The experiments that explained in Section 8.1 shows that the incremental class classifier performs exactly same accuracy of the classical classifier while it reduces the time of the processing more than 26 msec at using the whole data (see Table 8.3).

8.1.4 Polynomial classifier

Similar to the ELM classifier, the parameter matrix is $\boldsymbol{\beta}_{\text{P}} = {\boldsymbol{M}_{\text{P}}}^{-1} \cdot \boldsymbol{P}_{\text{P}}$ and $\boldsymbol{P}_{\text{P}} = {\boldsymbol{H}_{\text{P}}}^{T} \cdot \boldsymbol{T}_{\text{P}}$. Hence

$$\boldsymbol{P}_{\text{P}}(i,j) = \Sigma_{i,j} \boldsymbol{H}_{\text{P}}(j,i) \cdot \boldsymbol{T}_{\text{P}}(i,j) \tag{8.2}$$

Unlike the ELM, the target matrix $\boldsymbol{T}_{\text{P}}$ consist 1 in the place of the corresponding class and 0 elsewhere. Hence, when new classes need to be added, only zeros columns should

Table 8.3: The difference between the metrics of the incremental class Mahalanobis distance classifier and the classical Mahalanobis distance classifier.

Metric		$C1 - C3$	C4	C5	C6	C7	C8	C9
T_{update} (msec)	quantile 25%	0.00	−10.24	−13.74	−18.55	−21.99	−26.27	−28.45
	median	0.00	−9.60	−12.90	−16.40	−19.53	−22.72	−26.07
	quantile 75%	0.00	−8.83	−12.28	−15.47	−18.51	−21.73	−24.67
E_{total} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	median	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
M_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	–
	median	0.00	0.00	0.00	0.00	0.00	0.00	–
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	–
F_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	median	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	0.00

be added to the $\boldsymbol{P}_{\text{P}}$. The results of the evaluations show the accuracy is approximately equal to the classic classifier, while there is a difference in the time of processing where the ICL classifier is faster than the classic classifier of about 5 msec (see Table 8.4).

Table 8.4: The difference between the metrics of the incremental class polynomial classifier and the classical polynomial classifier.

Metric		$C1 - C3$	C4	C5	C6	C7	C8	C9
T_{update} (msec)	quantile 25%	0.00	−2.78	−3.54	−3.79	−4.50	−4.90	−5.70
	median	0.00	−2.15	−2.88	−3.06	−3.86	−4.35	−5.09
	quantile 75%	0.00	−1.61	−2.32	−2.38	−3.36	−3.67	−4.44
E_{total} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	median	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
M_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	–
	median	0.00	0.00	0.00	0.00	0.00	0.00	–
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	–
F_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	median	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	0.00

8.1.5 Parzen window kernel density estimators

The main parameters in the Parzen classifier are the centroids, the number of samples per each centroid and the width of the Parzen window. It is simply extended to new classes by adding new centroids of the new classes and how many samples each contains. The Parzen width is then updated using the normal method that explained in Section 5.3.2. The evaluation experiments show that there is no loss in the accuracy of the incremental

Table 8.5: The difference between the metrics of the incremental class Parzen classifier and the classical (incremental) Parzen classifier.

Metric		$C1 - C3$	C4	C5	C6	C7	C8	C9
T_{update} (msec)	quantile 25%	0.00	−41.65	−53.64	−60.75	−69.85	−85.41	−96.46
	median	0.00	−33.85	−45.18	−51.93	−62.91	−78.07	−87.18
	quantile 75%	0.00	−27.31	−39.24	−46.38	−56.66	−69.68	−78.71
E_{total} [%]	quantile 25%	0.00	−0.69	−0.52	−0.86	−0.34	−0.69	−0.34
	median	0.00	−0.17	0.00	−0.17	−0.09	0.00	0.00
	quantile 75%	0.00	0.17	0.34	0.34	0.52	0.26	0.34
M_{new} [%]	quantile 25%	0.00	−0.44	−0.51	−0.68	0.00	0.00	–
	median	0.00	0.00	0.00	0.00	0.00	0.00	–
	quantile 75%	0.00	0.27	0.32	0.41	0.50	0.00	–
F_{new} [%]	quantile 25%	0.00	−0.89	−0.62	−0.57	−0.45	−0.62	−0.34
	median	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	quantile 75%	0.00	0.42	0.43	0.59	0.26	0.21	0.34

class Parzen classifier while gain a large difference in the processing time more than 87 msec as shown in Table 8.5.

8.1.6 Support vector machine classifier

We used the one–vs–all technique to implement the multi-class SVM classifier; hence it is able to extend to any number of classes without complex computations. For each new class, all the available data of the old classes are considered as abnormal, and the new data of the new class consider as normal. This technique simplifies the extension to new classes. However, the incremental SVM classifier store only supports vectors plus some critical data and discard unimportant data according to the old classes. For the new classes might be some of the dropped data become necessary for the new discriminator. Hence, we expect some instability of the accuracy of the SVM classifier. Though, the results of the evaluation experiments show that it is very close to the classical SVM with a large difference in the time of processing, which is more than 51 seconds (see Table 8.6).

8.2 Self-adaptive Structure Semi-supervised Learning

Although ICL is essential in the machine learning to handle the problems of data streams, it is still limited if it is trained on the new classes in supervised learning. This is because we will face the same issue of the data stream that we need experts to label all data, while we want to automate the labelling process. Thus, the intelligence of the system will be decreased to be as same as that of traditional classifiers unless an intervention of a human enforces it to add a new class in the supervised scenario. Additionally, in

Table 8.6: The difference of the metrics of the incremental class SVM classifier and the classical SVM classifier.

Metric		$C1 - C3$	C4	C5	C6	C7	C8	C9
T_{update} (sec)	quantile 25%	0.00	−2.46	−5.33	−11.82	−24.06	−43.28	−80.98
	median	0.00	−1.77	−4.10	−8.49	−15.89	−28.77	−51.72
	quantile 75%	0.00	−1.29	−2.94	−6.32	−11.94	−20.45	−38.18
E_{total} [%]	quantile 25%	0.00	0.43	0.00	0.00	0.00	−0.17	−0.17
	median	0.00	1.20	0.34	0.17	0.00	0.00	−0.17
	quantile 75%	0.00	3.18	1.03	0.69	0.17	0.00	0.00
M_{new} [%]	quantile 25%	0.00	0.72	0.00	0.00	0.00	0.00	–
	median	0.00	2.14	0.70	0.47	0.00	0.00	–
	quantile 75%	0.00	5.71	2.05	1.99	0.71	0.00	–
F_{new} [%]	quantile 25%	0.00	0.00	0.00	0.00	−0.22	−0.21	−0.17
	median	0.00	0.00	0.00	0.00	0.00	0.00	−0.17
	quantile 75%	0.00	0.00	0.00	0.00	0.00	0.00	0.00

the case of the data stream, it will be tough to assign the samples that belong to novel concepts. Furthermore, it is not appropriate for online classification. For example, the new user who is unknown to the traditional gesture system will have to adapt his/her way to perform the gestures such that it corresponds to the "knowledge" of the system. Even if the system is ICL, it is hard to the user to label all his samples to add new classes to the system. As an alternative to such a closed-world gesture recognition system, we intend to build a classification system which adapts itself to new concepts without the need for intervention of a human operator. Thus, each user can train the system with a set of own favoured gestures, and new gesture classes may be added autonomously over time. Hence, when the system perceives new gestures under specific conditions, it will construct a new class and add it to the training data. While otherwise, if the gesture category has already been seen in the training phase, the new sample will be added to the particular class in the training data according to the self-learning paradigm (Zhu and Goldberg, 2009). Various approaches have been developed to increment their database by new samples or new classes without the need for processing again the previously acquired training data, but they still need to be trained on labelled data (e.g. Zhang et al., 2006; Ditzler and Polikar, 2013).

Here, we propose a novel classification system which can integrate new samples, or new classes into its class structure autonomously and thus updates the classifier architecture without human intervention. If a new sample belongs to existing classes, it will be automatically included in the training set. Otherwise, new labels will be issued to different novelties, and new classes will be added individually to the existing classes in the training set. The proposed system can distinguish between novelties belonging to different concepts. Samples belonging to random motion patterns rather than meaningful gestures are rejected as they do not repeatedly occur in a similar manner (see Fig. 8.1).

The learning process is divided into offline and online phases (see Fig. 8.2). Only the

initial training, which uses manually labelled data of the existing classes, is implemented in the offline phase. The classifier used here can be a single classifier or can be an ensemble classifier. The ensemble classifier can be composed by combining any of our proposed classifiers. In the online phase, new samples are received from the data stream, and they are classified as one of the existing classes or as an unknown. The samples that are classified as belonging to one of the existing classes, i.e. their believability flag is already set, are added to the training data by the standard incremental learning. The unknown samples are temporarily stored until their number becomes more than a pre-defined threshold. Afterwards, the unknown samples are clustered using mean-shift method (cf.Section 3.3.2). The initial window σ_{msh}, which is used in the mean-shift method, is estimated from the known classes. The distances between each sample and the first nearest neighbour that lie within the same class are calculated; then the initial window size σ_{msh} is the 75% quantile of these distances. The selected cluster should be bigger than 10 samples to continue; else it will be discarded. If the more massive cluster was greater than 10 samples, then the distances between the distribution of the selected samples and the distribution of each class are computed using Bhattacharyya distance (Bhattachayya, 1943), which measures the similarity between two distributions. Given two normal distributions $p_i = \mathcal{N}(\vec{\mu}_i, \boldsymbol{\Sigma}_i)$, the Bhattacharyya distance is computed as follow:

$$d_{\text{bh}}(p_1, p_2) = \frac{1}{8}(\vec{\mu}_1 - \vec{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\vec{\mu}_1 - \vec{\mu}_2) + \frac{1}{2}\ln\left(\frac{\det(\boldsymbol{\Sigma})}{\sqrt{\det(\boldsymbol{\Sigma}_1)\det(\boldsymbol{\Sigma}_2)}}\right), \tag{8.3}$$

$$\text{where} \quad \boldsymbol{\Sigma} = \frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}.$$

$\det(\boldsymbol{X})$ is the determinant of the matrix $\boldsymbol{X}$. If the smallest of these distances is less than the shortest distance of the distances among the known classes each other; then these samples are considered as an extension to the nearest class, and the classifier is updated by standard incremental learning (INC). Otherwise, a new label is issued for them, and they are used to update the classifier by our incremental class learning (ICL) method. A new clustering process is implemented on the rest of the unknown samples after updating the classifier, and the same process is implemented until the largest cluster contains less than 10 samples. Since we expect that the remaining unknown samples are more scattered, we enlarge the window of the mean-shift by a factor of 1.5. The learning process is illustrated in the Fig. 8.2.

Each of the previously proposed classifiers is used to build the system individually. Additionally, an ensemble of all the classifiers altogether is built. Majority voting is used to combine the different outputs of the classifiers in the ensemble. The same evaluation experiment is applied to all these systems. The gesture data is divided into the training set, learning set and a test set with fractions of 25%, 50% and 25% of the total data,

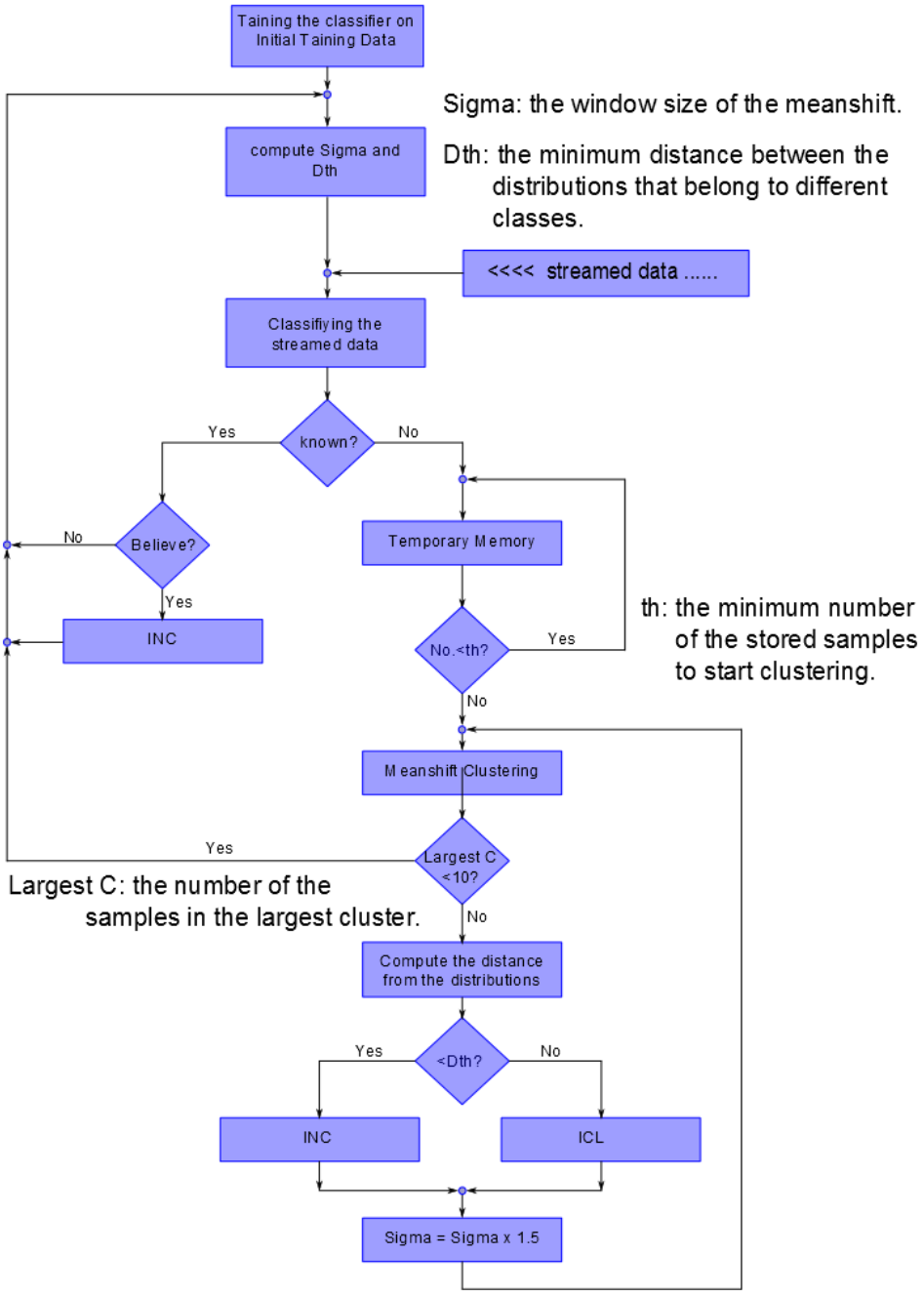


Figure 8.2: Overview of the proposed S4L algorithm. The initial training is implemented in the offline phase and all other processes are implemented in online phase. After clustering the data, only the largest cluster is selected, and it should contain 10 samples or more to continue, otherwise, it will be discarded.

respectively. Six classes are selected arbitrarily to be in the initial training, while the data of the remaining three classes are discarded from the training data. Additionally, random movements or outliers are added to the learning and tests set by using the function "gendatout" from the toolbox (ddtool) (Tax, 2015). The mean number of samples per discarded class is computed, i.e. the sum of all discarded samples divided by three (number of dropped classes). The noise added to the learning and testing sets are 20% and 10% of the computed mean number, respectively. Thus, the training set contains 25% of the samples of six classes only, respectively, while the learning and test sets include 50% and 25% of the samples of all classes, respectively, plus random movements samples sized as 20% and 10% of the average number of the discarded sample per class, respectively. Initially, the classifier is trained on the training set in the offline phase. The learning set is used as streamed data in the online phase. The samples that are classified as known samples and their believability flags are already triggered and are used to update the classifier in standard incremental learning while all the samples that are indicated as unknown are collected and submitted to the mean-shift algorithm. The largest cluster is selected and verified if it is an extension of any of the known classes by using the Bhattacharyya distance. A new label is issued for the cluster that does not belong to any of the known classes. This cluster is considered as a new class and used to update the classifier using the incremental class learning method. The process is continued until the largest cluster contains only 10 or fewer samples. Finally, the model is evaluated on the test set. The experiment is repeated 100 times with different random permutations and different random initial training classes to derive an expectation of the accuracy that is independent of the partition and the initial classes. For precise inspection, the confusion matrix is used to show the results. Additionally, the accuracy (Acc) and the reliability (Rel) are computed. Since the classes are unbalanced, we could not use a ratio in the confusion matrix, hence we used the real number of samples and sum them over the 100 runs. The columns of the confusion table represent the classifier outputs while the rows represent the sample in the testing set. The first six columns/rows represent the known classes, the next three columns/rows represent the novel classes, and the 10'th column/row represents the outliers. Additionally, the last column represents the class-wise accuracy of the classifier, and the last row represents the class-wise reliability.

The evaluations of the proposed classifier are shown in Tables 8.7 – 8.12. Additionally, the evaluation of the ensemble consisting of all of the proposed classifiers is shown in Table 8.13.

Table 8.7: Confusion matrix of the incremental class ELM–EVT classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11797	0	0	0	0	0	0	0	0	106	99.11
C2	0	8293	0	0	0	0	0	0	0	59	99.29
C3	0	0	6921	0	0	0	1	0	0	145	97.93
C4	0	0	0	8851	0	0	0	0	0	133	98.52
C5	0	0	0	0	6715	0	0	0	0	74	98.91
C6	0	0	0	0	0	4849	0	0	0	69	98.60
C7	0	0	0	0	0	0	10908	0	0	47	99.57
C8	0	0	0	0	0	0	0	7312	0	102	98.62
C9	0	0	1	0	0	0	4	0	5173	240	95.48
Out	4	3	0	0	3	1	1	0	0	731	98.38
Rel	99.97	99.96	99.99	100.00	99.96	99.98	99.95	100.00	100.00	42.85	–

Table 8.8: Confusion matrix of the incremental class ELM–AE classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11991	0	1	1	0	0	8	0	0	129	98.85
C2	0	8518	0	0	0	0	3	0	0	88	98.94
C3	0	0	7293	0	0	0	1	0	0	81	98.89
C4	0	0	2	8380	0	0	5	0	0	180	97.82
C5	0	0	0	0	6471	0	1	0	0	257	96.17
C6	0	0	0	0	0	4753	0	0	0	306	93.95
C7	0	0	0	0	0	0	10821	1	0	79	99.27
C8	3	0	2	1	0	2	3	6795	0	256	96.22
C9	6	0	6	4	0	0	161	3	4922	266	91.69
Out	0	0	0	0	2	0	0	0	0	726	99.73
Rel	99.92	100.00	99.85	99.93	99.97	99.96	98.35	99.94	100.00	30.66	–

Table 8.9: Confusion matrix of the incremental class Mahalanobis distance classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11618	0	0	0	0	0	0	0	0	64	99.45
C2	0	8762	0	0	0	0	0	0	0	24	99.73
C3	0	0	7000	0	0	0	0	0	0	26	99.63
C4	0	0	0	7954	0	0	0	0	0	17	99.79
C5	0	0	0	0	6412	0	0	0	0	14	99.78
C6	0	0	0	0	0	4973	0	0	0	9	99.82
C7	0	0	0	0	0	0	11278	0	0	47	99.58
C8	0	0	0	0	0	0	0	7843	0	84	98.94
C9	0	0	0	0	0	0	113	0	5408	154	95.30
Out	0	0	0	0	0	0	0	0	0	783	100.00
Rel	100.00	100.00	100.00	100.00	100.00	100.00	99.01	100.00	100.00	64.08	–

Table 8.10: Confusion matrix of the incremental class Polynomial classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11168	0	0	0	0	0	0	0	0	469	95.97
C2	0	8613	0	0	0	0	0	0	0	336	96.25
C3	0	0	7192	0	0	0	0	0	0	208	97.19
C4	0	0	0	7529	0	0	0	0	0	279	96.43
C5	0	0	0	0	6022	0	0	0	0	484	92.56
C6	0	0	0	0	0	4533	0	0	0	610	88.14
C7	0	0	0	0	0	0	10884	0	0	101	99.08
C8	0	0	0	0	0	0	0	7541	0	209	97.30
C9	0	0	0	0	0	0	98	0	5124	400	91.14
Out	0	0	0	0	1	0	0	0	0	759	99.87
Rel	100.00	100.00	100.00	100.00	99.98	100.00	99.11	100.00	100.00	19.69	–

Table 8.11: Confusion matrix of the incremental class Parzen classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11658	0	2	15	1	0	9	8	0	133	98.58
C2	0	9229	0	2	0	0	0	1	0	82	99.09
C3	2	0	7317	0	1	0	0	1	0	79	98.88
C4	2	0	3	8002	0	0	1	0	0	112	98.55
C5	2	0	0	1	6298	0	0	0	0	131	97.92
C6	0	1	0	0	0	4706	0	2	1	191	96.02
C7	0	0	0	4	2	0	10192	7	0	258	97.41
C8	0	3	0	1	5	0	0	7441	0	508	93.50
C9	0	1	91	177	101	0	261	55	3981	719	73.91
Out	1	1	0	0	0	2	1	0	0	737	99.33
Rel	99.94	99.94	98.70	97.56	98.28	99.96	97.40	99.02	99.97	24.98	–

Table 8.12: Confusion matrix of the incremental class SVM classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11830	0	0	0	0	0	1	0	0	36	99.69
C2	0	8277	0	0	0	0	2	2	0	120	98.52
C3	0	0	7429	0	0	0	27	6	1	145	97.65
C4	0	0	0	8149	0	0	10	0	0	140	98.19
C5	0	0	0	0	6223	0	0	0	0	223	96.54
C6	0	0	0	0	0	4795	3	1	0	335	93.40
C7	0	0	0	0	0	0	11206	0	0	57	99.49
C8	0	0	0	0	0	0	2	7337	0	135	98.17
C9	0	0	0	0	0	0	89	49	4923	247	92.75
Out	1	0	0	0	0	0	4	0	0	746	99.33
Rel	99.99	100.00	100.00	100.00	100.00	100.00	98.78	99.22	99.98	34.16	–

Table 8.13: Confusion matrix of the incremental class ensemble classifier.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	Out	*Acc*
C1	11657	0	0	0	0	0	0	0	0	30	99.74
C2	0	8240	0	0	0	0	0	0	0	22	99.73
C3	0	0	7294	0	0	0	0	0	0	19	99.74
C4	0	0	1	9128	0	0	1	0	0	36	99.59
C5	0	0	0	0	6341	0	0	0	0	53	99.17
C6	0	0	0	0	0	5064	0	0	0	81	98.43
C7	0	0	0	0	0	0	11233	0	0	66	99.42
C8	0	0	0	0	2	0	2	7162	0	152	97.87
C9	0	0	0	0	0	0	68	49	4813	286	92.27
Out	0	0	0	0	0	0	1	0	0	744	99.87
Rel	100.00	100.00	99.99	100.00	99.97	100.00	99.36	99.32	100.00	49.97	–

8.3 Mahalanobis Distance and Polynomial Classifiers Ensemble

Another example of the ensemble is implemented here by using just two classifiers, which are the Mahalanobis distance classifier and the polynomial classifier. Since it is difficult to find a benchmark classifier that has the capability of incremental learning, novelty detection, and construction of new classes, we performed two separate experimental evaluations. In the first evaluation, the ability of our proposed algorithm of constructing a new class is compared with a fully supervised classifier. The second experiment evaluates the ability of the classifier to detect all outliers and compares it with the SVDD classifier. In all of these experiments, the second set of the gesture data, which are non-linearly separable and their classes are strongly overlapping. Section 4.1.1 is used to evaluate the classifier.

8.3.1 Evaluation of the new class construction

In this experiment, an evaluation of the performance of the developed algorithm is performed regarding two scenarios. The first scenario corresponds to the fully supervised approach, which serves as a reference, and in the second scenario, we used our developed algorithm. The available data were divided into three sets: the initial training set, the learning data set for which class labels are generated by our algorithm, and the test set. Each set comprises one-third of the overall data set.

The training scheme is repeated in 9 sets of 100 runs each, where for the kth set all samples of class c are excluded from the initial training set of our algorithm while the dataset is kept complete for the supervised approach. The excluded class is considered the novel class. In each run, the data are divided into initial, learning and test data sets in a different random manner. Initially, both classifiers are trained as fully supervised

Table 8.14: Results of the proposed semi-supervised learning algorithm in comparison to the results of the fully supervised approach making use of all available manually set labels. The average values were computed over 100 training runs each; the error intervals correspond to plus/minus one standard deviation. Where the p_u represents the purity of the newly constructed class and the *size ratio* represents the ratio of the newly constructed class size to the corresponding original class size.

Excluded class	p_u [%]		*size ratio*	
	semi-sup.	supervised	semi-supervised	supervised
class 1	97.6 ± 2.2	100.0 ± 0.0	99.7 ± 0.7	99.9 ± 0.3
class 2	89.0 ± 8.4	87.3 ± 3.7	49.9 ± 14.5	96.3 ± 1.4
class 3	93.1 ± 3.9	97.6 ± 2.1	93.1 ± 1.8	93.5 ± 2.3
class 4	95.4 ± 3.6	94.1 ± 2.0	70.2 ± 6.1	94.2 ± 4.1
class 5	99.9 ± 0.3	99.6 ± 0.5	96.0 ± 2.2	99.2 ± 0.6
class 6	68.6 ± 11.1	92.3 ± 3.6	21.0 ± 6.4	97.1 ± 1.5
class 7	98.4 ± 2.4	100.0 ± 0.0	99.3 ± 1.5	100.0 ± 0.0
class 8	99.1 ± 1.6	100 ± 0.0	99.9 ± 0.5	100.0 ± 0.2
class 9	92.4 ± 3.1	95.4 ± 1.7	85.3 ± 4.9	85.0 ± 4.4

Table 8.15: The overall recognition rate of the proposed semi-supervised learning algorithm compared to the recognition rate of fully supervised learning. The average values were computed over all 900 training runs; the error intervals correspond to plus/minus one standard deviation.

Overall recognition (semi-supervised)	Overall recognition (supervised)	Novel class recognition (semi-supervised)	Novel class recognition (semi-supervised)
94.5% ± 3.0	96.1% ± 0.7	79.4% ± 26.69	96.1% ± 5.11

with their own initial training set. Labels are then generated for the learning (unlabelled) data set, and the corresponding samples are included in the training set together with the labels estimated by the classifier. The fully supervised classifier is trained on all classes. Hence, it can classify all samples in the learning set. In contrast, there are some samples in the learning set belonging to the unseen (excluded) class of our algorithm. However, it will classify them as "unknown", and it will construct a new class containing some of these unknown samples which fulfil the conditions of the new class, and it rejects the other novel gestures by assigning them as random movements (outliers).

Let N_c be the total number of samples in the learning set belonging to the excluded class c and let the newly constructed class be $\hat{c}$. Additionally, let ${N_{\hat{c}}}^{\text{total}}$ and ${N_{\hat{c}}}^{\text{true}}$ be the total number of samples in the newly constructed class $\hat{c}$ and the samples that belong to both classes c and $\hat{c}$, respectively. The "purity" (p_u) of the newly constructed class is measured at the first stage of the algorithm, $p_u = {N_{\hat{c}}}^{\text{true}}/{N_{\hat{c}}}^{\text{total}}$ and the size ratio of the newly constructed class to the total size of class c (${N_{\hat{c}}}^{\text{total}}/N_c$) are shown in Table 8.14. In the case of the fully supervised classifier, the purity parameter of class c is the ratio between the correct classifier assignments to that class and its total number of samples.

The size ratio of class c is computed by dividing the number of samples assigned to that class by N_c.

The obtained purity values are reasonably high (between 89% and 99.9%) for all novel classes except class 6. That class is also characterised by a low size ratio of only 21%, while for all other classes except class 2 the size ratio exceeds 70% and even reaches values larger than 99% for classes 1, 7 and 8. The purity values obtained by supervised learning, which is trained using samples of the novel class and is thus able to recognise the novel class, correspond to within a few percent to those obtained by the proposed semi-supervised method except for class 6. Regarding the size ratio, the comparison between the semi-supervised and the fully supervised learning results is quite encouraging since the recognition rate of supervised learning significantly exceeds that of the semi-supervised approach only for the apparently "difficult" classes 2, 4 and 6.

For the same 900 training runs of our algorithm, Table 8.15 shows the recognition rate achieved by our semi-supervised algorithm in comparison to the recognition rate of the same classifier trained in a fully supervised manner based on the same training and test data, where the average difference is only about 2.5% in favour of the fully supervised scenario. A larger difference of about 15% is observed for the recognition rate of the novel class.

This result is not unexpected since if a high purity of the novel class is desired using the employed Mahalanobis distance criterion, a considerable fraction of the possible novel samples may be rejected as random movements (outliers). If it is desired to include more gestures into the new class, it is necessary to increase the corresponding threshold, which in turn will lead to a decreased purity. All in all, it depends on the application scenario if a large number of assigned novel samples or a high purity of the newly constructed class is desired.

8.3.2 Evaluation of the outlier detection

The dataset has been divided in the same manner of the first experiment. In addition, a small number of outliers has been added to the test data. The outlier samples don't belong to any of gesture classes and it has been created using the function "gendatout" from the toolbox (ddtools) (Tax, 2015). The proposed classifier and the SVDD classifier are trained on the training data, and then they classify the test data. The proposed classifier assigns the test sample to one of three cases, which are: labelling it with the label of one of the known classes, indicate it as an outlier, or indicate it as belonging to a new class. The SVDD classifier assigns it either as one of the known classes or as an outlier. Hence, the result is computed as the accuracy regarding the known classes and the accuracy of detecting unknown classes and outliers. These two measures are computed for both classifiers.

Another two measures are computed for the proposed classifier only, which are the accuracy of the newly constructed class and the accuracy of the rejected samples. The

Table 8.16: Accuracies of the proposed semi-supervised learning algorithm in comparison to the accuracies of the SVDD classifier. The average of the accuracy values was computed over 100 training runs each; the error intervals correspond to plus/minus one standard deviation.

Excluded class	Known Classes		Total Unknown Samples		Newly Con-Created Class	Outliers
	semi-sup.	SVDD	semi-sup.	SVDD	semi-sup.	semi-sup.
Class 1	93.1 ± 1.1	36.8 ± 19.9	99.6 ± 0.6	72.6 ± 41.0	100.0 ± 0.0	95.0 ± 7.1
Class 2	94.1 ± 1.1	36.7 ± 20.8	54.6 ± 11.6	55.6 ± 33.6	50.5 ± 12.7	96.4 ± 5.3
Class 3	93.8 ± 0.9	34.7 ± 23.2	97.8 ± 1.2	61.1 ± 43.1	96.1 ± 1.6	97.1 ± 5.2
Class 4	93.7 ± 1.0	41.4 ± 19.4	77.6 ± 4.5	35.5 ± 18.2	76.1 ± 4.9	92.7 ± 8.6
Class 5	93.0 ± 1.0	47.0 ± 2.3	95.8 ± 2.3	59.8 ± 6.6	95.7 ± 2.4	97.4 ± 4.6
Class 6	94.0 ± 1.0	41.6 ± 17.5	27.7 ± 5.6	48.1 ± 23.0	21.2 ± 6.1	96.0 ± 6.3
Class 7	92.7 ± 1.2	35.3 ± 18.6	99.2 ± 1.1	63.1 ± 34.8	99.5 ± 1.1	95.9 ± 6.2
Class 8	92.8 ± 1.1	34.3 ± 19.3	99.7 ± 0.5	73.4 ± 43.8	100.0 ± 0.0	96.6 ± 6.3
Class 9	93.1 ± 1.0	37.6 ± 21.9	92.2 ± 3.2	64.0 ± 39.6	92.0 ± 3.4	91.9 ± 8.1

outliers which are added to the test data using the function "gendatout" are considered as a reference to the outliers (rejected samples) of the proposed algorithm in computing the outlier detection accuracy. Consequently, the test samples which belong to the excluded class in the training data are considered as references to the newly constructed class.

The experiment was run 900 times. The summary of the results is shown in Table 8.16. The advantage of the proposed algorithm over the SVDD algorithm is clearly apparent from this table. Again, both algorithms have difficulties in detecting the outliers of classes 2 and 6 as they strongly overlap with the other classes.

9

Summary and conclusion

The machine should be able to react to the human gestures to achieve natural interaction between the human and the machine in most of HCI applications. Human gestures are culture-based, and in principle, there is an unlimited number of possible gestures, which makes supervised learning problematic. Instead, the machine should be able to learn and respond to the gestures that are continuously streamed. To learn from the data stream, various problems may arise, e.g. infinite length, concept-drift and concept evolution. Semi-supervised learning provides a solution using the unlabelled data, but we still need to solve the other problems. Incremental learning is significant to develop the machine intelligence and bring it close to real interaction. It is important from two points of view. First, it should be able to acquire an experience from the environments incrementally and continuously. Second, it should be able to respond to new gestures over time. Two different types of the incremental learning are available, which are: the classifier updates its information of the already trained on classes (standard incremental learning), while the second type is to update its class structure by adding new classes (incremental class learning). The classifier should be able to distinguish if the sample belongs to the known classes or not. Additionally, to add the new samples autonomously in the standard incremental learning, the classifier should recognise if the assignment of this sample is trustworthy or not. Furthermore, it should discriminate between random movements and the meaningful movements. Moreover, it must be able to discriminate among the samples that belong to different novel classes.

Due to the varieties of the gesture types, we proposed in this work several methods to fit as many as possible types of gestures. The proposed methods are based on different techniques: the incremental Parzen window and the incremental support vector machine are non-parametric methods, the incremental neural network (extreme learning machine) and the incremental polynomial classifiers are non-linear classifiers, and the incremental Mahalanobis distance classifier and the kernel distance are metric learning algorithms. The novelty detection of all of these classifiers is implemented by using extreme value

theory (EVT). The EVT is more accurate than the conventional methods, and it does not need additional labelled data or a time-consuming cross-validation to estimate the novelty threshold. Additionally, the threshold in EVT has a direct statistical interpretation and does not depend on the distribution of the classes whereas the conventional thresholds depend on the distribution of the classes. Another type of novelty detection, which does not require additional labelled data, is used for some classifiers in combination with the EVT. Both types of incremental learning are incorporated with the proposed classifiers. We proposed a paradigm to provide a self-adaptive structure semi-supervised learning, which is used as a wrapper for any of the proposed classifiers or even an ensemble of them. In addition to the standard incremental learning and the incremental class learning, it integrates the unsupervised learning (mean-shift clustering algorithm) to discriminate the different novel classes.

The standard incremental learning and the novelty detection performance of each classifier are evaluated by training the classifier initially on a training set which has 8 classes only and applying it to the test set, which has 9 classes. The data stream is emulated by providing the learning set data in buckets, where the learning set also has 9 classes. The experiment was repeated 9 times by considering a different class as a novel in each run. Since the data partitioning depends on a random permutation, each individual configuration was run 100 times. Thus, the total number of runs to evaluate one classifier is 900 times. The classifier outputs are compared with the outputs of the same classifier type from the state-of-art classifiers. The results show that the proposed classifiers have superior properties.

The incremental class learning was evaluated by using two versions of the same classifier; one of them was trained using the supervised method while the other one was updated by using the incremental class learning. The difference between the outputs of the two classifiers is as follows. The results show that the ICL needs less time for the processing while it maintains similar outputs.

The classifier of self-adaptive structure semi-supervised learning is trained on only 6 classes, while the learning and testing sets contain 9 classes plus some random movements; we added them as outliers. The system succeeds to predict all the missing classes and updates itself on them while rejecting all the outliers. The confusion matrix is computed for each classifier and also for the ensemble that consists of all the classifiers is evaluated. The average prediction accuracy is more than 99 %, and the reliability (the purity of the predicted classes) is also more than 99 %. Additionally, another experiment is implemented by using an ensemble of the polynomial classifier and the Mahalanobis classifier on strongly overlapping data. The outputs of the ensemble classifier are compared with a second version of the classifier that is trained in a supervised way. The results show that the accuracy of the proposed methods is on average just 2.5 % less than that of the fully supervised learning, although it shows a high variability depending on which class is discarded.

Future works The system is designed to work with one-handed gestures, and we expect that it could be adapted to two-handed gestures. Recently, community recognition has become a hot research topic; the proposed classifiers may be possible to be updated to work on community recognition. Other types of sensors can be used such as inertial measurement unit or may include current smartwatches. Although we used different kinds of data to evaluate the system, it may be helpful to evaluate it on further types of data. Furthermore, we expect that adding new types of classifiers, such as Hidden Markov Models, will allow for more specific selection of appropriate classifiers, given the regarded type of data.

Humans use voice and gestures in their communications. Although the system is designed to work on gestures only, it might become more robust and similar to human interaction if voice recognition or voice source recognition is integrated into it.

This thesis aims to improve the intelligence of machines and develop proper methodologies and efficient systems to produce an artificial intelligence closer to what humans have. The proposed methods are a step forward towards this aim and may build a foundation for future improvements.

Bibliography

Abid, M. R. *Visual Recognition of a Dynamic Arm Gesture Language for Human-Robot and Inter-Robot Communication*. PhD thesis, University of Ottawa, 2015.

Adams, J. B. Interpretation of visible and near-infrared diffuse reflectance spectra of pyroxenes and other rock-forming minerals. In Karr, C., editor, *Infrared and Raman Spectroscopy of Lunar and Terrestrial Minerals*, chapter 4, pages 91–116. Academic Press, New York, San Francisco, London, 1975.

Akima, H. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the Association for Computing Machinery*, 17(4):589–602, 1970.

Al-Behadili, H., Grumpe, A., and Wöhler, C. Semi-supervised learning of emblematic gestures. *At-Automatisierungstechnik*, 62(10):732–739, 2014.

Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Extreme learning machine based novelty detection for incremental semi-supervised learning. In *Third IEEE International Conference on Image Information Processing (ICIIP)*, pages 230–235, Dec 2015a. doi: 10.1109/ICIIP.2015.7414771.

Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Incremental class learning and novel class detection of gestures using ensemble. In *Workshop New Challenges in Neural Computation 2015*, pages 122–132, 2015b.

Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Non-linear distance based large scale data classifications. In *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 613–617, Dec 2015c. doi: 10.1109/PIC.2015.7489921.

Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Incremental learning and novelty detection of gestures using extreme value theory. In *IEEE International Conference on*

Computer Graphics, Vision and Information Security (CGVIS), pages 169–174, Nov 2015d. doi: 10.1109/CGVIS.2015.7449915.

Al-Behadili, H., Grumpe, A., Dopp, C., and Wöhler, C. Semi-supervised learning using incremental polynomial classifier and extreme value theory. In *3rd IEEE International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 332–337, 2015e. doi: 10.1109/AIMS.2015.60.

Al-Behadili, H., Grumpe, A., and Wöhler, C. Incremental learning and novelty detection of gestures in a multi-class system. In *3rd IEEE International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 304–309, 2015f. doi: 10.1109/AIMS.2015.55.

Al-Behadili, H., Grumpe, A., Migdadi, L., and Wöhler, C. Semi-supervised learning using incremental support vector machine and extreme value theory in gesture data. In *18th IEEE International Conference on Computer Modelling and Simulation (UKSim-AMSS)*, pages 184–189, 2016a. doi: 10.1109/UKSim.2016.5.

Al-Behadili, H., Grumpe, A., Migdadi, L., and Wöhler, C. Incremental parzen window classifier for a multi-class system. *International Journal of Simulation–Systems, Science & Technology*, 17(34), 2016b.

Al-Behadili, H., Grumpe, A., and Wöhler, C. Confidence band and extreme value theory based outlier detection for semi-supervised learning of incremental polynomial classifier. *International Journal of Simulation Systems, Science & Technology*, 17(34), 2016c.

Al-Behadili, H., Grumpe, A., and Wöhler, C. Neural network based novelty detection for incremental semi-supervised learning in multi-class gesture recognition. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP,*, pages 287–294, 2016d. ISBN 978-989-758-175-5. doi: 10.5220/0005674202870294.

Al-Behadili, H., Grumpe, A., and Wöhler, C. Non-linear distance-based semi-supervised multi-class gesture recognition. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP,*, pages 280–286, 2016e. ISBN 978-989-758-175-5. doi: 10.5220/0005674102800286.

Ali, S. and Shah, M. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 32(2):288–303, 2010.

Alon, J., Athitsos, V., and Sclaroff, S. Accurate and efficient gesture spotting via pruning and subgesture reasoning. In *International Workshop on Human-Computer Interaction*, pages 189–198. Springer, 2005.

Alon, J., Athitsos, V., Yuan, Q., and Sclaroff, S. A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 31(9):1685–1699, 2009.

Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

Amos, E. Wikimedia commons, Microsoft Kinect, 2017a. URL `https://upload.wikimedia.org/wikipedia/commons/6/67/Xbox-360-Kinect-Standalone.png`. Accessed: 2017-01-11.

Amos, E. Wikimedia commons, Head Mounted Display, 2017b. URL `https://commons.wikimedia.org/wiki/File%3AVictorMaxx-StuntMaster.jpg`. Accessed: 2017-01-11.

Anand, S., Mittal, S., Tuzel, O., and Meer, P. Semi-supervised kernel mean shift clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1201–1215, 2014.

Balcan, M.-F., Blum, A., and Yang, K. Co-training and expansion: Towards bridging theory and practice. In *Advances in neural information processing systems*, pages 89–96, 2004.

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Baraldi, S., Del Bimbo, A., and Landucci, L. Natural interaction on tabletops. *Multimedia Tools and Applications*, 38(3):385–405, 2008.

Barber, D. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2013.

Baudat, G. and Anouar, F. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10):2385–2404, 2000.

Bellman, R. and Kalaba, R. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

Bennett, K., Demiriz, A., et al. Semi-supervised support vector machines. *Advances in Neural Information processing systems*, pages 368–374, 1999.

Berg, B. A. and Neuhaus, T. Multicanonical ensemble: A new approach to simulate first-order phase transitions. *Physical Review Letters*, 68(1):9, 1992.

Berry, G. A. *Small-wall: A Multimodal Human Computer Intelligent Interaction Test Bed with Applications*. University of Illinois at Urbana-Champaign, 1998.

Bhattachayya, A. On a measure of divergence between two statistical population defined by their population distributions. *Bulletin Calcutta Mathematical Society*, 35:99–109, 1943.

Bhuyan, M., Bora, P., and Ghosh, D. Trajectory guided recognition of hand gestures having only global motions. *International Journal of Computer Science, Fall*, 2008.

Bhuyan, M., Kumar, D. A., MacDorman, K. F., and Iwahori, Y. A novel set of features for continuous hand gesture recognition. *Journal on Multimodal User Interfaces*, 8(4): 333–343, 2014.

Birdal, A. and Hassanpour, R. Region based hand gesture recognition. In *16th International conference in central Europe on computer graphics, visualization and computer vision*, pages 1–7. Václav Skala-UNION Agency, 2008.

Bishop, C. M. Novelty detection and neural network validation. In *IEE Proceedings: Vision, Image and Signal Processing*, volume 141, pages 217–222. IET, 1994.

Bishop, C. M. et al. *Neural networks for pattern recognition*. Clarendon press Oxford, 1995.

Bishop, C. M. et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Bobick, A. and Davis, J. Real-time recognition of activity using temporal templates. In *Proceedings 3rd IEEE Workshop on Applications of Computer Vision, 1996. WACV'96.*, pages 39–42. IEEE, 1996.

Bodor, R., Morlok, R., and Papanikolopoulos, N. Dual-camera system for multi-level activity recognition. In *IROS*, pages 643–648. Citeseer, 2004.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Brand, M., Oliver, N., and Pentland, A. Coupled hidden markov models for complex action recognition. In *IEEE computer society conference on computer vision and pattern recognition, 1997.*, pages 994–999. IEEE, 1997.

Bretzner, L., Laptev, I., and Lindeberg, T. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 2002.*, pages 423–428. IEEE, 2002.

Bruzzone, L., Cossu, R., and Vernazza, G. Detection of land-cover transitions by combining multidate classifiers. *Pattern Recognition Letters*, 25(13):1491–1500, 2004.

Burns, R. G., Abu-Eid, R. M., and Huggins, F. E. Crystal field spectra of lunar pyroxenes. In *Lunar and Planetary Science Conference Proceedings*, volume 3, pages 533–543, 1972.

Cabibihan, J.-J., So, W.-C., and Pramanik, S. Human-recognizable robotic gestures. *IEEE Transactions on Autonomous Mental Development*, 4(4):305–314, 2012.

Cambria, E., Huang, G.-B., Kasun, L. L. C., Zhou, H., Vong, C. M., Lin, J., Yin, J., Cai, Z., Liu, Q., Li, K., et al. Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, 28(6):30–59, 2013.

Cauwenberghs, G. and Poggio, T. Incremental and Decremental Support Vector Machine Learning. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 388–394, Cambridge, MA, USA, 2000. MIT Press.

Chang, C.-K. and Huang, J. Video surveillance for hazardous conditions using sensor networks. In *IEEE International Conference on Networking, Sensing and Control, 2004.*, volume 2, pages 1008–1013. IEEE, 2004.

Chang, J.-Y., Shyu, J.-J., and Cho, C.-W. Fuzzy rule inference based human activity recognition. In *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*, pages 211–215. IEEE, 2009a.

Chang, K.-Y., Liu, T.-L., and Lai, S.-H. Learning partially-observed hidden conditional random fields for facial expression recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 533–540. IEEE, 2009b.

Chapelle, O., Chi, M., and Zien, A. A continuation method for semi-supervised svms. In *Proceedings of the 23rd international conference on Machine learning*, pages 185–192. ACM, 2006a.

Chapelle, O., Schölkopf, B., Zien, A., et al. *Semi-supervised learning*. MIT press Cambridge, 2006b.

Chapelle, O., Sindhwani, V., and Keerthi, S. S. Branch and bound for semi-supervised support vector machines. In *Advances in neural information processing systems*, pages 217–224, 2006c.

Chapelle, O., Sindhwani, V., and Keerthi, S. S. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9: 203–233, 2008.

Chawla, N. V. and Karakoulas, G. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *J. Artif. Int. Res.*, 23:331–366, 2005.

Chen, F.-S., Fu, C.-M., and Huang, C.-L. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and vision computing*, 21(8):745–758, 2003.

Chen, Q., Cordea, M. D., Petriu, E. M., Varkonyi-Koczy, A. R., and Whalen, T. E. Human computer interaction for smart environment applications using hand gestures and facial expressions. *International Journal of Advanced Media and Communication*, 3(1-2):95–109, 2009.

Chen, S. and He, H. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2011.

Chen, S., Wang, H., Zhou, S., and Yu, P. S. Stop chasing trends: Discovering high order models in evolving data. In *Proc. ICDE*, pages 923–932. IEEE, 2008.

Chen, Y., Liu, M., Liu, J., Shen, Z., and Pan, W. Slideshow: Gesture-aware ppt presentation. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–4. IEEE, 2011.

Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Choi, C., Ahn, J.-H., and Byun, H. Visual recognition of aircraft marshalling signals using gesture phase analysis. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 853–858. IEEE, 2008.

Chow, C. K. On optimum recognition and error and reject tradeoff. *IEEE Transactions on Information Theory (IT)*, 16:41–46, 1970.

Cipolla, R. and Hollinghurst, N. J. Human-robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171–178, 1996.

Clifton, D. A. *Novelty detection with extreme value theory in jet engine vibration data*. PhD thesis, University of Oxford, 2009.

Clifton, D. A., Bannister, P. R., and Tarassenko, L. Learning shape for jet engine novelty detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3973 LNCS:828–835, 2006.

Clifton, D. A., Clifton, L. A., Bannister, P. R., and Tarassenko, L. Automated novelty detection in industrial systems. In *Advances of Computational Intelligence in Industrial Systems*, pages 269–296. Springer, 2008.

Clifton, D. A., Hugueny, S., and Tarassenko, L. A comparison of approaches to multivariate extreme value theory for novelty detection. In *15th Workshop on Statistical Signal Processing, 2009 IEEE/SP. SSP'09.*, pages 13–16, 2009.

Clifton, D. A., Hugueny, S., and Tarassenko, L. Novelty detection with multivariate extreme value statistics. *Journal of signal processing systems*, 65(3):371–389, 2011.

Clifton, L. A. *Multi-Channel Novelty Detection and Classifier Combination*. PhD thesis, University of Manchester, Manchester, 2007.

Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., et al. A system for video surveillance and monitoring. Technical report, Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, 2000.

Comaniciu, D. and Meer, P. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

Cordella, L. P., Stefano, C. D., Tortorella, F., and Vento, M. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147, 1995.

Corradini, A. Real-time gesture recognition by means of hybrid recognizers. In *International Gesture Workshop*, pages 34–47. Springer, 2001.

Cozman, F. G., Cohen, I., Cirelo, M. C., et al. Semi-supervised learning of mixture models. In *ICML*, pages 99–106, 2003.

Crowley, J., Berard, F., Coutaz, J., et al. Finger tracking as an input device for augmented reality. In *International Workshop on Gesture and Face Recognition*, pages 195–200, 1995.

Culp, M. and Michailidis, G. An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3):545–571, 2008.

Cutler, R. and Turk, M. View-based interpretation of real-time optical flow for gesture recognition. In *Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998*, pages 416–421, 1998.

CyberGlove Systems LLC. CyberGlove II, 2017. URL https://static1.squarespace.com/static/559c381ee4b0ff7423b6b6a4/55fa5c5de4b01cd1ed50f232/55fa5cb0e4b07da7e5fef7cb/1442471090106/01.png?format=1500w. Accessed: 2017-01-11.

Dadgostar, F., Sarrafzadeh, A., and Gholamhosseini, H. A component-based architecture for vision-based gesture recognition. In *Image and Vision Computing New Zealand Conference, University of Otago, Dunedin*, pages 28–29, 2005.

Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

Darrell, T. J., Essa, I. A., and Pentland, A. P. Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1236–1242, 1996.

Dasarathy, B. V. and Sheela, B. V. A composite classifier system design: concepts and methodology. *Proceedings of the IEEE*, 67(5):708–713, 1979.

Dasarathy, B. *Nearest neighbor (NN) norms: nn pattern classification techniques*. IEEE Computer Society Press tutorial. IEEE Computer Society Press, 1991. ISBN 9780818659300.

Dasgupta, D. and Majumdar, N. S. Anomaly detection in multidimensional data using negative selection algorithm. In *Proceedings of the World on Congress on Computational Intelligence*, volume 2, pages 1039–1044. IEEE, 2002.

Dautenhahn, K. Socially intelligent robots: dimensions of human–robot interaction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):679–704, 2007.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

Derpanis, K. G., Wildes, R. P., and Tsotsos, J. K. Hand gesture recognition within a linguistics-based framework. In *European Conference on Computer Vision*, pages 282–296. Springer, 2004.

Diaz, I. and Hollmén, J. Residual generation and visualization for understanding novel process conditions. In *International Joint Conference on Neural Networks, 2002. IJCNN'02.*, volume 3, pages 2070–2075. IEEE, 2002.

Didaci, L. and Roli, F. *Using Co-training and Self-training in Semi-supervised Multiple Classifier Systems*, pages 522–530. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

Diehl, C. P. and Cauwenberghs, G. Svm incremental learning, adaptation and optimization. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 2685–2690. IEEE, 2003.

Ditzler, G. and Polikar, R. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, 2013.

Eisenstein, J., Barzilay, R., and Davis, R. Discourse topic and gestural form. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 836–841, 2008.

Elmezain, M., Al-Hamadi, A., Appenrodt, J., and Michaelis, B. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *19th International Conference on Pattern Recognition, 2008. ICPR 2008.*, pages 1–4. IEEE, 2008.

Elmezain, M., Al-Hamadi, A., Rashid, O., and Michaelis, B. Posture and Gesture Recognition for Human-Computer Interaction. In *Advanced Technologies*, pages 415–440. InTech, 2009.

Elmezain, M. O. S. M. *Hand gesture spotting and recognition using HMMs and CRFs in color image sequences*. PhD thesis, PhD thesis, Computer Science Dept. Magdeburg Univ, 2010.

Emamian, V., Kaveh, M., and Tewfik, A. H. Robust clustering of acoustic emission signals using the kohonen network. In *Proc. ICASSP'00*, volume 6, pages 3891–3894. IEEE, 2000.

Embrechts, P., Klüppelberg, C., and Mikosch, T. *Modelling extremal events: for insurance and finance*, volume 33. Springer Science & Business Media, 2013.

Esbensen, K. H. and Geladi, P. Principles of Proper Validation: use and abuse of re-sampling for validation. *Journal of Chemometrics*, 24:168–187, 2010.

Faria, E. R., Gama, J., and Carvalho, A. Novelty detection algorithm for data streams multi-class problems. In *Proceedings of the 28th annual ACM symposium on applied computing*, pages 795–800. ACM, 2013.

Farid, D. M., Zhang, L., Hossain, A., Rahman, C. M., Strachan, R., Sexton, G., and Dahal, K. An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications*, 40(15):5895–5906, 2013.

Fels, S. S. *Glove-talkii: Mapping hand gestures to speech using neural networks an approach to building adaptive interfaces*. PhD thesis, Citeseer, 1994.

Fels, S. S., Pritchard, B., and Lenters, A. Fortouch: A wearable digital ventriloquized actor. In *NIME*, pages 274–275, 2009.

Feng, G., Huang, G.-B., Lin, Q., and Gay, R. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20(8):1352–1357, 2009.

Feng, Y., Liu, Z., and Li, B. Gestureflow: streaming gestures to an audience. In *2011 Proceedings IEEE INFOCOM*, pages 748–756. IEEE, 2011.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

Fisher, R. A. and Tippett, L. H. C. Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 24, pages 180–190. Cambridge Univ Press, 1928.

Fong, T., Nourbakhsh, I., and Dautenhahn, K. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3):143–166, 2003.

Fothergill, S., Mentis, H., Kohli, P., and Nowozin, S. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1737–1746. ACM, 2012.

Freund, Y., Schapire, R. E., et al. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*, volume 96, pages 148–156, 1996.

Fu, Z., Robles-Kelly, A., Caelli, T., and Tan, R. T. On automatic absorption detection for imaging spectroscopy: A comparative study. *IEEE Transactions on Geoscience and Remote Sensing*, 45(11):3827–3844, 2007.

Fujino, A., Ueda, N., and Saito, K. Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):424–437, 2008.

Fukumoto, M., Suenaga, Y., and Mase, K. "finger-pointer": Pointing interface by image processing. *Computers & graphics*, 18(5):633–642, 1994.

Fukunaga, K. and Hostetler, L. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1): 32–40, 1975.

Fung, G. M. and Mangasarian, O. L. Multicategory Proximal Support Vector Machine Classifiers. *Machine Learning*, 59(1):77–97, 2005.

Gama, J. *Knowledge discovery from data streams*. CRC Press, 2010.

Gao, W., Fang, G., Zhao, D., and Chen, Y. Transition movement models for large vocabulary continuous sign language recognition. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 553–558. IEEE, 2004.

GarcíA-RodríGuez, J., Angelopoulou, A., García-Chamizo, J. M., Psarrou, A., Escolano, S. O., and GiméNez, V. M. Autonomous growing neural gas for applications with time constraint: optimal parameter estimation. *Neural Networks*, 32:196–208, 2012.

Gatignon, H. *Statistical Analysis of Management Data*. Springer-Verlag, New York, 2010.

Geng, X. and Smith-Miles, K. *Incremental Learning*, pages 731–735. Springer US, 2009.

Ghobadi, S. E., Loepprich, O. E., Ahmadov, F., Bernshausen, J., Hartmann, K., and Loffeld, O. Real time hand based robot control using multimodal images. *IAENG International Journal of Computer Science*, 35(4):500–505, 2008.

Gillian, N. E. *Gesture Recognition for Musician Computer Interaction*. PhD thesis, Queen's University Belfast, 2011.

Goldberg, A. B. and Zhu, X. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 45–52. Association for Computational Linguistics, 2006.

Goldberg, A. B. *New directions in semi-supervised learning*. PhD thesis, University of Wisconsin–Madison, 2010.

Graetzel, C., Fong, T., Grange, S., and Baur, C. A non-contact mouse for surgeon-computer interaction. *Technology and Health Care*, 12(3):245–257, 2004.

Grumpe, A. and Wöhler, C. Recovery of elevation from estimated gradient fields constrained by digital elevation maps of lower lateral resolution. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94:37–54, 2014a. doi: 10.1016/j.isprsjprs.2014.04.011.

Grumpe, A. and Wöhler, C. Automatic segmentation of petrographic geologic units based on elemental abundance maps. In *Proc. European Lunar Symposium*, pages 100–101, London, UK, 2014b.

Grumpe, A., Zirin, V., and Wöhler, C. A normalisation framework for (hyper-)spectral imagery. *Planetary and Speca Science*, 111:1–33, 2015. doi: 10.1016/j.pss.2014.10.013.

Guo, L., Hao, J.-h., and Liu, M. An incremental extreme learning machine for online sequential learning problems. *Neurocomputing*, 128:50–58, 2014.

Gutierrez, D. D. *Machine Learning and Data Science: An Introduction to Statistical Learning Methods with R.* Technics Publications, 2015.

Gwadera, R., Atallah, M. J., and Szpankowski, W. Markov models for identification of significant episodes. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 404–414, 2005.

Haffari, G. R. and Sarkar, A. Analysis of semi-supervised learning with the yarowsky algorithm. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 159–166. AUAI Press, 2007.

Haggett, S. J. *Towards a multipurpose neural network approach to novelty detection.* PhD thesis, University of Kent, United Kingdom, 2008.

Han, J., Shao, L., Xu, D., and Shotton, J. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics.*, 43:1318–1334, 2013.

Hand, C., Sexton, I., and Mullan, M. A linguistic approach to the recognition of hand gestures. In *Designing Future Interaction Conference*, 1994.

Hapke, B. Bidirectional reflectance spectroscopy: 3. Correction for macroscopic roughness. *Icarus*, 59(1):41–59, 1984.

Hapke, B. Bidirectional reflectance spectroscopy: 5. The Coherent Backscatter Opposition Effect and Anisotropic Scattering. *Icarus*, 157(2):523–534, 2002.

Harper, M. P. and Shriberg, E. Multimodal model integration for sentence unit detection. In *Proceedings of the 6th international conference on Multimodal interfaces.*, pages 121–128. ACM, 2004.

Harris, T. Neural network in machine health monitoring. *Professional Engineering*, 1993.

Hassan, S. M., Al-Sadek, A. F., and Hemayed, E. E. Rule-based approach for enhancing the motion trajectories in human activity recognition. In *10th International Conference on Intelligent Systems Design and Applications.*, pages 829–834. IEEE, 2010.

Hastie, T., Tibshirani, R., et al. Classification by pairwise coupling. *The annals of statistics*, 26(2):451–471, 1998.

Hawkins, S., He, H., Williams, G., and Baxter, R. Ohtlier Detection Using Replicator Neural Networks. In *Data Warehousing and Knowledge Discovery: 4th International Conference, DaWaK 2002 Aix-en-Provence, France, Septemer 4–6*, pages 170–180. Springer Berlin Heidelberg, 2002.

He, H. *Self-adaptive systems for machine intelligence*. John Wiley & Sons, 2011.

He, Z., Deng, S., Xu, X., and Huang, J. Z. A fast greedy algorithm for outlier mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 567–576. Springer, 2006.

Hein, M., Audibert, J.-Y., and Von Luxburg, U. Graph laplacians and their convergence on random neighborhood graphs. *J. Mach. Learn. Res.*, 2007.

Hertz, J., Krogh, A., and Palmer, R. G. *Introduction to the theory of neural computation*, volume 1. Basic Books, 1991.

Hiroshi, K., Makito, S., Kazuhiko, S., Ken-ichi, T., and Kazuo, K. Pattern recognition for video surveillance and physical security. Technical report, Technical Report 375, The Institute of Electronics, Information and Communication Engineers., 2006.

Hoste, L., De Rooms, B., and Signer, B. Declarative gesture spotting using inferred and refined control points. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 144–150, 2013.

Hua, X. and Ding, S. Incremental learning algorithm for support vector data description. *Journal of Software*, 6(7):1166–1173, 2011.

Huang, G.-B. and Chen, L. Convex incremental extreme learning machine. *Neurocomputing*, 70(16):3056–3062, 2007.

Huang, G.-B. and Chen, L. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71(16):3460–3468, 2008.

Huang, G.-B., Saratchandran, P., and Sundararajan, N. An efficient sequential learning algorithm for growing and pruning rbf (gap-rbf) networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(6):2284–2292, 2004.

Huang, G.-B., Chen, L., and Siew, C.-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006a.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006b.

Hugueny, S., Clifton, D. A., and Tarassenko, L. Novelty detection with multivariate extreme value theory, part ii: An analytical approach to unimodal estimation. In *Proc. MLSP*, pages 1–6. IEEE, 2009.

Jain, A. and Nikovski, D. Incremental exemplar learning schemes for classification on embedded devices. *Machine Learning*, 72(3):189–203, 2008.

Japkowicz, N., Myers, C., Gluck, M., et al. A novelty detection approach to classification. In *Proc. IJCAI*, pages 518–523, 1995.

Jeon, B. and Landgrebe, D. A. Fast parzen density estimation using clustering-based branch and bound. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):950–954, 1994.

Johnson, R. and Zhang, T. On the effectiveness of laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research*, 8(4), 2007.

Ju, S. X., Black, M. J., and Yacoob, Y. Cardboard people: A parameterized model of articulated image motion. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996.*, pages 38–44. IEEE, 1996.

Ju, S. X., Black, M. J., Minneman, S., and Kimber, D. Analysis of gesture and action in technical talks for video indexing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997.*, pages 595–601. IEEE, 1997.

Kaâniche, M. B. *Human gesture recognition*. PhD thesis, Citeseer, 2009.

Kahol, K., Tripathi, P., and Panchanathan, S. Automated gesture segmentation from dance sequences. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004.*, pages 883–888. IEEE, 2004.

Kang, H., Lee, C. W., and Jung, K. Recognition-based gesture spotting in video games. *Pattern Recognition Letters*, 25(15):1701–1714, 2004.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

Karami, A., Zanj, B., and Sarkaleh, A. K. Persian sign language (psl) recognition using wavelet transform and neural networks. *Expert Systems with Applications*, 38(3):2661–2667, 2011.

Kardaun, O. J. *Classical methods of statistics: with applications in fusion-oriented plasma physics*, volume 1. Springer Science & Business Media, 2005.

Karlen, M., Weston, J., Erkan, A., and Collobert, R. Large scale manifold transduction. In *Proceedings of the 25th international conference on Machine learning*, pages 448–455. ACM, 2008.

Kendall, W. S., Marin, J.-M., and Robert, C. P. Confidence bands for brownian motion and applications to monte carlo simulation. *Statistics and Computing*, 17(1):1–10, 2007.

Kendon, A. Gesticulation and speech: Two aspects of the process of utterance. *The relationship of verbal and nonverbal communication*, 25(1980):207–227, 1980.

Kendon, A. *Gesture: Visible Action as Utterance*. Cambridge University Press, 2004.

Kettebekov, S., Yeasin, M., and Sharma, R. Prosody based audiovisual coanalysis for coverbal gesture recognition. *IEEE transactions on multimedia*, 7(2):234–242, 2005.

Kevin, N. Y. Y., Ranganath, S., and Ghosh, D. Trajectory modeling in gesture recognition using cybergloves® and magnetic trackers. In *TENCON 2004, IEEE Region 10 Conference*, pages 571–574. IEEE, 2004.

Khoshelham, K. Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, page W12, 2011.

Kita, S., Van Gijn, I., and Van der Hulst, H. Movement phases in signs and co-speech gestures, and their transcription by human coders. In *International Gesture Workshop*, pages 23–35. Springer, 1997.

Kjeldsen, R. and Kender, J. Visual hand gesture recognition for window system control. In *International Workshop on Automatic Face and Gesture Recognition*, pages 184–188, 1995.

Ko, H. and Jacyna, G. M. Dynamical behavior of autoassociative memory performing novelty filtering for signal enhancement. *IEEE Transactions on Neural Networks*, 11 (5):1152–1161, 2000.

Koch, R. Dynamic 3-d scene analysis through synthesis feedback control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):556–568, 1993.

Kohonen, T. *Self-organisation and associative memory*. Springer-Verlag, 1988.

Kohonen, T. *Self-organizing maps*. Springer, 2001.

Kopp, S., Sowa, T., and Wachsmuth, I. Imitation games with an artificial agent: From mimicking to understanding shape-related iconic gestures. In *Gesture-based communication in human-computer interaction*, pages 436–447. Springer, 2003.

Kruskall, J. and Liberman, M. The symmetric time warping algorithm: From continuous to discrete. time warps, string edits and macromolecules, 1983.

Kuncheva, L. I. and Bezdek, J. C. Nearest prototype classification: clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 28(1):160–164, 1998.

Kung, S. Y. *Kernel Methods and Machine Learning*. Cambridge University Press, 2014.

Kuno, Y., Murashina, T., Shimada, N., and Shirai, Y. Intelligent wheelchair remotely controlled by interactive gestures. In *15th International Conference on Pattern Recognition, 2000.*, volume 4, pages 672–675. IEEE, 2000.

Labib, K. and Vemuri, R. Nsom: A real-time network-based intrusion detection system using self-organizing maps. *Networks and Security*, pages 1–6, 2002.

Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

Lan, Y., Soh, Y. C., and Huang, G.-B. Ensemble of online sequential extreme learning machine. *Neurocomputing*, 72(13):3391–3395, 2009.

Laptev, I. On space-time interest points. *International Journal of Computer Vision*, 64 (2-3):107–123, 2005.

Larose, D. T. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

Laskov, P., Gehl, C., Krüger, S., and Müller, K.-R. Incremental support vector learning: Analysis, implementation and applications. *The Journal of Machine Learning Research*, 7:1909–1936, 2006.

Lawrence, D. J., Feldman, W. C., Barraclough, B. L., Binder, A. B., Elphic, R. C., Maurice, S., and Thomsen, D. R. Global Elemental Maps of the Moon: The Lunar Prospector Gamma-Ray Spectrometer. *Science*, 281(5382):1484–1489, 1998.

Leap Motion Inc. Designing Leap Motion Controller, 2017. URL `https://www.leapmotion.com/product`. Accessed: 2017-01-11.

Lee, D.-H. and Hong, K.-S. Game interface using hand gesture recognition. In *Proc. ICCIT*, pages 1092–1097. IEEE, 2010.

Lee, H.-K. and Kim, J.-H. An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 21(10):961–973, 1999.

Lee, H.-j. and Roberts, S. J. On-line novelty detection using the kalman filter and extreme value theory. In *19th International Conference on Pattern Recognition, 2008. ICPR 2008.*, pages 1–4. IEEE, 2008.

Lee, S., Henderson, V., Hamilton, H., Starner, T., Brashear, H., and Hamilton, S. A gesture-based american sign language game for deaf children. In *Extended Abstracts on Human Factors in Computing Systems (CHI'05)*, pages 1589–1592, 2005.

Leigh, W., Purvis, R., and Ragusa, J. M. Forecasting the nyse composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support. *Decision support systems*, 32(4):361–377, 2002.

Li, H., Cabibihan, J.-J., and Tan, Y. K. Towards an effective design of social robots. *International Journal of Social Robotics*, 3(4):333–335, 2011.

Li, H. and Greenspan, M. Segmentation and recognition of continuous gestures. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I–365. IEEE, 2007.

Li, Y., Pont, M. J., and Jones, N. B. Improving the performance of radial basis function classifiers in condition monitoring and fault diagnosis applications whereunknown'faults may occur. *Pattern Recognition Letters*, 23(5):569–577, 2002.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423, 2006.

Liao, Y., Vemuri, V. R., and Pasos, A. Adaptive anomaly detection with evolving connectionist systems. *Journal of Network and Computer Applications*, 30(1):60–80, 2007.

Lin, H.-J., Kao, Y.-T., Yang, F.-W., and Wang, P. S. Content-based image retrieval trained by adaboost for mobile application. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(04):525–541, 2006.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Lütz, A., Rodner, E., and Denzler, J. I want to know more-efficient multi-class incremental learning using gaussian processes. *Pattern recognition and image analysis*, 23(3):402–407, 2013.

MacQueen, J. et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

Madeo, R. C. B., Lima, C. a. M., and Peres, S. M. Gesture unit segmentation using support vector machines. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, pages 46–52, 2013a.

Madeo, R. C. B., Wagner, P. K., and Peres, S. M. A review of temporal aspects of hand gesture analysis applied to discourse analysis and natural conversation. *International Journal of Computer Science & Information Technology*, 2013b.

Madeo, R. C. B., Peres, S. M., and de Moraes Lima, C. A. Gesture phase segmentation using support vector machines. *Expert Systems with Applications*, 56:100–115, 2016.

Maimon, O. and Rokach, L. Ensemble of decision trees for mining manufacturing data sets. *Machine Engineering*, 4(1-2):32–57, 2004.

Malima, A., Ozgur, E., and Çetin, M. A fast algorithm for vision-based hand gesture recognition for robot control. In *14th Signal Processing and Communications Applications*, pages 1–4. IEEE, 2006.

Mańdziuk, J. and Shastri, L. Incremental class learning approach and its application to handwritten digit recognition. *Information Sciences*, 141(3):193–217, 2002.

Manevitz, L. and Yousef, M. Learning from positive data for document classification using neural networks, 2000.

Mangiameli, P., West, D., and Rampal, R. Model selection for medical diagnosis decision support systems. *Decision Support Systems*, 36(3):247–259, 2004.

Markou, M. and Singh, S. Novelty detection: a reviewpart 2:: neural network based approaches. *Signal processing*, 83(12):2499–2521, 2003.

Marsland, S. *Machine Learning: An Algorithmic Perspective*. Chapman and Hall/CRC, 2009. ISBN 1420067184.

Martyna, G. J., Klein, M. L., and Tuckerman, M. Nosé–hoover chains: the canonical ensemble via continuous dynamics. *The Journal of chemical physics*, 97(4):2635–2643, 1992.

Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859–874, 2011.

Masud, M. M., Woolam, C., Gao, J., Khan, L., Han, J., Hamlen, K. W., and Oza, N. C. Facing the reality of data stream classification: coping with scarcity of labeled data. *Knowledge and information systems*, 33(1):213–244, 2012.

McNeill, D. *Hand and mind: What gestures reveal about thought*. University of Chicago press, 1992.

McNeill, D. *Gesture and thought*. University of Chicago Press, 2008.

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

Merkwirth, C., Mauser, H., Schulz-Gasch, T., Roche, O., Stahl, M., and Lengauer, T. Ensemble methods for classification in cheminformatics. *Journal of chemical information and computer sciences*, 44(6):1971–1978, 2004.

Microsoft. Skeleton positions relative to the human body, 2013a. URL `https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx`. Accessed: 2017-01-11.

Microsoft. Kinect for windows sdk. *Microsoft Developer*, 2013b. URL `http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx`.

Migdadi, L. Novelty detection in gesture recognition systems. Master's thesis, University of Dortmund, 2015.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K.-R. Fisher discriminant analysis with kernels. In *Proceedings of the 1999 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing IX, Madison, WI, USA*, pages 23–25, 1999.

Mistry, P. and Maes, P. Sixthsense: a wearable gestural interface. In *ACM SIGGRAPH ASIA 2009 Sketches*, page 11. ACM, 2009.

Mitchell, T. M. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

Morguet, P. and Lang, M. Spotting dynamic hand gestures in video image sequences using hidden markov models. In *International Conference on Image Processing, 1998. ICIP 98.*, pages 193–197. IEEE, 1998.

Müller, M. *Information retrieval for music and motion*, volume 2. Springer, 2007.

Murakami, K. and Taguchi, H. Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 237–242. ACM, 1991.

Murugappan, S., Liu, H., Ramani, K., et al. Shape-it-up: Hand gesture based creative expression of 3d shapes using intelligent generalized cylinders. *Computer-Aided Design*, 45(2):277–287, 2013.

Nehaniv, C. L. Classifying types of gesture and inferring intent. In *Procs of the AISB 05 Symposium on Robot Companions*. AISB, 2005.

Neto, F., Meira, S., et al. Improving novelty detection in short time series through RBF-DDA parameter adjustment. In *IEEE International Joint Conference on Neural Networks*, pages 2123–2128. IEEE, 2004.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2-3):103–134, 2000.

Niu, W., Jiao, L., Han, D., and Wang, Y.-F. Real-time multiperson tracking in video surveillance. In *Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, and Fourth Pacific Rim Conference on Multimedia*, volume 2, pages 1144–1148. IEEE, 2003.

Niu, Z.-Y., Ji, D., Tan, C.-L., and Yang, L. Word sense disambiguation by semi-supervised learning. In *Computational Linguistics and Intelligent Text Processing*, pages 238–241. Springer, 2005.

Ntalampiras, S., Potamitis, I., and Fakotakis, N. Probabilistic novelty detection for acoustic surveillance under real-world conditions. *IEEE Transactions on Multimedia*, 13(4):713–719, 2011.

Oka, R. Spotting method for classification of real world data. *The Computer Journal*, 41 (8):559–565, 1998.

Oliveira, A. L., Neto, F. B., and Meira, S. R. Novelty detection for short time series with neural networks. In *Design and application of hybrid intelligent systems*, pages 66–75. IOS Press, 2003.

Ottenheimer, H. J. *The anthropology of language: an introduction to linguistic anthropology*. Cengage Learning, 2008.

Parzen, E. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

Pavlovic, V. I., Sharma, R., and Huang, T. S. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pepper. Softbank Robotics, 2017. URL `https://www.ald.softbankrobotics.com/en/cool\discretionary{-}{}{}robots/pepper`. Accessed: 2017-01-04.

Pieters, C. M. The moon as a spectral calibration standard enabled by lunar samples: The clementine example. In *New Views of the Moon 2: Understanding the Moon Through the Integration of Diverse Datasets*, volume 1, page 47, 1999.

Pieters, C., Besse, S., Boardman, J., Buratti, B., Cheek, L., Clark, R., Combe, J., Dhingra, D., Goswami, J., Green, R., et al. Mg-spinel lithology: A new rock type on the lunar farside. *Journal of Geophysical Research: Planets*, 116(E6), 2011.

Pimentel, M., Clifton, D., Clifton, L., and Tarassenko, L. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Polikar, R., Upda, L., Upda, S. S., and Honavar, V. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(4):497–508, 2001.

Quam, D. L. Gesture recognition with a dataglove. In *Proceedings of the IEEE National Aerospace and Electronics Conference, (NAECON 1990).*, pages 755–760. IEEE, 1990.

Quattoni, A., Wang, S., Morency, L.-P., Collins, M., and Darrell, T. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1848–1852, 2007.

Quek, F., McNeill, D., Bryll, R., Duncan, S., Ma, X.-F., Kirbas, C., McCullough, K. E., and Ansari, R. Multimodal human discourse: Gesture and speech. *ACM Trans. Comput.-Hum. Interact.*, 9:171–193, 2002.

Quek, F. K. Toward a vision-based hand gesture interface. In *Virtual Reality Software and Technology Conference*, volume 94, pages 17–29, 1994.

Quek, F. K. Eyes in the interface. *Image and vision computing*, 13(6):511–525, 1995.

Rabiner, L. and Juang, B.-H. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-015157-2.

Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Ramakrishnan, A. S. *Segmentation of hand gestures using motion capture data*. University of California, Davis, 2011.

Ramakrishnan, A. S. and Neff, M. Segmentation of hand gestures using motion capture data. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1249–1250, 2013.

Ramani, K. et al. Extracting hand grasp and motion for intent expression in mid-air shape deformation: A concrete and iterative exploration through a virtual pottery application. *Computers & Graphics*, 55:143–156, 2016.

Rao, C. R. The Utilization of Multiple Measurements in Problems of Biological Classification. *Journal of the Royal Statisticl Society*, X(2):159–203, 1948.

Rao, C. R. and Mitra, S. K. *Generalized inverse of matrices and its applications*, volume 7. Wiley New York, 1971.

Ratsaby, J. and Venkatesh, S. S. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 412–417. ACM, 1995.

Ren, A. Gesture research, 2017. URL `http://adainspired.mit.edu/gesture-research/`. Accessed: 2017-01-12.

Resnick, S. I. *Extreme values, regular variation and point processes*. Springer, 2013.

Richarz, J. and Fink, G. A. Visual recognition of 3d emblematic gestures in an hmm framework. *Journal of Ambient Intelligence and Smart Envirements 3*, pages 193–211, 2011.

Riloff, E., Wiebe, J., and Wilson, T. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32, 2003.

Roark, B., Saraclar, M., Collins, M., and Johnson, M. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 47, 2004.

Roberts, S. J. Novelty detection using extreme value statistics. *IEE Proceedings-Vision, Image and Signal Processing*, 146(3):124–129, 1999.

Roberts, S. J. Extreme value statistics for novelty detection in biomedical data processing. *IEE Proceedings-Science, Measurement and Technology*, 147(6):363–367, 2000.

Rokach, L. *Pattern Classification Using Ensemble Methods*. World Scientific Publishing Co., Inc., 2010. ISBN 9789814271066, 9814271063.

Rosenberg, C., Hebert, M., and Schneiderman, H. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTIONS).*, volume 1, pages 29–36. IEEE Press, 2005.

Roudavski, S., Dave, B., Li, A., Gu, N., and Park, H. Virtual environments as techno-social performances: virtual west cambridge case-study. In *CAADRIA2010: New Frontiers, the 15th International Conference on Computer Aided Architectural Design Research in Asia*, pages 477–486, 2010.

Ruffieux, S., Lalanne, D., and Mugellini, E. Chairgest: a challenge for multimodal mid-air gesture recognition for close hci. In *Proceedings of the 15th ACM International conference on multimodal interaction*, pages 483–488. ACM, 2013.

Ryoo, M. S. and Aggarwal, J. K. Recognition of composite human activities through context-free grammar based representation. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1709–1718. IEEE, 2006.

Sakic, D. Semi-supervised learning using ensemble methods gestures recognition. Master's thesis, University of Dortmund, 2012.

Salem, M., Kopp, S., Wachsmuth, I., Rohlfing, K., and Joublin, F. Generation and evaluation of communicative robot gesture. *International Journal of Social Robotics*, 4 (2):201–217, 2012.

Sanin, A., Sanderson, C., Harandi, M. T., and Lovell, B. C. Spatio-temporal covariance descriptors for action and gesture recognition. In *2013 IEEE Workshop on applications of Computer Vision (WACV)*, pages 103–110. IEEE, 2013.

Sato, E., Yamaguchi, T., and Harashima, F. Natural interface using pointing behavior for human–robot gestural interaction. *IEEE transactions on Industrial Electronics*, 54 (2):1105–1112, 2007.

Schapire, R. E. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

Schlömer, T., Poppinga, B., Henze, N., and Boll, S. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14. ACM, 2008.

Schneider, P., Biehl, M., and Hammer, B. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

Scholten, F., Oberst, J., Matz, K.-D., Roatsch, T., Wählisch, M., Speyerer, E. J., and Robinson, M. S. GLD100: The near-global lunar 100 m raster DTM from LROC WAC stereo image data. *Journal of Geophysical Research*, 117, 2012.

Schröder, M., Elbrechter, C., Maycock, J., Haschke, R., Botsch, M., and Ritter, H. Real-time hand tracking with a color glove for the actuation of anthropomorphic robot hands. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 262–269. IEEE, 2012.

Schumacher, J., Sakič, D., Grumpe, A., Fink, G. A., and Wöhler, C. Active learning of ensemble classifiers for gesture recognition. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 498–507. Springer, 2012.

Schürmann, J. *Pattern classification: a unified view of statistical and neural approaches*. Wiley Online Library, 1996.

Scudder, H. J. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

Seki, S., Takahashi, K., and Oka, R. Gesture recognition from motion images by spotting algorithm. In *Proc. ACCV*, volume 2, pages 759–762, 1993.

Settles, B. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66): 11, 2010.

Shin, M. C., Tsap, L. V., and Goldgof, D. B. Gesture recognition using bezier curves for visualization navigation from registered 3-d data. *Pattern Recognition*, 37(5):1011–1024, 2004.

Shkuratov, Y. G., Kaydash, V. G., Stankevich, D. G., Starukhina, L. V., Pinet, P. C., Chevrel, S. D., and Daydou, Y. H. Derivation of elemental abundance maps at intermediate resolution from optical interpolation of lunar prospector gamma-ray spectrometer data. *Planetary and Space Science*, 53(12):1287–1301, 2005.

Singh, S. and Markou, M. An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):396–407, 2004.

Sohn, H., Worden, K., and Farrar, C. R. Novelty detection under changing environmental conditions. In *Proc. SPIE*, volume 4330, pages 108–118, 2001.

Song, Y., Demirdjian, D., and Davis, R. Continuous body and hand gesture recognition for natural human-computer interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(1):5, 2012.

Sonka, M., Hlavac, V., and Boyle, R. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

Soutschek, S., Penne, J., Hornegger, J., and Kornhuber, J. 3-d gesture-based scene navigation in medical imaging applications using time-of-flight cameras. In *CVPRW Workshops*, pages 1–6. IEEE, 2008.

Spano, L. D., Cisternino, A., and Paternò, F. A compositional model for gesture definition. In *International Conference on Human-Centred Software Engineering*, pages 34–52. Springer, 2012.

Starner, T., Weaver, J., and Pentland, A. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

Starner, T., Auxier, J., Ashbrook, D., and Gandy, M. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *the fourth international symposium on Wearable computers.*, pages 87–94. IEEE, 2000.

Stefanov, N., Galata, A., and Hubbold, R. Real-time hand tracking with variable-length markov models of behaviour. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 73–73. IEEE, 2005.

Stiefelhagen, R., Fugen, C., Gieselmann, R., Holzapfel, H., Nickel, K., and Waibel, A. Natural human-robot interaction using speech, head pose and gestures. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2422–2427, 2004.

Su, M.-C. A fuzzy rule-based approach to spatio-temporal hand gesture recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(2):276–281, 2000.

Subramanya, A. and Talukdar, P. P. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.

Surace, C., Worden, K., and Tomlinson, G. A novelty detection approach to diagnose damage in a cracked beam. In *Proceedings-SPIE The International Society For Optical Engineering*, pages 947–953, 1997.

Suykens, J. A. K. and Vandewalle, J. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

Syed, N. A., Huan, S., Kah, L., and Sung, K. Incremental learning with support vector machines. In *of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99)*, Stockholm, Sweden, 1999.

Synertial Labs LTD. IGS-Cobra, 2017. URL `https://synertial.com/products/suits/`. Accessed: 2017-01-11.

Takahashi, K., Seki, S., and Oka, R. Spotting recognition of human gestures from motion images. *Time Varying and moving objects recognition*, 3:10–11, 1992.

Tan, A. C., Gilbert, D., and Deville, Y. Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics*, 14:206–217, 2003.

Tanaka, F., Isshiki, K., Takahashi, F., Uekusa, M., Sei, R., and Hayashi, K. Pepper learns together with children: Development of an educational application. In *15th International Conference on Humanoid Robots (Humanoids), 2015 IEEE-RAS*, pages 270–275. IEEE, 2015.

Tarassenko, L., Hayton, P., Cerneaz, N., and Brady, M. Novelty detection for the identification of masses in mammograms. In *4th IEEE International Conference on Artificial Neural Networks*, volume 4, pages 442–447. IET, 1995.

Tavakkoli, A., Nicolescu, M., Nicolescu, M., and Bebis, G. Incremental svdd training: Improving efficiency of background modeling in videos. In *Proceedings of the 10th IASTED International Conference*, volume 623, page 092, 2008.

Tax, D. M. J. and Duin, R. P. W. Support vector data description. *Machine learning*, 54 (1):45–66, 2004.

Tax, D. M. Ddtools, the data description toolbox for matlab, June 2015. URL `http://prlab.tudelft.nl/david-tax/dd_tools.html`. version 2.1.2.

Tax, D. M. and Duin, R. P. Support vector domain description. *Pattern Recognition Letters*, 20(11):1191–1199, 1999.

Tax, D. M. and Laskov, P. Online svm learning: from classification to data description and back. In *13th IEEE Workshop on Neural Networks for Signal Processing, NNSP'03*, pages 499–508. IEEE, 2003.

Tax, D. *One-class classification: concept-learning in the absence of counter-examples*. PhD thesis, TU Delft, Delft University of Technology, 2001.

Thalmann, N. M. and Thalmann, D. *Computer Animation: Theory and Practice*. Springer, 1990.

Theodoridis, S. and Koutroumbas, K. *Pattern Recognition, Fourth Edition*. Academic Press, 2009.

Theofilou, D., Steuber, V., and De Schutter, E. Novelty detection in a kohonen-like network with a long-term depression learning rule. *Neurocomputing*, 52:411–417, 2003.

Thompson, B. B., Marks, R. J., Choi, J. J., El-Sharkawi, M. A., Huang, M.-Y., and Bunje, C. Implicit learning in autoencoder novelty assessment. In *International Joint Conference on Neural Networks, 2002. IJCNN'02.*, volume 3, pages 2878–2883. IEEE, 2002.

Thompson, D. Biomechanics of the hand. *Perspectives in computing*, 1(3):12–19, 1981.

Toriyama, C., Kawanishi, Y., Takahashi, T., Deguchi, D., Ide, I., Murase, H., Aizawa, T., and Kawade, M. Hand waving gesture detection using a far-infrared sensor array with thermo-spatial region of interest. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, volume 4, pages 545–551, 2016.

Tran, C., Doshi, A., and Trivedi, M. M. Modeling and prediction of driver behavior by foot gesture analysis. *Computer Vision and Image Understanding*, 116(3):435–445, 2012.

Tseng, F. et al. Real time novelty detection modeling for machine health prognostics. In *Annual Meeting of the North American Fuzzy Information Processing Society - NAFIPS-2006 Meeting*, pages 529–534, 2006.

Tukey, J. *Exploratory Data Analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company, 1977.

Turk, M. Gesture recognition. *Computer Vision: A Reference Guide*, pages 346–349, 2014.

Vaananen, K. and Bohm, K. Gesture driven interaction as a human factor in virtual environments-an approach with neural networks. *Virtual reality systems*, pages 93–106, 1993.

Vail, D. L., Veloso, M. M., and Lafferty, J. D. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 235. ACM, 2007.

Valpola, H. and Karhunen, J. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural computation*, 14(11):2647–2692, 2002.

Vapnik, V. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

Vapnik, V. N. *Statistical learning theory*, volume 1. Wiley New York, 1998.

Vasconcelos, G. C., Fairhurst, M. C., and Bisset, D. L. A bootstrap-like rejection mechanism for multilayer perceptron networks. In *II Simposio Brasileiro de Redes Neurais, São Carlos-SP, Brazil*, pages 167–172. Citeseer, 1995.

Vedaldi, A. and Soatto, S. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, pages 705–718. Springer, 2008.

Vogel, D. and Balakrishnan, R. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42. ACM, 2005.

Vogler, C. and Metaxas, D. Parallel hidden markov models for american sign language recognition. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 116–122. IEEE, 1999.

Wachs, J., Stern, H., Edan, Y., Gillam, M., Feied, C., Smith, M., and Handler, J. A real-time hand gesture interface for medical visualization applications. In *Applications of Soft Computing*, pages 153–162. Springer, 2006.

Wachs, J. P., Stern, H. I., Edan, Y., Gillam, M., Handler, J., Feied, C., and Smith, M. A gesture-based tool for sterile browsing of radiology images. *Journal of the American Medical Informatics Association*, 15(3):321–323, 2008.

Wachs, J. P., Kölsch, M., Stern, H., and Edan, Y. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

Wahba, G. et al. Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87, 1999.

Wang, H., Ullah, M. M., Klaser, A., Laptev, I., and Schmid, C. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, pages 124–1. BMVA Press, 2009.

Wang, J., Thiesson, B., Xu, Y., and Cohen, M. Image and video segmentation by anisotropic kernel mean shift. In *European conference on computer vision*, pages 238–249. Springer, 2004.

Wang, S. B., Quattoni, A., Morency, L.-P., Demirdjian, D., and Darrell, T. Hidden conditional random fields for gesture recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1521–1527. IEEE, 2006.

Webb, A. R. *Statistical pattern recognition.* John Wiley & Sons, New York, USA, 2003.

Webb, J. and Ashley, J. *Beginning Kinect Programming with the Microsoft Kinect SDK.* Apress, 2012.

Weston, J., Bengio, S., and Usunier, N. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.

Wikipedia. Leap Motion, 2016. URL `http://en.wikipedia.org/wiki/Leap_Motion`. Accessed: 2016-07-25.

Williams, G., Baxter, R., He, H., Hawkins, S., and Gu, K. A comparative study of RNN for outlier detection in data mining. In *IEEE International Conference on Data Mining*, ICDM '02, pages 709–712, 2002.

Wilson, A. D. and Bobick, A. F. Parametric hidden markov models for gesture recognition. *IEEE transactions on pattern analysis and machine intelligence*, 21(9):884–900, 1999.

Wilson, A. D. and Bobick, A. F. Realtime online adaptive gesture recognition. In *15th International Conference on Pattern Recognition.*, volume 1, pages 270–275. IEEE, 2000.

Wöhler, C., Berezhnoy, A., and Evans, R. Estimation of elemental abundances of the lunar regolith using clementine UVVIS+NIR data. *Planetary and Space Science*, 59: 92–110, 2011.

Wöhler, C., Grumpe, A., Berezhnoy, A., Bhatt, M. U., and Mall, U. Integrated topographic and spectral analysis of the lunar surface: Application to impact melt flows and ponds. *Icarus*, 235:86–122, 2014. doi: 10.1016/j.icarus.2014.03.010.

Wong, S.-F. and Cipolla, R. Continuous gesture recognition using a sparse bayesian classifier. In *18th IEEE International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 1084–1087. IEEE, 2006.

Worden, K., Allen, D. W., Sohn, H., and Farrar, C. R. Damage detection in mechanical structures using extreme value statistics. In *SPIE's 9th Annual International Symposium on Smart Structures and Materials*, pages 289–299. International Society for Optics and Photonics, 2002.

Xu, J., Gannon, P. J., Emmorey, K., Smith, J. F., and Braun, A. R. Symbolic gestures and spoken language are processed by a common neural system. *Proceedings of the National Academy of Sciences*, 106(49):20664–20669, 2009.

Xun, D. and Chang-shan, W. An efficient sequential learning algorithm for growing and pruning direct-link rbf (drbf) networks. In *IEEE International Conference on Neural Networks and Brain*, volume 1, pages 494–498. IEEE, 2005.

Yang, H.-D., Park, A.-Y., and Lee, S.-W. Robust spotting of key gestures from whole body motion sequence. In *7th IEEE International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 231–236. IEEE, 2006.

Yang, H.-D., Sclaroff, S., and Lee, S.-W. Sign language spotting with a threshold model based on conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1264–1277, 2009a.

Yang, L., Jin, R., and Sukthankar, R. Semi-supervised learning with weakly-related unlabeled data: Towards better text categorization. In *Advances in Neural Information Processing Systems*, pages 1857–1864, 2009b.

Yang, Y., Wang, Y., and Yuan, X. Parallel chaos search based incremental extreme learning machine. *Neural processing letters*, 37(3):277–301, 2013.

Yao, Y. *Hand gesture recognition in uncontrolled environments*. PhD thesis, University of Warwick, 2014.

Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

Ye, Y., Squartini, S., and Piazza, F. Online sequential extreme learning machine in nonstationary environments. *Neurocomputing*, 116:94–101, 2013.

Yeung, D.-Y. and Chow, C. Parzen-window network intrusion detectors. In *Proceedings of the 16th IEEE International Conference on Pattern Recognition*, volume 4, pages 385–388. IEEE, 2002.

Yin, G., Zhang, Y.-T., Li, Z.-N., Ren, G.-Q., and Fan, H.-B. Online fault diagnosis method based on incremental support vector data description and extreme learning machine with incremental output structure. *Neurocomputing*, 128:224–231, 2014a.

Yin, Y. et al. *Real-time continuous gesture recognition for natural multimodal interaction*. PhD thesis, Massachusetts Institute of Technology, 2014b.

Yoon, H.-S., Soh, J., Bae, Y. J., and Yang, H. S. Hand gesture recognition using combined features of location, angle and velocity. *Pattern recognition*, 34(7):1491–1501, 2001.

Yorita, A. and Kubota, N. Cognitive development in partner robots for information support to elderly people. *IEEE Transactions on Autonomous Mental Development*, 3 (1):64–73, 2011.

Ypma, A., Ypma, E., and Duin, R. P. Novelty detection using self-organizing maps. In *In Proc. of ICONIP'97*, pages 1322–1325. Springer, 1997.

Yusoff, Y. A., Basori, A. H., and Mohamed, F. Interactive hand and arm gesture control for 2d medical image and 3d volumetric medical visualization. *Procedia-Social and Behavioral Sciences*, 97:723–729, 2013.

Zhang, B.-f., Su, J.-s., and Xu, X. A class-incremental learning method for multi-class support vector machines in text classification. In *IEEE International Conference on Machine Learning and Cybernetics*, pages 2581–2585. IEEE, 2006.

Zhang, T. and Ando, R. Analysis of spectral kernel design based semi-supervised learning. *Advances in neural information processing systems*, 18:1601, 2006.

Zhao, Z., Chen, Z., Chen, Y., Wang, S., and Wang, H. A class incremental extreme learning machine for activity recognition. *Cognitive Computation*, 6(3):423–431, 2014.

Zhou, X., Zhuang, X., Yan, S., Chang, S.-F., Hasegawa-Johnson, M., and Huang, T. S. Sift-bag kernel for video event analysis. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 229–238. ACM, 2008.

Zhou, Z.-H., Zhan, D.-C., and Yang, Q. Semi-supervised learning with very few labeled training examples. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 675. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Zhu, X. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Zhu, X. Semi-supervised learning. In *Encyclopedia of machine learning*, pages 892–897. Springer, 2011.

Zhu, X. and Goldberg, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

Zhu, Y., Xu, G., and Kriegman, D. J. A real-time approach to the spotting, representation, and recognition of hand gestures for human–computer interaction. *Computer Vision and Image Understanding*, 85(3):189–208, 2002.

Zorriassatine, F., Al-Habaibeh, A., Parkin, R., Jackson, M., and Coy, J. Novelty detection for practical pattern recognition in condition monitoring of multivariate processes: a case study. *The International Journal of Advanced Manufacturing Technology*, 25(9-10): 954–963, 2005.

List of Figures

List of Tables

List of Symbols

Symbol	Unit[1]	Description
$\boldsymbol{\Sigma}$	–	Covariance matrix of data
$\boldsymbol{\Sigma}'$	–	Covariance matrix of new data
$\hat{\boldsymbol{\Sigma}}$	–	Total covariance matrix of the old and the new data
σ_{evt}	–	Scale parameter in the EVT
$\boldsymbol{\Sigma}_{\text{res}}$	–	Variance of the residuals
σ_{rbf}	–	Predefined width of the RBF kernel function
σ_{graph}	–	Bandwidth parameters in the graphical semi-supervised learning
σ_{msh}	–	Window size of the mean shift clustering algorithm
α_{evt}	–	Shape parameter in the EVT
α_{conf}	–	Probability threshold of the confidence band
μ_{evt}	–	Location parameter in the EVT
$\vec{\mu}_{\text{cent}}$	–	Centroid position
$\vec{\mu}_{\text{res}}$	–	Mean of the residual
$\vec{\mu}$	–	Mean of data
$\vec{\mu}'$	–	Mean of new data
$\hat{\vec{\mu}}$	–	Mean of the total data i.e. the old and the new data
$\tilde{\vec{\mu}}$	–	Projection of mean vector $\vec{\mu}$ on a new space
Ψ	–	Transform maps the distribution of f to a space that fit the Gumbel distribution

[1]Some quantities, e.g. error functions or polynomial coefficients, have physical units. For simplicity, these units are ignored.

Symbol	Unit	Description
γ	–	Learning steps
$\boldsymbol{\beta}_{\mathrm{E}}$	–	Extreme learning machine parameters
$\hat{\boldsymbol{\beta}}_{\mathrm{E}}$	–	Approximate ELM parameter
$\boldsymbol{\beta}_{\mathrm{AE}}$	–	Extreme learning machine parameters when using it as an auto-encoder
$\boldsymbol{\beta}_{\mathrm{P}}$	–	Parameter matrix of the polynomial classifier
$\hat{\boldsymbol{\beta}}_{\mathrm{P}}$	–	Approximate polynomial parameter
η_{conf}	–	Wide of the confidence band
θ_{prob}	–	Probabilistic model parameters
θ	–	Edge weight in the graphical semi-supervised learning
θ_{drift}	–	Factor to cover the concept's movement in the feature space
$\vec{\varphi}$	–	Non-linear mapping to the high-dimensional hyper-space
A	–	First parameter of the sigmoid function
$\vec{a}$	–	ELM's hidden layer function learning parameter
B	–	Second parameter of the sigmoid function
b	–	ELM's hidden layer function learning second parameter
b'	–	Optimal bias of the SVM
C_d	–	Normalisation coefficient of the mahalanobis function of the EVT
$\boldsymbol{C}_{\mathrm{W}}$	–	Accumulated minimum warping cost matrix in the DTW
c	–	c'th class
$\hat{c}$	–	Estimated class label
$\mathcal{D}$	–	Training dataset
$\mathcal{D}_{\mathrm{l}}$	–	Labelled training dataset
$\mathcal{D}_{\mathrm{u}}$	–	Unlabelled training dataset
d	–	Dimension of the data
d'	–	New dimension of the data
d_{Eucl}	–	Euclidean distance
d_{Mahal}	–	Mahalanobis distance
d_{Kernel}	–	Kernel distance
d_{rbf}	–	RBF kernel distance between two vectors
d_{bh}	–	Bhattacharyya distance between two distributions

Symbol	Unit	Description
d_{W}	–	Distance between two vectors projected on the $\boldsymbol{W}$ sub-space
E_{total}	–	Total misclassification rate
F_{e}	–	Percentage of samples belonging to a known class and assigned to an incorrectly known class
F_{n}	–	Percentage of the outliers that the classifier misses
F_{p}	–	Percentage of existing class samples which are wrongly indicated as outliers by the classifier
F_{new}	–	Percentage of samples belonging to known classes indicated as an unknown
F^{e}	–	Accumulative extreme value distribution for the distribution f
f	–	Probability density function
f^{e}	–	Extreme value distribution for the distribution f
f_{svm}	–	SVM classifier function
f_{elm}	–	ELM classifier function
f_{ssl}	–	Mapping function of the semi-supervised learning
G_d	–	Form of the distribution function (df) according to which f is distributed on probability distribution set
G_d^{e}	–	Extreme value distribution of G_d
$G_d^{\leftarrow}$	–	Quantile of the function G_d
G_{h}	–	Hidden neuron output
g	–	Weighting function that defines the EVD in terms of f
$\mathcal{H}$	–	Set of the data after they are transformed to hyper-space
$\mathcal{H}^+$	–	Extreme value distribution (EVD) of the maxima
$\boldsymbol{H}_{\mathrm{E}}$	–	Output of the hidden layer
$\boldsymbol{H}'_{\mathrm{E}}$	–	Output of the hidden layer that corresponds to new data
$\boldsymbol{H}_{\mathrm{P}}$	–	Matrix of polynomial basis features
$\boldsymbol{H}'_{\mathrm{P}}$	–	Matrix of polynomial basis features that corresponds to new data
h_{w}	–	Parzen window size

Symbol	Unit	Description
K	–	Kernel function
K_{RBF}	–	RBF kernel function
L	–	Number of variables in the polynomial basis vector
l	–	Number of labelled data
M	–	Mahalanobis distance function
$\boldsymbol{M}_{\mathrm{E}}$	–	Squared hidden layer output of the extreme learning machine
$\boldsymbol{M}'_{\mathrm{E}}$	–	Squared hidden layer output corresponding to new data
$\boldsymbol{M}_{\mathrm{P}}$	–	Squared of polynomial basis features
$\boldsymbol{M}'_{\mathrm{P}}$	–	Squared of the new polynomial basis features
M_{new}	–	Fraction of missed novel samples
$\mathcal{N}$	–	Normal distribution set
N	–	Number of the samples in the dataset
N'	–	Number of the new samples
N_{cent_j}	–	Number of samples in the centroid j
N'_{cent_j}	–	Number of new samples that should add to the centroid j
N_{total}	–	Number of the total samples in the data set (particularly in test set)
N_{novel}	–	Number of novel samples in the data set (particularly in test set)
N_{class}	–	Number of classes
N_{c}	–	Number of sample in the class c
$N_{\hat{\mathrm{c}}}$	–	Number of sample in the newly constructed class $\hat{\mathrm{c}}$
N_{cent}	–	Number of the centroids of the data
N_{b}	–	Number of sample in a bucket
N_{p}	–	Number of free parameters of the model
N^{+}	–	Number of samples belong to the first class in the binary SVM classifier
N^{-}	–	Number of samples belong to the second class in the binary SVM classifier
N_{th}	–	Novelty threshold of normal methods
N_{Sc}	–	Novelty score of normal methods
n	–	Number of neurons in the hidden layer of the neural network
n_{p}	–	Polynomial degree
$\mathcal{P}$	–	Probability distribution set

Symbol	Unit	Description
$\boldsymbol{P}_{\mathrm{E}}$	–	Multiplication of the hidden layer output and the target matrix
$\boldsymbol{P}'_{\mathrm{E}}$	–	Multiplication of the hidden layer output and the target matrix for the new data
$\boldsymbol{P}_{\mathrm{AE}}$	–	Multiplication of the output of the hidden layer in the auto-encoder ELM by the input matrix
$\boldsymbol{P}_{\mathrm{P}}$	–	Equals to $\boldsymbol{H}_{\mathrm{P}} \times \boldsymbol{T}_{\mathrm{P}}$
$\boldsymbol{P}'_{\mathrm{P}}$	–	Equals to $\boldsymbol{H}'_{\mathrm{P}} \times \boldsymbol{T}'_{\mathrm{P}}$
$\vec{P}_{\mathrm{evt}}$	–	Scores of the extreme value theory
P_{evt}	–	Minimum score of the extreme value theory
P_{th}	–	Novelty threshold on the EVT outputs
p_i	–	Symbole of $p(y = 1\|f_{svm})$ for the sample i
p_{u}	–	Purity of the newly constructed class
$p(x)$	–	Probability distribution of x
$p(x, y)$	–	Joint probability of x and y distributions
$p(x\|y)$	–	Class conditional probability of distribution x given y distribution
$p(y\|x)$	–	Predictive probability or posterior
$\mathbb{R}$	–	Continuous set
r	–	Residual value between the estimated value and the target value
$\mathcal{S}$	–	Subset from the available dataset
$\boldsymbol{T}_{\mathrm{E}}$	–	Target matrix of the ELM for the input samples
$\boldsymbol{T}'_{\mathrm{E}}$	–	Target matrix of the ELM for the new input samples
$\hat{\boldsymbol{T}}_{\mathrm{E}}$	–	Estimated target matrix of the ELM for the input samples
$\boldsymbol{T}_{\mathrm{P}}$	–	Target matrix of the polynomial classifier for the input samples
$\boldsymbol{T}'_{\mathrm{P}}$	–	Target matrix of the polynomial classifier for the new input samples
$\hat{\boldsymbol{T}}_{\mathrm{P}}$	–	Estimated target matrix of the polynomial classifier for the input samples
T_{update}	sec	Processing time of the classifier update
$\vec{t}$	–	Target vector
$\vec{\hat{t}}$	–	Estimated target of the input sample

Symbol	Unit	Description
$\vec{t}_P$	–	Target vector of the input sample to the polynomial classifier
$\vec{\hat{t}}_P$	–	Polynomial classifier estimated target vector of the input sample
t_i	–	Target output of the classifier for the input $\vec{x}_i$
$t_{\mathrm{v},\alpha_{\mathrm{conf}}}$	–	Critical value of the t-distribution
u	–	Number of unlabelled data
v	–	Number of degrees of freedom
$\boldsymbol{W}$	–	Low-rank projection matrix
$\vec{w}$	–	Warping path in the DTW
$\vec{w}_{\mathrm{svm}}$	–	Optimal weights of SVM
$\mathcal{X}$	–	Input sequence or input feature vectors dataset
$\mathcal{X}_{\mathrm{l}}$	–	Only the labelled subset of the input feature vectors dataset
$\mathcal{X}_{\mathrm{u}}$	–	Unlabelled subset of the input feature vectors dataset
$\boldsymbol{X}$	–	Set of input feature vectors
$\vec{x}$	–	Sample from the data set
$\vec{x}_i$	–	Input vector of the i'th sample in $\mathcal{X}$
$\vec{\tilde{x}}$	–	Projection of input vector $\vec{x}$ on new space
$\vec{x}'$	–	New sample from the data set
$\mathcal{Y}$	–	Label set of the input dataset $\mathcal{X}$
y	–	Label of a sample $\vec{x}$
y_i	–	Label of the i'th input vector x_i
$\hat{y}$	–	Estimated label of an input samples $\vec{x}$
y_{e}	–	Reduced variate of $\vec{x}$ that used in the EVT
z	–	Constant that is used in the confidence bands

A

Reference Methods

Details about the classifiers that are used as references to evaluate the proposed classifiers outputs, e.g. SVDD and AANN are out of the scope of this thesis. They can be found in the Tax and Duin (1999), Tax (2001) and Tax and Duin (2004). The necessary implementation details of the SVDD and the AANN algorithms, however, are given in Section A.1 and Section A.2, respectively.

This chapter has been adapted and/or adopted from: (Al-Behadili et al., 2015a,b,e,f, 2016a,c,b,d)

A.1 Support vector data description

Support vector data description (SVDD) is a potent kernel-based algorithm developed by Tax and Duin (2004). It is a one-class classifier that computes a minimum volume hypersphere that encloses almost, or all, if applicable, of a specific class data as a novelty boundary to that class. Tax and Laskov (2003) proposed an incremental approach of SVDD that uses a part of the training data by adding a new sample and deleting the sample that had no real effect on the performance of the classifier. Although it has a good performance and it was used in several applications, such as novelty detection Clifton (2007), it has many limitations in high-dimensionality, and in the large number of samples used for training due to the optimization (Tavakkoli et al., 2008). Many extension of SVDD are proposed in different applications (e.g. Tavakkoli et al., 2008; Hua and Ding, 2011; Lütz et al., 2013; Chen and He, 2011).

In this thesis, we used SVDD in combination with the function "multic" from the data description toolbox (ddtools) presented by Tax (2015) as both a multi-class novelty detection and an online classifier in order to compare its performance to those of the proposed classifiers. In this toolbox, the sample is considered as novel when it does not fit into any class due to a specific threshold, which is determined by the SVDD. If the posterior exceeds the threshold, the sample will be assigned to the corresponding class. If

there are many classes that have a posterior exceeding their thresholds, the sample will be assigned to the class for which the highest posterior probability has been determined.

Since the SVDD is a one-class classifier constructing a hypersphere enclosing the data, it considers some data which are far from the centre as outliers, and by selecting the ratio of the outliers within this class in the training phase, the SVDD code sets the threshold of the novelty. The default value of the outlier ratio to the total data of each class is 5%. We used the default value since it gives the best performance. The "multic" function selects the class of the maximum output and identifies the sample as an outlier if all classifiers indicate it as an outlier.

A.2 Auto-associative neural networks

Auto-associative neural networks (AANN) (Japkowicz et al., 1995) or auto-encoder networks (AutoENC) as they are called by Tax (2001) are neural networks that learn a data representation (Hertz et al., 1991), i.e. an AANN reconstructs the input pattern at their output layer. In this work, we apply the auto-encoder from the toolbox (ddtools) presented by Tax (2015). In this toolbox, the auto-encoder (the function in the toolbox is called autoenc_dd) architecture has only one hidden layer with h_{auto} hidden units. Sigmoid transfer functions are used for the hidden neurons. The auto-encoder network is trained by minimizing the mean squared deviation of the input from the output. The error is used as a measure for the novelty detection. It is supposed that the target patterns will be reconstructed with smaller errors than outliers. The error E_{AANN} of an input $\vec{x}$ is

$$E_{\text{AANN}} = \|f_{\text{AANN}}(\vec{x}, \vec{w}) - \vec{x}\|^2 \tag{A.1}$$

where f_{ANN} is transfer function of the AANN and $\vec{w}$ is a vector containing its parameters.

The problems of the AANN to novelty detection are the same problems arising from the conventional application of neural networks to classification problems. It requires a pre-defined number of neurons, a learning rate, and stopping criteria from an expert user. We tried a different number of neurons, and we found that the best number is 5which is the default number in the function of the toolbox by Tax and Duin (1999). We also used the default outlier percentage 0.05 that is used to compute the threshold of the novelty as it used in the SVDD classifier, which is in the same toolbox. Again, since the AANN classifier is originally one class classifier in this toolbox, we used the function "multic" to obtain a multi-class classifier.