Diss. ETH No. 20005

Adaptation of activity recognition systems

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

Kilian Förster

Dipl.-Ing. TU Ilmenau
MEngSt University of Auckland
born July 23, 1980
citizen of Germany

accepted on the recommendation of

Prof. Dr. Gerhard Tröster, examiner
Prof. Dr. José del R. Millán, co-examiner

2011

Kilian Förster
Adaptation of activity recognition systems
Diss. ETH No. 20005

ETH Zurich, Switzerland, 2011
ISBN 978-3-909386-22-2

Printed by Lulu.com
Copies may be ordered online from http://www.lulu.com

Acknowledgements

My time in the Wearable Computing Group of ETH Zurich has been a fantastic and rewarding experience. I am grateful to Prof. Dr. Gerhard Tröster, my academic supervisor, for giving me this opportunity, offering his support and allowing me freedom in my work. I would also like to thank Prof. Dr. José del R. Millán for co-examining my PhD thesis.

In addition I would also particularly like to thank Dr. Daniel Roggen for welcoming my participation in the Opportunity project. During our long discussions he provided valuable academic advice and contributions to my work. Thank you too to Dr. Ricardo Chavarriaga for his expertise in electroencephalography, all the awesome ideas he shared and his support within our collaboration, and to Dr. Thomas Stiefmeier for introducing me to the lab in the first months when we shared an office and worked together on the WearIT@Work project.

To all members of the IfE, thank you for the collegial spirit and the warm, supportive environment you provided. In particular, I would like to mention the colleagues who participated in my experiments and contributed to my research: Alberto, Bernd, Bert, Christina, Christoph, Claudia, Clemens, Conny, Fredy, Holger, Johannes, Marc, Martin, Mirco, Niko, Thomas H., Thomas K., Sebastian, Silvan. I would also like to acknowledge the help and support of Ruth, who always had a stamp at hand when one was needed. A valuable contribution to my research was made by Fabian and Samuel within their master projects. I'm grateful for their initiative and hard work.

As my spare time activities provided the essential balance to my work in the last few years, I would like to acknowledge all my friends who made that time memorable. In particular, thanks to Lisa for all the weekend visits to Switzerland, and to Lesley for not only proof-reading this thesis, but also for our numerous and spontaneous trips into the wild.

Finally I want to thank the most important people to me, my parents and my sister. They have always been there supporting me in every way possible, and if it was not for their continuous help and advice I would not have been able to pursue my goals.

Zürich, November 2011 Kilian Förster

Contents

Abstract

Advances in mobile computer systems, signal processing and sensing technology enable new computing applications to support the user anywhere and anytime. One such application is activity aware computing where the user's activities are taken into account to provide appropriate assistance.

The crucial element enabling this assistance is activity recognition. Human motion patterns, associated with a specific activity, have to be found in signals captured from motion sensors. These sensors are on the user's body or within their environment. Activity recognition systems are usually trained during the design phase with prerecorded, annotated examples of typical activity patterns. Accounting for all possible user or sensor characteristics, potentially resulting in non-stationary motion signals, would be costly and requires a vast amount of training data. New characteristics might also be developed throughout the ongoing usage of the system, which cannot be captured in training data.

The objective of this thesis is to explore adaptive techniques with the goal of overcoming the restrictions posed by static, trained at design time only, activity recognition systems. We focus on adaptive learning, which has shown potential in different recognition tasks with non-stationary signal characteristics.

Ideally, the activity recognition system adapts continuously to the current signal characteristics, actively guided by the user. Supervision, where the user consciously provides ground truth information, is deemed obtrusive and may interfere with the target application. We therefore investigate incremental learning techniques with particular focus on reducing the amount of supervision required.

One step in this direction is an unsupervised self-calibration strategy, which allows for classifier adaptation. The key benefit of this approach is the reduction of obtrusive supervision required from the user. Validation on two datasets has shown an average recognition accuracy improvement of up to 33.3% and 13.4% for displaced motion sensors. However, improvement can not be guaranteed for each individual case.

The user can also be taken in the loop to indicate system's recognition errors. Indicating recognition errors is less obtrusive compared to providing ground truth, yet enables supervision according to the user's intention. We evaluate two approaches capitalizing on this form of su-

pervision, firstly an incremental learning k-Nearest-Neighbor classifier, and secondly a reinforcement learning method. The adaptation of the recognition system, to a new user, results in an accuracy increase of 10.3% and 36% respectively.

Another possibility, to reduce obtrusiveness of supervision, is to detect system recognition errors implicitly through the user's brain signals. The adaptation of an activity recognition system to a specific user, capitalizing on brain based error feedback, results in an accuracy improvement of 6.8%.

Finally we investigate the influence of an adaptive activity recognition system on the user behavior. In an online gesture recognition experiment users interact with a static as well as an adaptive recognition system. We found indications that the interaction with the adaptive recognition system not only increases the recognition performance, but also enables the users to perform gestures in a way most convenient and natural to them.

Zusammenfassung

Die Miniaturisierung von Computersystemen und Sensoren, sowie Fortschritte in der Signalverarbeitung, erlauben die Nutzung dieser Technologien nahezu jederzeit und an jedem Ort. Dies erschließt neue Anwendungsbereiche, in denen das Computersystem den Benutzer kontinuierlich und situationsabhängig unterstützt. Ein wichtiger Baustein für eine situationsabhängige Unterstützung ist die Erkennung und Verarbeitung von Aktivitäten, die vom Benutzer ausgeführt werden.

Ein Aktivitätserkennungssystem misst die Bewegungen des Benutzers mit Hilfe von Sensoren und durchsucht die resultierenden Signale nach spezifischen Bewegungsmustern. Die Bewegungssensoren können hierbei sowohl am Körper des Benutzers als auch in dessen Umgebung angebracht sein. Die Zuordnung von Bewegungsmustern zu Aktivitätsklassen wird in der Regel basierend auf vorab aufgezeichneten Datensätzen durch Lernalgorithmen erstellt. Um alle möglichen Benutzer- und Sensorcharakteristiken in das Lernen mit einbeziehen zu können, sind umfangreiche Lerndatensätze nötig, deren Erstellung sehr aufwendig ist. Des Weiteren können Signalcharakteristiken, die erst in der Zukunft auftreten, nicht durch einen solchen Lerndatensatz abgedeckt werden.

Das Ziel dieser Arbeit ist es, adaptive Lernmethoden zur Aktivitätserkennung zu untersuchen, um Bewegungsmuster nicht nur einmalig zu lernen, sondern bei Bedarf auch anpassen zu können. Hierbei liegt der Fokus auf adaptiven Lernalgorithmen, die in ähnlicher Form bereits in andere Mustererkennungsanwendungen eingesetzt werden. Im Idealfall passt sich das Aktivitätserkennungssystem kontinuierlich an die Charakteristiken des Eingangssignals an, was durch inkrementelle Lernalgorithmen ermöglicht wird.

Der Benutzer kann die Adaption direkt steuern, indem er zu jeder Zeit angibt, welche Aktivitäten er gerade ausführt. Die Bereitstellung zusätzliche Informationen ist jedoch störend für den Benutzer und schränkt die Benutzbarkeit eines adaptiven Aktivitätserkennungssystems ein. Aus diesem Grund befasst sich ein weiterer Teil dieser Arbeit mit der Frage, wie die zusätzliche Informationsmenge, die zur Adaption benötigt wird, sowie der Aufwand für deren Bereitstellung reduziert werden können.

Eine Möglichkeit der Adaption, bei der vom Benutzer keine

zusätzlichen Informationen benötigt werden, ist eine Selbstkalibrierung der Aktivitätserkennung. Die Validierung dieses Ansatzes wird auf zwei Datensätzen durchgeführt, bei denen sich die Signalcharakteristiken durch unterschiedliche Sensorpositionen am Körper ändern. Die Adaption der Aktivitätserkennung auf eine neue Sensorposition mit Hilfe der Selbstkalibrierung führte zu einer Verbesserung der Erkennungsrate um 33.3% bzw. 13.4% auf den beiden Datensätzen.

In einem weiteren Lernansatz werden Informationen über Erkennungsfehler, die vom Benutzer identifiziert und angegeben werden, für die Adaption genutzt. Verglichen mit der Angabe der tatsächlich ausgeführten Aktivität ist die Angabe von Erkennungsfehlern einfacher und weniger störend. Zwei verschiedene Ansätze zur Nutzung der Erkennungsfehler werden untersucht - ein erweiterter 'k-Nearest-Neighbor-Klassifizierer' sowie eine 'Reinforcement Learning' Methode. Die Adaption des Aktivitätserkennungssystems auf die spezifischen Charakteristiken eines neuen Benutzers resultierte in einer Verbesserung der Erkennungsrate um 10.3% mit dem 'k-Nearest-Neighbor-' bzw. 36% mit dem 'Reinforcement Learning' Ansatz.

Statt einer expliziten Angabe der Erkennungsfehler durch den Benutzer können diese auch indirekt über dessen Hirnströme ermittelt werden. Ein so auf einen neuen Benutzer adaptiertes Aktivitätserkennungssystem ergab eine Verbesserung der Erkennungsrate um durchschnittlich 6.8%.

Abschließend wird untersucht, wie sich eine Adaption des Aktivitätserkennungssystems auf das Verhalten des Benutzers auswirkt. Im Rahmen eines Experiments werden Handgesten des Benutzers vom Aktivitätserkennungssystem erkannt, welches in verschiedenen Durchgängen sowohl mit wie auch ohne Adaption betrieben wird. Die Nutzung des adaptiven Aktivitätserkennungssystem führt zu einer Verbesserung der Erkennungsrate. Außerdem gibt es Anzeichen dafür, dass der Benutzer bei aktivierter Adaption seine Gesten anpassen kann, so dass diese komfortabler und natürlicher ausgeführt werden können.

1

Introduction

In this chapter we motivate the use of adaptive techniques for activity recognition. We present the requirements and challenges adaptation poses on activity recognition systems. Furthermore the main research questions, as well as the outline of the thesis, are given.

1.1. Activity and gesture recognition in non-stationary environments

The type and usage of personal computing systems has changed significantly in the last decade. Miniaturized and mobile systems are superseding immobile computing terminals. In combination with advances in signal processing and sensing technology new applications have arisen which support the user anywhere and anytime, taking their location, situation, activities and even emotion, into account. This is commonly termed *context aware computing* [1]. A typical example is a mobile smart-phone with an integrated positioning system for location aware support to find local amenities. Sports computers are another typical application, providing information about physiological and movement parameters, used to improve the users workout efficiency.

In this thesis we focus on activities and gestures as contextual information. An activity aware system can, for example, monitor user's activities of daily living to automatically create a diary [2]. It can also read the user's gestures which allows for human computer interaction [3].

The crucial building block that enables activity aware computing is activity recognition. In this thesis we focus on motion based human activity recognition. This can be accomplished by finding specific patterns in human motion captured either by sensors worn on the body or placed in the user's environment. The motion based activity recognition system must be trained before use in order for it to distinguish different motion patterns. Training is usually performed initially during the design phase, making use of prerecorded annotated examples of typical activity patterns. The underlying assumption is, that the training patterns contain all possible variations which might be observed during future use of the system. These variations may include different measurement setups, different environments or different user behavior to name but a few.

Given that the activity aware system is intended to be used by anybody at any time and place, this assumption is difficult to fulfill. Human motion patterns are not stationary and can vary due to differences between users or environments. Specific user characteristics such as gender, age, height and weight can affect movement characteristics. In addition variations within the user's environment can be influential, such as altered movements due to different types of clothing. Changes in the sensor system may also occur. Sensors can be mounted

at different locations or be replaced by newer models. Such factors can affect the sensor readings and therefore the subsequent recognition. We refer to variations that occur over time and affect the recognition performance of a recognition system as non-stationarities (definition in Section 2.2.5).

Accounting for all possible non-stationarities would require a vast amount of training examples. Not only would they be expensive to acquire, but also new signal characteristics, developed in the future, cannot be captured.

Novel techniques are required to cope with non-stationarities in activity recognition. A promising direction, gaining attention not only in the field of activity recognition, is adaptive learning. The use of adaptive techniques in activity recognition poses new challenges. The learning should be guided by the user to ensure their intention is taken into account. At the same time this supervision needs to be minimized as it is deemed obtrusive. Interaction with an adaptive system also raises questions about effects on the user's behavior.

1.1.1. Adaptive learning

Capitalizing on adaptive learning techniques may result in a recognition system able to cope with non-stationarities. An adaptive recognition system is designed to learn changing user behaviors or novel environmental characteristics throughout its usage. In this thesis we focus on incremental learning [4], which allows to integrate new knowledge as soon as it becomes available, and address the resulting challenges for activity recognition.

1.1.2. Reducing supervision

In tasks, where a defined mapping between activity and input data is to be learned, supervision is essential. We see supervision as information provided by the user, in addition to any sensed motion signal. Typically incremental learning approaches require a ground truth label for each activity instance to learn. This annotation information would have to be given continuously by the user during system use. This is very obtrusive, may interfere with the target application and affect the acceptance of such adaptive activity recognition systems.

Ways to reduce supervision required, as well as making supervision less obtrusive for the user, need to be found.

1.1.3. Interaction with an adaptive recognition system

Users can adjust their behavior to meet the expectation of an activity recognition system they interact with. For example in a gesture recognition application, the user may learn to perform some gestures in a way that the system can recognize them better.

When the activity recognition is static, the user exploring different behaviors cannot influence the system's recognition parameters. In the case of adaptive activity recognition however, it is not clear how the user, that adjusts their behavior, affects the learning of the recognition system. Likewise, adaptation of the recognition system may influence the user's behavior. Understanding this co-adaptation enables us to evaluate the limitations as well as the usefulness of adaptive activity recognition.

1.2. Research questions

With the ideas and motivations given above, we address the following research questions in this thesis:

- Which methods allow for incremental adaptation of an activity recognition system, also in the case of limited supervision?
- How can we minimize the supervision required?
- Can an activity recognition system adapt without supervision?
- How can supervision provided by the user be less obtrusive and more implicit?
- How do adaptive recognition system and users both adjust their behavior to influence each other?

1.3. Thesis outline

In Chapter 2 we give definitions for typical terms used throughout the thesis. Furthermore an overview of adaptive learning strategies in activity recognition and related fields is provided.

A method for unsupervised adaptation is proposed in Chapter 3. A thorough evaluation of this approach on artificially generated data, as well as real motion data, is performed.

In Chapter 4 incremental learning in a setting with reduced supervision is presented. The focus is on the use of error feedback as additional information which is provided by the users. Two different approaches are investigated. The first is based on an incremental k-Nearest-Neighbor (kNN) classifier while the second capitalizes on reinforcement learning. Both methods are simulated in a gesture recognition scenario. The reinforcement learning approach is furthermore evaluated in an online adaptive gesture recognition case study.

A novel way of automatic error feedback generation by means of brain signals is presented in Chapter 5. For that purpose an extended dataset was recorded in a gesture recognition scenario comprising of hand motion data and brain signals. The potential of brain signals as a form of supervision for the incremental kNN learning, as presented in Chapter 4, is investigated based on the data recorded.

The influence of adaptive gesture recognition on the behavior of the user is the subject of Chapter 6. In an experiment users interact with an online gesture recognition system in both a static and an adaptive setting. We investigate differences in user behavior in the different experimental settings based on motion trajectories.

In Chapter 7 we summarize the significant achievements of this thesis alongside a discussion and give an outlook for future research.

2

Activity recognition and adaptation

In this chapter we introduce the activity recognition chain. Furthermore we present definitions for typical terms used throughout the thesis. We also give an overview of the types of non-stationarities present in related recognition applications. In addition related work in adaptive learning is presented and reviewed.

2.1. Introduction

Activity recognition is a way to automatically infer human activities from sensor readings by means of machine learning techniques [5]. A typical state of the art activity recognition system is built from three elements, as depicted in figure 2.1.

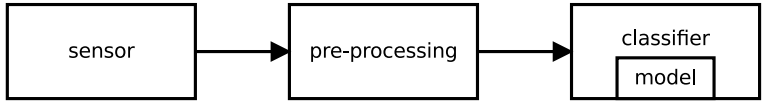


Figure 2.1: The typical activity recognition chain. Sensor readings are preprocessed and fed into a classifier which maps the signal to an activity-class based on a model.

Sensors capture information related to user activities. In a preprocessing step the sensor signal is segmented into regions of interest. Discriminative features, specific to the target activities, are extracted from these segments. The classifier assigns an activity class to each signal segment by mapping the according feature values to a given model. Besides the activity classes there may also be a NULL-class to which all signal segments, that do not contain an activity, are assigned to. In this thesis we do not deal with the NULL-class problem by making sure that each signal segment forwarded to the classifier contains an activity.

Classifier models are typically trained with signals recorded in a similar setting as the target application. This includes the type and position of sensors as well environmental properties and the user. The underlying assumption is that the characteristics captured in the training signals are also present during actual use of the activity recognition system.

Trained models may not be fully applicable if signal characteristics are non-stationary. In such cases adaptive techniques are required to fit the trained models to the new characteristics.

2.2. Definitions and terminology

In this section we give definitions and explanations for specific terms as they are used throughout this thesis.

2.2.1. Motion based human activity recognition

Activity recognition describes the inference of an agent's activities from observations related to the agent and their environment. In our case the agent is human whose motion patterns are observed. Therefore the activities we take into account are all tied to patterns measurable with motion sensors. Typical activities can be for example modes of locomotion, sports exercises or hand gestures.

In this thesis we focus on two distinct activity types: hand gestures and continuous periodic movements, for example running. Since we refer to hand gestures as a form of motion we use activity recognition and gesture recognition interchangeably throughout this thesis.

2.2.2. Segmentation

Segmentation is the process of finding regions in a continuous signal stream that contain human activities. Each resulting segment contains exactly one type of activity. If the activity has a distinct start and end, which is the case for example for a hand gesture, the length of the segment is exactly the length of this activity.

2.2.3. Feature

Features are measurable properties of the observed activity. They are selected so that they allow for a discrimination of the target activities. Usually several different features are combined in a feature vector $\mathbf{f}$. Ideally the features are independent from each other, so that each of them describes a different property. Typcially we calculate a feature vector on a signal segment containing an activity. In this case we refer to the feature vector for this segment as activity instance $\mathbf{x}$.

2.2.4. Classification

Classification is the process of assigning an observation a class membership. An observation is represented by the feature vector of an activity instance $\mathbf{x}$. The class c is one activity from a set of possible activities.

To each observation represented by $\mathbf{x}$ the most probable class c, of all possible activity classes K, is assigned.

$$c = \underset{c \in K}{\operatorname{argmax}}(P(c|\mathbf{x})) \tag{2.1}$$

The posterior probability $P(c|\mathbf{x})$ is the probability of class c given the feature vector $\mathbf{x}$. It can be derived via Bayes theorem.

$$P(c|\mathbf{x}) = \frac{P(\mathbf{x}|c)P(c)}{P(\mathbf{x})} \tag{2.2}$$

With $P(\mathbf{x}|c)$ being the conditional probability of $\mathbf{x}$ given c, $P(c)$ being the class prior, and $P(\mathbf{x})$ being the marginal probability of $\mathbf{x}$.

2.2.5. Non-stationarity

We employ the stationarity definition commonly used in the field of stochastic processes. "A stationary process is one whose distribution remains the same as time progresses" [6]. We can apply this definition to the distributions the activity classification is based on.

We consider a classification problem stationary if the posterior probability $P(c|\mathbf{x})$, the class prior $P(c)$, and the conditional probability $P(\mathbf{x}|c)$ remain the same for any time difference τ.

$$P_t(\mathbf{x}) = P_{t+\tau}(\mathbf{x}) \tag{2.3}$$

$$P_t(c) = P_{t+\tau}(c) \tag{2.4}$$

$$P_t(\mathbf{x}|c) = P_{t+\tau}(\mathbf{x}|c) \tag{2.5}$$

Accordingly a non-stationarity is present in the classification problem, if at least one of the Equations 2.3 – 2.5 is not fulfilled. This is in line with the non-stationarity definition given by Kelly et al. [7], Widmer and Kubat [8] and Yang and Zhou [9]. The cause for a non-stationaritiy is usually hidden, which means it is not know a priori, it is not explicitly sensed and not represented by predictive features [10].

Kuncheva [11] summarizes four different types of non-stationarities:

NS1 **Random noise:** The amount of noise present in the system. According to our definition this is not an actual non-stationarity since it won't affect the distributions given in Equations 2.3 – 2.5 unless the amount of noise varies over time.

NS2 **Abrupt changes:** The statistical properties of the distributions (Equations 2.3 – 2.5) change abruptly at a specific time t_1 (e.g. $P_{t_1}(\mathbf{x}) \neq P_{t_1+\tau}(\mathbf{x})$ for $\tau \to 0$).

NS3 **Drift:** A drift is a gradual change which follows a trend. If for example $P_{t+\tau}(\mathbf{x})$ is Gaussian distributed, changing the mean of the distribution μ by $\Delta\mu$ for every time step τ would be considered a drift.

$$\mu_{t+k\tau} = \mu_t + k\Delta\mu \qquad k = 1, 2, 3, \ldots \tag{2.6}$$

NS4 **Systematic trends:** This refers to the idea of recurring contexts. For example a distribution $P(\mathbf{x})$ already observed at time $t + \tau_1$ may reappear at time $t + \tau_2$ with $\tau_1 < \tau_2$, allowing the reuse of information obtained earlier.

$$P_t(\mathbf{x}) = P_{t+\tau}(\mathbf{x}), \quad \tau \leq \tau_1 \tag{2.7}$$

$$P_t(\mathbf{x}) \neq P_{t+\tau}(\mathbf{x}), \quad \tau_1 < \tau < \tau_2 \tag{2.8}$$

$$P_t(\mathbf{x}) = P_{t+\tau}(\mathbf{x}), \quad \tau \geq \tau_2 \tag{2.9}$$

In this thesis we focus on two typical sources of non-stationarities likely to occur during every day use of an activity recognition system.

In the first case the non-stationarity is caused by changes in motion capturing. A displacement of body-worn motion sensors may occur in particular when a sensor is removed in the evening and reattached the next morning for example, resulting in an abrupt change of the motion signal distribution (NS2). Also variations in the sensor position are likely, as sensors have to be comfortable to wear and cannot be fixed too tightly. A sensor that slowly slides down a limb segment for example may result in a drift in the sensed motion signal (NS3).

In the second case the non-stationarity is caused by changes in the user behavior. The user's performance of activities or gestures can vary over time resulting in abrupt changes or drifts (NS2, NS3). Furthermore the system may be taken over by a new user. A different user is likely to show a different behavior when performing gestures or activities and therefore abruptly change the motion signal distribution (NS2).

2.2.6. Stability and plasticity

In the context of adaptive learning, especially when considering life long learning, there is the issue of stability and plasticity [12].

Stability

A recognition system that continuously learns and adapts should not be affected by irrelevant inputs. For example it should not discard previously learned knowledge (e.g. about input data distributions) when irrelevant inputs (e.g. noisy signals caused by a temporary sensor failure) are observed.

Plasticity

An adaptive recognition system should integrate new knowledge as soon as it becomes available. For example the statistics of the expected input data distribution should be adapted in response to inputs that are processed.

Stability-plasticity dilemma

Carpenter and Grossberg state that "the properties of plasticity and stability are intimately related" [12]. According to them a system has to be plastic to learn from significant new inputs while at the same time remaining stable in response to irrelevant inputs.

2.3. Related work

In several recognition applications adaptive approaches have been proposed to tackle non-stationary characteristics in the input signals. In this section we give an overview of non-stationarities typical for activity recognition and related fields, and discuss adaptive techniques to cope with them.

2.3.1. Non-stationarities in recognition tasks

Non-stationarities are a challenge in many machine learning applications and recognition tasks. In data mining the term concept drift is utilized [8]. Typical causes for a non-stationarity are for example a new user of an activity recognition system [13] or the change of illumination in image recognition [14].

The type of non-stationarities and their effect on the statistical properties of the input signal can be manifold and is application dependent. In a natural and continuous environment gradual drifts as

well as abrupt changes are expected (e.g. seasonal, demographic, habitual) [11]. In the following we list recognition applications together with typical reasons for the occurrence of non-stationarities.

- **Brain-computer interfaces:** Brain signals can contain drifts within a recording session as well as abrupt changes between different recording sessions for the same user. Varying background brain activities are one cause for these drifts and changes [15, 16, 17, 18].
- **Handwriting recognition:** Individuals have different writing styles. Therefore abrupt changes in the shape of the characters may occur when a handwriting recognition system gets used by a different user [19, 20, 21, 22].
- **Speech recognition:** Vocal characteristics vary between speakers leading to abrupt changes in the signal statistics when the speaker changes [23, 24, 25, 26]. Other sources of non-stationarities within this application are abrupt or gradual changes in background noise (e.g. when moving to a different location) or variations in channel characteristics (e.g. when changing the recording device) [27].
- **Activity recognition:** The change to a new user can lead to abrupt changes in the sensor signal statistics [28]. Furthermore users may vary their behavior for example due to psycho-motor learning [29], aging [30, 31], injuries or illnesses [32], which may result in gradual or abrupt signal changes. The position and characteristics of sensors can also influence the stationarity of the signal readings [33, 34].

2.3.2. Adaptive learning and classification

Adaptive learning techniques tackle the challenge of non-stationarities in recognition tasks by adapting the classifier model to changed data distributions. We can distinguish between two different learning strategies, batch learning and incremental learning.

Batch and incremental learning

In batch learning a new classifier model is trained on a batch of data. This batch contains the most recent data available reflecting the latest

statistical properties of the data distribution [8]. This learning principle can be applied in combination with any standard classification algorithm like a Support Vector Machine (SVM) [35] or Bayes Classifier [36]. A drawback of this approach is that statistics obtained from a previous batch are completely discarded every time a new batch is learned. This includes any knowledge that was previously learned and which might still be partly valid. It may also take time to collect a new batch of data large enough to train a classifier model with the new statistics. A change detection method may be necessary to indicate when retraining a new classifier model is required [37, 11, 38].

With incremental learning strategies the classifier model continuously adapts to match the statistics from the incoming data [10]. In contrast to batch learning, it is not necessary to collect a certain amount of data for training. Instead each incoming data instance is used to incrementally learn the classifier model. Some learning and classification methods support incremental learning by nature. Examples of those are the k-Nearest-Neighbor (kNN) [36] classifier, and methods based on vector quantization like the Self Organizing Map (SOM) [39] and the Neural Gas Network (NGN) [40]. Other classification algorithms have been extended to allow for incremental learning, e.g. the support vector machine [41] or hidden Markov models (HMM) [42, 43, 44, 45, 46].

Both strategies, incremental and batch learning, have to forget the outdated knowledge when learning new characteristics [8]. In batch mode this can be accomplished by setting a batch size which contains enough data to reflect the current statistical properties of the input distributions, without containing any data from outdated input distributions. This allows the amount of data used from the past to be adjusted and subsequently used for training the new model [47]. Incremental learning algorithms usually implement a learning rate parameter. It regulates how much information captured in the model is updated by the newly acquired data instance. The learning rate is a parameter in adaptive learning that is strongly related to the stability-plasticity dilemma (Section 2.2.6). A high learning rate will allow for a rapid adaptation to new input data statistics while a lower learning rate makes a system more stable.

Learning and supervision

Aside from different learning strategies, learning algorithms can also be distinguished by the amount of supervision they require. In supervised

learning the ground truth label is available for every data instance. This is typical for initial training of a classifier based on an annotated training dataset. Furthermore in some data mining applications it is assumed that ground truth labels become available at some point after the classification of each instance [8]. Widely used learning and classification methods like Naive Bayes, kNN or SVM rely on supervised learning [36].

In unsupervised learning no ground truth information is available for the training data. In this case learning is based on finding structures in the data. This is typical for clustering algorithms, which assign similar data instances the same cluster membership [48]. The SOM or the NGN also represent structures in the input data without an explicit class assignment [39, 40]. The SOM and the NGN also have been extended for learning in non-stationary environments [49, 50].

In between the extremes of unsupervised and supervised learning lies semi-supervised learning [51, 52, 53]. In this case labels are only available for some instances in the training data. The classifier model is trained based on the labeled examples supported by structures found in the training data. Semi-supervised approaches are usually not capable of incremental learning.

In Reinforcement Learning (RL) [54] the training of a classifier is not based on ground truth labels but on rewards. For each decision made by the classifier a reward is given. Rewards do not directly contain information about the ground truth labels but instead provide a measure of how good the decision was. The classifier model is adapted based on past experiences, so that the expected reward for future decisions is maximized.

2.3.3. Adaptation in activity recognition and related fields

Several adaptive approaches have been proposed for different recognition tasks to tackle non-stationarities.

Handwriting recognition

Incremental prototype adaptation has been proposed to adapt a handwriting recognition system to the writing style of a certain user [55]. Letter prototypes are adapted using letter instances collected during system use. The labels for the letter instances can be collected in a self-supervised manner, where the prediction of the classifier is reused as the ground truth label [56, 57]. It is also possible to calibrate a

generic classifier model towards a certain user prior to system use [58]. In this case labeled data has to be collected for system calibration.

Brain-computer interfaces

For brain-computer interface (BCI) applications a calibration step is proposed before each session. Incremental classifier model adaptation has been effective for such purposes [15, 17, 59, 60, 61]. In this case a supervised approach is used requiring the collection of labeled examples during system use. This can be difficult depending on the BCI application.

Speech recognition

Adaptation by model selection is widely used in speech recognition [62, 23, 63]. Several models are trained based on different speaker characteristics. During system use the best fitting model for the current speaker is chosen at runtime. In addition the calibration of a generic speaker model, with small amounts of user specific training data, was proposed [64]. The collection of calibration data can either be supervised [26] or self-supervised based on the generic speaker model [25].

Activity recognition

A widely used approach to cope with different users in activity recognition applications is to train a model based on large datasets recorded from a high number of subjects [28, 13, 34]. This is supposed to result in a user independent classifier which is valid for any future subject. It has been shown though, that the performance of such user independent classifiers often leads to a lower performance compared to user specific classifiers [28, 13, 34]. The calibration of a generic classifier model towards a specific user has also been proposed for activity recognition. In this case either labeled user-specific data is collected to calibrate the system [65] or a general bio-mechanics model is adjusted based on the physiological features of the new user [33, 66].

To address the variations associated with sensors mounted on different body locations a model selection approach has been used [67]. For each specific body location a classifier model is trained and chosen according to the detected sensor location. Lester et al. [34] trained a classifier on data recorded from several different sensor locations on the user's body. This is supposed to result in a classifier model, valid

for all possible sensor locations. Kunze et al. [68] proposed the use of an additional sensor modality, namely a gyroscope, to compensate for the impact that a changed sensor location has on the readings of an acceleration sensor.

2.3.4. Evaluating adaptive recognition systems and co-adaptation

"Evaluating systems is a difficult task, and it becomes even more difficult when the system is adaptive" [69]. Typically adaptive recognition systems are evaluated empirically in the following manner [56, 57, 58, 18, 61, 59, 60, 62, 26, 25, 28, 13]:

- A dataset containing data typical for the application is recorded or taken from a repository.
- The recognition system is simulated using the dataset - once with the adaptive method under evaluation and once with a non-adaptive baseline method.
- A performance metric is calculated from the simulation results. Usually the recognition accuracy or the recognition error rate are used as performance metric.
- The performances of the adaptive and the non-adaptive recognition are compared to evaluate the benefit of the adaptation.

Such empirical evaluation strategies are, amongst others, also common for user-adaptive human-computer interfaces (HCI) [70]. The evaluation focus is solely on the adaptive system. Interactions between the user and the adaptive system are only investigated from the system's point of view. Even the collection of user's opinions (e.g. through questionnaires or user interviews) is usually targeted only at getting information about the success of the adaptation compared to the non-adaptive case [71]. A change of user behavior, caused by the adaptive nature of the system, is not evaluated.

Mackay claims that people and technology co-adapt, when people interpret and adapt technology, which in return influences the behavior of people [72, 73]. Such a co-adaptation has for example been observed for BCIs [74] or a robotic prosthetic hand controlled via electromyography (EMG) signals [75]. A co-adaptation between user and system may also exist when using an adaptive activity recognition system. To

our knowledge co-adaptation for adaptive activity recognition has not been studied yet.

3

Unsupervised classifier self-calibration

In this chapter we propose a method for classifier adaptation through unsupervised self-calibration. We investigate the basic behavior of this approach, based on an artifical dataset, in order to understand it's potential and limitations. The applicability of the calibration method to activity and gesture recognition is validated on two activity recognition datasets.

3.1. Introduction

We propose a classifier self-calibration strategy for adapting an activity recognition system to non-stationarities caused by the displacement of acceleration sensors on one limb segment.

With an unsupervised approach continuous supervision by the user is not required. The adaptation is guided by characteristics contained in the input data.

We use classifiers capable of incremental learning to adapt to variations in the statistics of the input signal, especially drifts or displacements of class distributions in the feature space. The signal of a displaced sensor will show a similar distribution shape compared to the original sensor position. The mean and potentially also the variance are likely to change though.

The activity recognition system is initially trained on pre-recorded data. When a sensor displacement occurs resulting in a reduced recognition performance the calibration adapts the classifier to the changed characteristics.

We apply the self-calibration method to one synthetic dataset in addition to two activity recognition datasets, namely HCI gestures and aerobic fitness activities. We use mean and variance features in a two dimensional feature space together with a nearest centroid classifier (NCC) as a way to simplify visualization and analysis of the systems's behavior.

3.2. Method description

Our method integrates in the typical activity recognition chain as a self-calibration extension as depicted in Figure 3.1. The system can operate in two settings: normal operation where the self-calibration is disabled, and calibration mode with active self-calibration. As long as the sensor remains exactly in the position used during training no calibration is necessary. If sensor displacement leads to reduced classification performance, the calibration mode is activated ($calib\_start\_cond = 1$), adapting the classifier to the changed situation. When a predefined stop condition is fulfilled ($calib\_stop\_cond = 1$) the calibration is stopped and the classifier continues in normal operation.

In calibration mode the classifier classifies the incoming feature vectors $\mathbf{x}$. The predicted class $\tilde{c}$ is used as the class label c for incremental learning. We term c a self-label since it is generated by the classifier

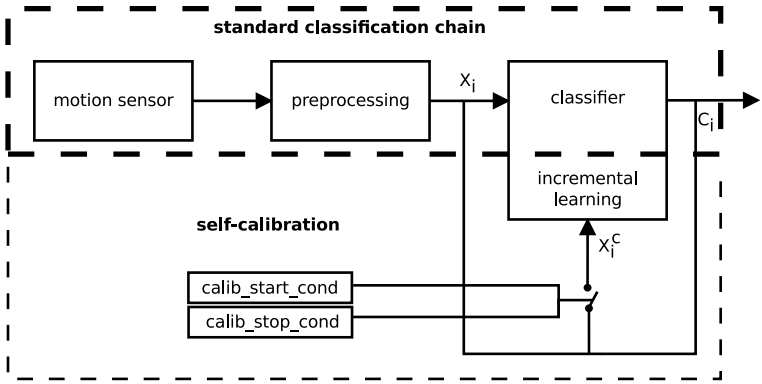


Figure 3.1: The classification chain with additional self-calibration. This approach differs from the standard classification chain as self-labeled samples $\mathbf{x}_i^c$ can be fed into the incremental learning mechanism of the classifier.

itself. The self-calibration module receives the feature vectors $\mathbf{x}_i$ as well as the related self-label c_i from the classifier. In the calibration process these self-labeled samples $\mathbf{x}_i^c$ are used to adapt the model of the classifier. The combination of feature vectors with labels provided by the classifier enables the use of supervised incremental learning strategies available for several classification algorithms.

We use a NCC classifier which allows for the observation of the underlying model, namely the class centers in feature space. This assists in explaining the adaption process behavior. For incremental learning with the NCC classifier we adopt the learning rule typically used in vector quantization techniques like the SOM [39] or the growing neural gas (GNG) [76]. This learning strategy consists of the following steps:

- A data sample $\mathbf{x}_i$ is presented and classified to class c_i by the NCC classifier
- The class center CC_i^c of class c within the NCC model at time i is moved towards the presented self-labeled sample $\mathbf{x}_i^c$ yielding a new class center CC_{i+1}^c.

The amount of learning is regulated by a constant learning rate (LR) which adjusts by how much the class center CC_i^c is moved towards $\mathbf{x}$.

The learning rule is:

$$CC_{i+1}^{c} = (1 - LR) \cdot CC_{i}^{c} + LR \cdot \mathbf{x}_{i}^{c} \tag{3.1}$$

The self-calibration start condition is triggered when the sensor displacement happens and subsequently the performance of the classifier decreases. This is comparable to a user noticing degrading system performance and triggering a classifier self-calibration.

The stopping criterion we use for the synthetic dataset is based on the gradient of the Euclidean distance $d_{0,i}(CC_0^c, CC_i^c)$ between the class center at time i and the class center before adaption at time 0. We stop the calibration when the absolute mean of the gradient over the last n calibration steps is below a preset threshold ST.

$$|\frac{1}{n}\sum \nabla(d_{(0,i-n)}, d_{(0,i-n+1)}, d_{(0,i-n+2)}, \ldots, d_{(0,i)}))| \leq ST \tag{3.2}$$

In the beginning of the adaptation process distance $d(.)$ increases rapidly, resulting in a high gradient. Once the adapted class center is close to the real class center this distance only changes slightly. In this case the gradient is small and below ST, therefore the calibration is stopped.

In our validation on the two real-world activity datasets (Sections 3.4 and 3.5) we are limited in the number of data instances we can use for the calibration. For these simulations we do not use the gradient based stop criterion and stop the calibration when all available instances have been presented.

3.3. Characterization on synthetic dataset

To characterize the self-calibration we perform several simulations based on a two class problem with Gaussian distributions in a two dimensional feature space. We are interested in the accuracy increase by the calibration in case of a displaced sensor, and therefore displaced class centers. The self-calibration is characterized for different distribution overlaps reflected by different maximum accuracies an optimal classifier can reach on this dataset.

Figure 3.2 shows the synthetic dataset in the feature space and the NCC classifier adaptation process. The two 2D Gaussian distributions of class A and B both have a variance of 1. The distributions shown reflect the new situation, for example after a sensor displacement. Their

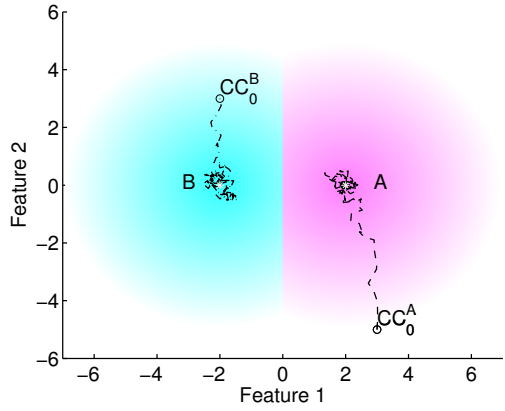


Figure 3.2: We show one specific example of how the adaptation of the class centers can look like. The clouds represent the Gaussian distributions for the classes A and B. The lines are the adaptation paths from the learned class centers CC_0^A and CC_0^B of the initial distribution (initial sensor position) towards the new class centers of the new distribution (displaced sensor).

overlap can be varied by changing the distance between the distribution centers. Changing the overlap allows us to vary the maximum accuracy an optimal classifier can achieve on these distributions. Points CC_0^A and CC_0^B in this feature space represent the ideal NCC model before the change (sensor displacement). The figure demonstrates the paths of the initially trained class centers moving towards the center of the distributions during the calibration. This represents one simulation run. Each simulation run consists of the following steps:

1. **Initialization:**
    - Set dataset distribution overlap for an accuracy between 60-100% of an optimal NCC classifier.
    - Randomly initialize the initial class centers CC_0^A and CC_0^B.
    - Calculate accuracy before calibration based on the selected distribution means and the class centers CC_0^A and CC_0^B.
2. **Self-calibration:**
    - Set $i = 0$

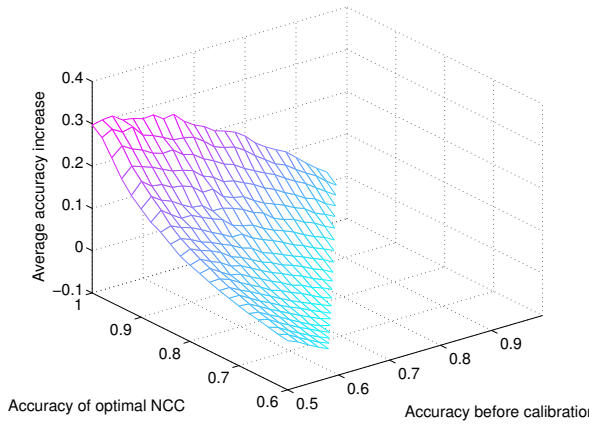


Figure 3.3: Average increase in accuracy for the simulated self-calibration. We varied the distance between the initial distributions A and B to change the accuracy an optimal NCC can reach. Furthermore the new (displaced) dsitributions were placed randamly in the feature space, with the same distance as initially, to simulate different accuracies before adaptation.

- Draw an instance $\mathbf{x}_i$ randomly from the distributions and classify it to class c_i according to CC_i^A and CC_i^B.
- Update CC_i^A and CC_i^B according to the learning rule (Equation 3.1) with the learning rate set to $LR = 0.01$, and set $i = i + 1$.
- Repeat the previous two steps until the calibration stop criterion (Equation 3.2) with $n = 10$ and $ST = 0.005$ is met.

3. **Evaluation**

   - Calculate accuracy after calibration based on the selected distribution means and the final class centers CC_i^A and CC_i^B.
   - Calculate accuracy increase = (accuracy after calibration) - (accuracy before calibration)

In Figure 3.3 the average increase in accuracy is plotted for different accuracies before calibration and different distribution overlaps result-

ing in different optimal accuracies. The results are averages over 10,000 simulation runs. When accuracy before calibration is higher than 50%, meaning better than guessing in our two class problem, there is in average an increase in accuracy for the calibrated classifier. The higher the accuracy before calibration is, the lower the room for improvement and therefore the increase in accuracy. The lower the distribution overlap, and the higher the accuracy an optimal classifier can achieve, the higher the expected increase in accuracy. For high distribution overlaps, and a close to optimal initial classifier, the expected improvement by the self-calibration is slightly negative. It is a surprising result that with an imperfect self-labeling by the initial classifier a better calibrated classifier can be achieved.

In an actual activity recognition scenario more classes have to be distinguished and their distributions are usually not Gaussian. We investigate this case in the following sections.

3.4. Validation on HCI scenario

We characterize our approach on a gesture based human computer interface scenario. The activity classes are arm gestures describing five geometric structures: a triangle, an upside-down triangle, a circle, a square and an infinity symbol.

As the focus is on sensor displacement we aim to minimize the variability induced by the subject. In order to achieve this we cut out a template of each shape from a Styrofoam plate. The subject must move their hand alongside these templates, effectively guiding their movements.

In order to capture the gestures we attached six USB acceleration sensors to the right forearm of the subject. We roughly aligned them to minimize rotational variation. The sensors were all calibrated and verified before the experiment. For each of the five gestures we recorded 50 repetitions. The duration of each recorded gesture is between five and eight seconds.

Several typical features were extracted from the acceleration signal. We chose to use only mean and variance features calculated on the acceleration signal y-axis. The two dimensional feature set provides sufficient discriminative power and allows for good visualization of the feature space. This is helpful to analyze the calibration mechanism.

Figure 3.4: HCI scenario, left image: Six sensors are attached to the subjects right forearm. Fitness scenario, right image: Ten sensors are attached to the subject's left leg - five on the thigh and five on the lower leg.

3.4.1. Without self-calibration

In Figure 3.5 we show the results of training the NCC classifier on one sensor position u and testing on all sensor positions v. Training and testing on the same sensor position ($v = u$) result in the highest accuracies, with an 84.9% average. Testing on the direct neighboring sensors of u ($|v - u| = 1$) results in an accuracy decline to 50.0%. If we test on sensor positions which are even further apart ($|v - u| > 1$), accuracy decreases to 48.7%.

The classifiers trained on sensor positions $u = 1$ and $u = 2$ perform better on the more distant sensor positions $v = 4$ and $v = 5$ than on the closer sensor position $v = 3$. The same behavior can be observed for sensor positions $u = 4$ and $u = 5$ with respect to sensor position $v = 6$. This can be explained by a slight rotation of sensors 3 and 6 compared to other sensors, despite the alignment we performed. As we operate only on the y-axis of the accelerometer, and not on the magnitude of all three axes, the measurements are affected by sensor rotation. This effect is shown in Figure 3.6 where we show the feature spaces for sensor positions 2, 3 and 4.

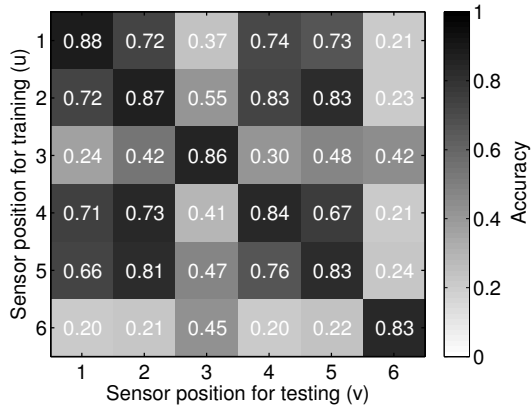


Figure 3.5: Accuracies for training a classifier on sensor position u and testing on position v

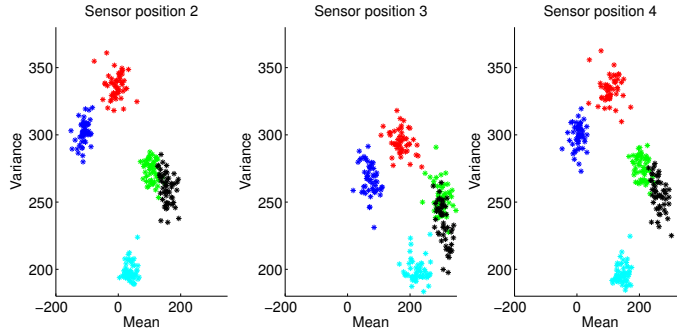


Figure 3.6: Feature spaces for the sensor positions 2, 3 and 4. The feature spaces of sensor positions 2 and 4 are more similar compared to positions 2 and 3. This explains the differences in the classification accuracies when training on position 2 and testing on positions 3 and 4.

3.4.2. With self-calibration

We calibrate the classifiers trained on the data from sensor position u by using a subset of the data obtained from sensor position v. After the calibration the classifier is tested on the remainder of the data obtained

	$v = u$		$\lvert v - u \rvert = 1$		$\lvert v - u \rvert > 1$	
	mean	std	mean	std	mean	std
HCI dataset:						
before calib.	84.9	2.1	50.0	21.0	48.7	24.4
after calib.	82.4	2.0	63.5	19.8	59.4	22.5
rel. imp.	-2.9	1.2	33.3	25.6	31.1	30.2
Full fitness dataset:						
before calib.	83.0	5.7	65.7	4.1	42.0	9.1
after calib.	82.8	5.9	74.4	9.9	49.5	9.4
rel. imp.	-0.2	1.8	13.4	14.8	20.5	23.1
Reduced fitness dataset:						
before calib.	95.1	3.4	89.4	4.8	67.3	9.5
after calib.	95.4	3.6	95.8	3.6	69.8	10.8
rel. imp.	0.4	4.0	7.2	5.1	4.1	12.6

Table 3.1: Accuracies (in %) before calibration, after calibration and the relative improvement by the calibration. The improvement is relative to the accuracy before calibration. v denotes the sensor position before, and u after the change.

from sensor position v. A three fold cross calibration is applied, using two folds to calibrate the classifier and one fold to test it on the new sensor position. The data samples used for the calibration are selected randomly from all classes and are not presented in any specific order.

The calibration results of all sensor displacement combinations and folds are plotted in Figure 3.7. The accuracies of the classifiers after the calibration are plotted against the accuracies of the initial classifiers. The points above the diagonal line are the cases where the calibration improved the classifier. The points below the line are the cases where the calibration deteriorated the classifier. For sensor positions where the accuracy before calibration is already $> 80\%$ (e.g. $v = u$) the calibration is only beneficial for half of the cases. This is according to our simulation results on the synthetic dataset where an already close to optimal classifier was likely to be worsened by the calibration. When the class centers CC_0^c in the NCC model are already optimal it is likely

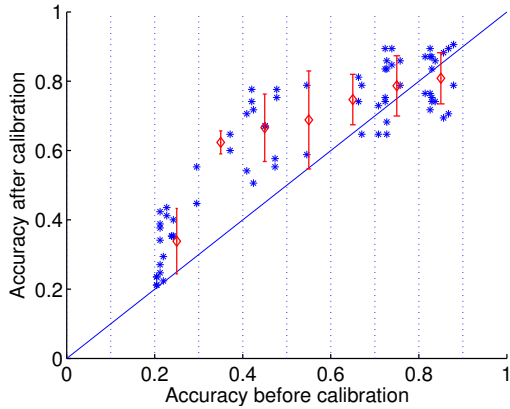


Figure 3.7: This plot shows the accuracies before calibration against the accuracies after calibration for the HCI dataset. Each * marks one sensor displacement combination, for example training on $u = 1$ and testing on $v = 2$. With six sensor positions 72 sensor displacement combinations are possible, including calibration on the undisplaced sensor. The vertical bars show mean and std in the according bins.

that the calibration slightly changes them, resulting in a worse classification. The cases where the calibration is beneficial outranges the cases where it is not. A summary of the results is listed in Table 3.1.

3.5. Validation on fitness scenario

In addition to the HCI scenario we validate our approach on an aerobic fitness scenario.

We recorded the acceleration of the left leg for six different typical aerobic movements shown in Figure 3.8. For this purpose we placed ten bluetooth acceleration sensors at the subject's leg, five on the lower leg and five on the thigh, as depicted in figure 3.4. We placed the sensors equidistantly and with the same orientation as we focus on translational displacement. The sensors were all calibrated and verified before the experiment.

For the recordings an experienced subject copied the movements an aerobic instructor performed in a video. The video, containing all six

classes in equal shares, had a duration of 4:22 minutes and was repeated five times.

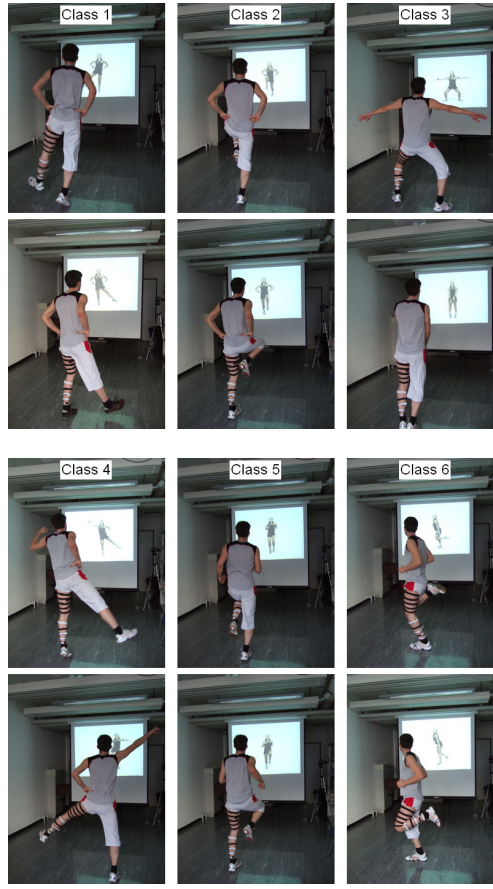


Figure 3.8: The fitness scenario includes 6 classes: (1) flick kicks, (2) knee lifts, (3) jumping jacks, (4) superman jumps, (5) high knee runs, (6) feet back runs. For each class, the extent of the body movements is shown on two example pictures.

For the data of each sensor we calculat the acceleration magnitude and extract mean and variance features based on a sliding window of eight seconds with two thirds overlap. The resulting two dimensional feature space for each sensor position is depicted in Figure 3.9. We

observe that directly adjacent sensor positions show less difference in feature space than sensor positions which are further apart. There is also less similarity between sensor positions 5 and 6, despite being adjacent, as they are on different limb segments. For the positions (4,5,7-10) we can observe overlaps between the classes "flick kicks" and "high knee runs". There are also overlaps between the classes "jumping jacks" and "sumperman jumps" for all sensor positions.

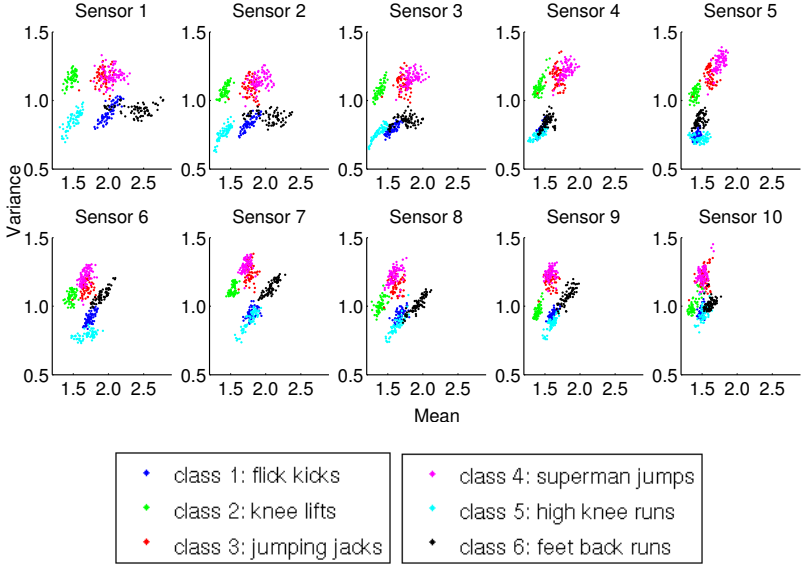


Figure 3.9: Feature space of the fitness dataset for each of the 10 sensors

The feature spaces of sensors placed on different limb segments are too different for succesful calibration. Therefore we focus solely on sensor displacements on the same limb segment. Since the results we obtained for the lower leg and the thigh are similar, we will detail only sensor displacements on the lower leg.

We train on sensor position v and test on all five sensors positions u in order to simulate a sensor displacement on the lower leg. The accuracies obtained when training a classifier on one sensor location and testing it on the same ($v = u$) is in average 83.0%. If we test on the direct neighboring sensors $|v - u| = 1$ the average accuracy drops to 65.7%. If we test on sensor positions which are even further apart

($|v - u| > 1$) the accuracy of the classifiers trained on u decreases to 42.0%.

We apply the classifier self-calibration described in Section 3.2 to the displaced sensors in a similar manner as described in Section 3.4. The accuracies after calibration versus the accuracies before calibration are plotted in Figure 3.10 for all sensor combinations and folds.

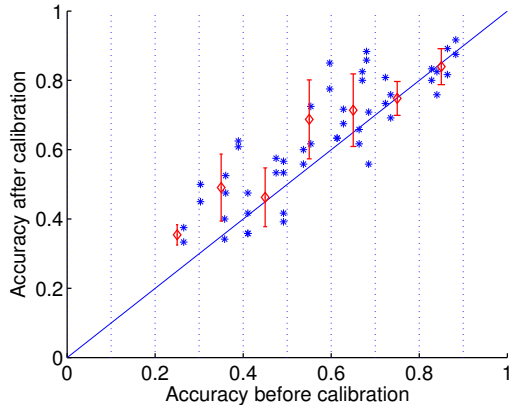


Figure 3.10: This plot shows the accuracies before calibration against the accuracies after calibration for the fitness dataset. Each * marks one sensor displacement combination, for example training on $u = 1$ and testing on $v = 2$. With five sensor positions 50 sensor displacement combinations are possible, including calibration on the undisplaced sensor. The vertical bars show mean and std in the according bins.

The calibration of classifiers operated on displaced sensor positions on the lower leg is beneficial in most of the cases. Classifiers with a close to optimal initial accuracy (in this case above 70%) are less likely to be improved through self-calibration.

The simulations in Section 3.3 show that the expected improvement of the self-calibration increases when there is a better class separation and therefore a higher accuracy of an optimal classifier. To validate this finding we remove the two overlapping classes (1 and 3) from the fitness dataset and evaluate the self-calibration on the resulting reduced fittness dataset.

In Figure 3.11 we show two examples of the calibration dynamics. In the first case we apply the calibration to the fitness dataset. For class

4 the calibrated class centers do not end up close to the optimal class center. This is a typical case where less class separation confuses the self-calibration and only one class center, here that of class 3, benefits. For the reduced fitness dataset the class center of class 4 ends up closer to the optimal class center.

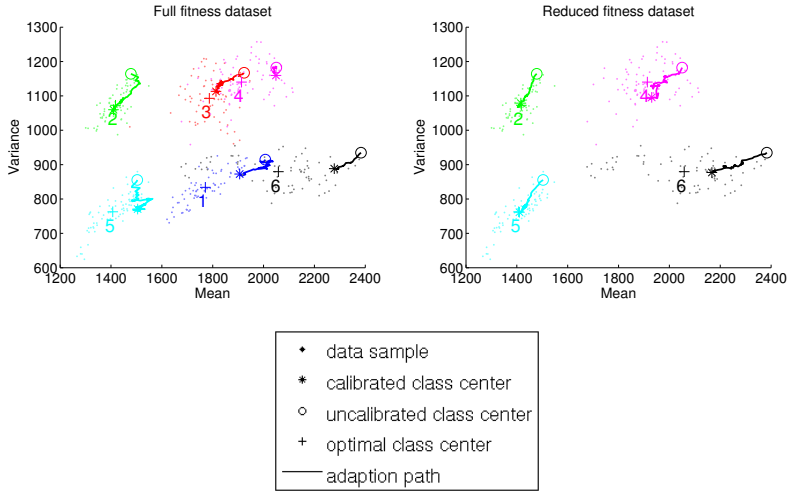


Figure 3.11: Adaptation paths of the class centers during calibration on the full (top) and the reduced (bottom) fitness dataset. The NCC classifier is initially trained on sensor position 1 and calibrated on sensor position 2. The numbers in the plots indicate the different classes aligned to Figure 3.9.

For both, the reduced and the full fitness datasets, we see that some class centers (e.g. class 6) end up quite distant from the optimal class centers even though their paths seem to lead directly to the optimum. This indicates that we did not use enough instances for the calibration to reach the optimal class centers.

The accuracies obtained when training a classifier on one body location and testing it on the same location ($v = u$) reach an average of 95.1% with the reduced dataset. If we test on the direct neighboring sensors $|v - u| = 1$ the average accuracy drops to 89.4%. If we test on sensor positions which are even more distant ($|v - u| > 1$) the average accuracy of the classifiers decreases to 67.3%. In the case of the reduced

dataset the sensor displacement has less influence on the accuracies of classifiers working on displaced sensors, compared to the full dataset.

The accuracies after calibration against the accuracies before calibration, for the reduced fitness dataset, are plotted in Figure 3.12. In this case the classifiers with an accuracy before calibration greater than 80% are likely to improve to an accuracy after calibration of greater than 90%. The classifiers with an accuracy before calibration of less than 80% are less likely to be improved by the self-calibration. The ones which benefit from the calibration improve only to 75%. This is because the classifier improves for three of the four classes, sacrificing one class. A detailed summary of all results is given in Table 3.1.

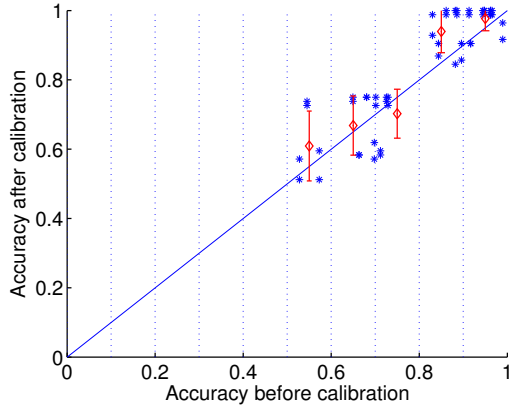


Figure 3.12: This plot shows the accuracies before calibration against the accuracies after calibration for the reduced fitness dataset. Each * marks one sensor displacement combination, for example training on $u = 1$ and testing on $v = 2$. With five sensor positions 50 sensor displacement combinations are possible, including calibration on the undisplaced sensor. The vertical bars show mean and std in the according bins.

3.6. Discussion

The unsupervised classifier self-calibration method presented in this chapter allows for the adaptation of a classifier to a new sensor position. The validation on two scenarios, namely HCI gesture recognition

and fitness aerobic activity recognition, has shown improvements in the classification accuracy by 33.3% and 13.4%, in case of a slightly displaced sensor ($|v - u| = 1$).

The validation is based on mean and variance features which are affected by sensor displacement. Other features might be less position dependent and reduce the influence of sensor displacement.

Currently the calibration procedure has to be started when there is a decrease in classification performance, e.g. from sensor displacement. The displacement itself does not have to be explicitly detected. In many applications the calibration start may be a manual trigger by the user. There could also be a first time use calibration mechanism, for example after attachment to the body. Change detection algorithms [77, 78] may eventually trigger calibration automatically.

From our experiments on synthetic and real-world datasets we have seen that our approach is more effective when there is minimal confusion between classes. The calibration is likely to worsen the classifiers in cases where there is a high confusion between classes. The amount of sensor displacement also has an influence on the calibration. In many cases the calibration of less displaced sensors leads to better results compared to highly displaced sensors. This is related to the class overlap mentioned. In particular for different body segments the self-calibration was not beneficial due to highly different class distributions in the feature space. It also has to be noted that the calibration in average leads to an improved classification. However this improvement can not be guaranteed for individual cases.

Our approach operates as an add-on to the typical classification chain and should not be restricted to the NCC classification algorithm, so long as incremental learning is supported. For example a k-Nearest-Neighbor classifier could potentially be used instead of the NCC since it supports incremental learning by nature. Other classification methods, like the Support Vector Machine or Hidden Markov Models have also been extended for incremental learning [41, 42, 43]. The behavior of these methods in combination with the self-calibration has yet to be evaluated though. In addition, our approach is potentially applicable to different sensor modalities or other sources of variability.

4

Classifier adaptation based on error feedback

In this chapter we present two approaches for classifier adaptation based on error feedback. The first is an improved incremental kNN classifier with two learning modes, one for error- and one for correct-learning. The second approach is based on RL where the error feedback is translated into a reward signal. Both approaches are evaluated on a real world gesture recognition dataset. Furthermore we validate our findings for the RL approach in an adaptive online gesture recognition case study.

4.1. Introduction

The unsupervised self-calibration method described in Chapter 3 does not take the user's intention into account. Therefore it can not be guaranteed that the adapted classifier performs better, according to the user's expectation. In this chapter we address this drawback and focus on a specific type of supervision, namely error feedback, given by the user. We investigate the learning and adaptation towards a new user within a gesture recognition scenario.

We assume that the users can become aware of the system's recognition result for each input gesture, directly after the gesture is performed. They can therefore compare their intended gesture command with the recognition results, and identify recognition errors.

A feedback given by the user about recognition errors can be translated into a *correct/error* (CE) teacher signal, indicating for each gesture or activity if the recognition result was correct or wrong. If the user indicates an error after the recognition of a certain instance, an *error* signal is generated. If no error is indicated a *correct* signal is generated.

The user can indicate recognition errors to the system via a user→system feedback channel. In our case the feedback input can be generated by a simple button, pushed whenever an error is perceived. Compared to ground truth feedback this is less obtrusive since the actual class label doesn't have to be communicated. This is especially true in applications where many different classes have to be distinguished, as more sophisticated input mechanisms, e.g. a full keyboard, would be required.

Another advantage of the error feedback is that interaction is only required from the user in the *error* case. When no error feedback is given by the user it is assumed that the recognition was *correct*. Once the system has adapted and reached reasonable performance only little interaction will be required from the user.

There are also other possibilities to capture the error information from the user without explicit interaction. Implicit interaction could occur through physiological signals like heart rate, electrodermal activity or brain waves. It has been shown that changes in the heart rate or the electrodermal activity of a person are linked to stress and emotions [79, 80, 81]. Stress symptoms or specific emotions could be triggered particularly when using a system that makes errors and does not react in the way the user expects it. In electroencephalography (EEG) signals that capture brain activity, typical patterns linked to errors have been

identified. These patterns have been observed both for users making the errors, as well as for users affected by a system making errors [82]. In Chapter 5 we investigate this option in more detail.

4.2. Incremental kNN using CE signal for classifier adaptation

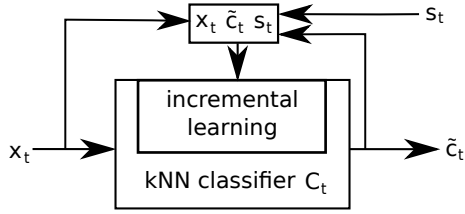


Figure 4.1: An instance $\mathbf{x}_t$ is classified by C_t to prediction $\tilde{c}_t$. The instance $\mathbf{x}_t$ together with the prediction $\tilde{c}_t$ and the teacher signal s_t are used for incremental learning.

In Figure 4.1 we show the principle of our teacher based kNN learning. An instance $\mathbf{x}_t$ is fed into the classifier C_t and classified to prediction $\tilde{c}_t$. For each classification result we get the according teacher signal s_t. The instance $\mathbf{x}_t$ together with the prediction $\tilde{c}_t$ and the teacher signal s_t are used for incremental learning the classifier C_t, resulting in an adapted classifier C_{t+1}.

The typically incremental learning kNN classifier requires ground truth supervision for learning. In this section we extend such a classifier to support learning with a supervising CE teacher signal s_t and evaluate it on a gesture recognition dataset.

We assume the following learning scenario. Initially the recognition system is trained on prerecorded training data from different potential users. After this initialization phase the system is deployed to a new, so far unseen user for operation (no data of this new user is contained in the training data). It is assumed that this new user will behave in a similar way as captured in the training data. The recognition system is therefore assumed to have a performance better than guessing, but lower than if it was specifically trained for this user [13]. To adapt the pre-trained classifier to the new user we perform incremental learning guided by a CE teacher signal.

4.2.1. Incremental learning kNN classifier

We use a weighted kNN classifier [83] where a weight is assigned to every data point in the classifier model. The kNN classifier C_t valid at time t consists of a model $M_{c,t}$ for each class c.

$$C_t = \{M_{1,t}, M_{2,t}, M_{3,t}, ...\} \tag{4.1}$$

Each model $M_{c,t}$ consists of data points which are represented by tuples mapping a feature vector $\mathbf{f}_i$ of weight $w_{i,t}$ to class c.

$$M_{c,t} = \{(\mathbf{f}_1, w_{1,t}, c), (\mathbf{f}_2, w_{2,t}, c), (\mathbf{f}_3, w_{3,t}, c), ...\} \tag{4.2}$$

When training the classifier in non-incremental mode, all weights ($w_{i,t=0}$) of the added feature vector $\mathbf{f}_i$ are set to 1.

For the classification of new instances $\mathbf{x}_t$ the weights ($w_{i,t}$) of the k nearest neighbors (NN), belonging to the same class, are summed up to the sum of weights (SW_c).

$$SW_c = \sum_i w_{i,t} \quad \forall \quad \text{NN}_i \text{ of class c} \tag{4.3}$$

The resulting prediction is

$$\tilde{c} = \underset{c}{\text{argmax}}(SW_c) \tag{4.4}$$

We extend the kNN learning to support learning from the CE teacher signal s_t. For each labeled training instance this signal has one of the following two states:

- *correct*: We assume the class prediction $\tilde{c}_t$ of this instance is correct and corresponds to the ground truth
- *error*: We assume the class prediction $\tilde{c}_t$ of this instance is incorrect and differs from the ground truth

Our kNN learning strategy comprises of two learning processes - one for instances $\mathbf{x}_t$ where $s_t = \textit{correct}$ and one for instances $\mathbf{x}_t$ where $s_t = \textit{error}$.

To learn a *correct* instance $\mathbf{x}_t$ the feature vector $\mathbf{f}$ of this instance is added to the classifier model together with a weight $w_{i,t} = 1$ and the class label $\tilde{c}_t$. The weights of the l neighboring data points, which are of the same class as the learned instance, are increased according

Algorithm 1 *correct* learning of instance $\mathbf{x}_t$ with label $\tilde{c}_t$

add $(\mathbf{x}_t, 1, \tilde{c}_t)$ to $M_{c,t}$
find the l NN of $\mathbf{f_t}$ within C_t
for each found NN j **do**
  if $c_j = \tilde{c}_t$ **then**
    $w_{j,t+1} = W_{inc}(w_{j,t})$
  end if
end for

to the weight adjustment function W_{inc}. The *correct* learning strategy is outlined in Algorithm 1.

When learning from *error* instances the actual ground truth class is not known. It is known however, that the ground truth is different from the predicted class label $\tilde{c}_t$. We exploit this information in our *error* learning strategy. We find the l data points in the classifier model, which are neighbors of the instance $\mathbf{x}_t$ to learn. The weights of the neighboring points, which are of the *error* class $\tilde{c}_t$, are decreased according to the weight adjustment function W_{dec}. Data points with a weight below a fixed removal threshold κ are removed from the classifier models, as they contribute only marginally to the classification result, but increase the computational complexity. The *error* learning strategy is outlined in Algorithm 2.

Algorithm 2 *error* learning of instance $\mathbf{x}_t$ with label $\tilde{c}_t$

find the l NN of $\mathbf{x_t}$ within C_t
for each found NN j **do**
  if $c_j = \tilde{c}_t$ **then**
    $w_{j,t+1} = W_{dec}(w_{j,t})$
  end if
end for
for each $(\mathbf{f_m}, w_{m,t}, c)$ in C_t **do**
  if $w_{m,t} < \kappa$ **then**
    remove $(\mathbf{f_m}, w_{m,t}, c)$ from C_t
  end if
end for

The weight adjustment functions (see figure 4.2) were chosen as follows:

$$W_{inc}(w_{i,t}) = -\frac{(w_{i,t}-2)^2}{2} + 2; \qquad W_{dec}(w_{i,t}) = \frac{w_{i,t}^2}{2} \tag{4.5}$$

The concave and convex shape control the weight increase and decrease. Weights are limited ($w_{i,t} \in [0, 2]$) to prevent single instances

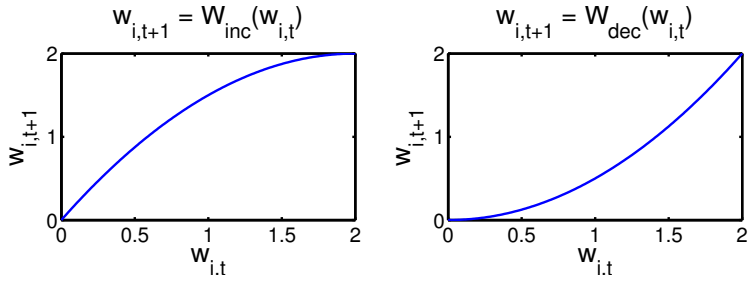


Figure 4.2: Weight adjustment functions W_{inc} and W_{dec}, used to increase and decrease the weights of the neighboring instances.

biasing the classification. The quadradic functions affect low weight values ($w_{i,t} < 1$) more than high weight values ($w_{i,t} > 1$). When the classifier has been operated in a stationary condition the weights in the classifier model will be high ($w_{i,t} \sim 2$). In this case the classifier model is more stable compared to a situation where the weights are small ($w_{i,t} << 2$), but will also take longer to adapt to a non-stationarity.

Due to the incremental learning the number of data points in the classifier models are not constant. On one hand this could lead to continuous growth of the number of data points and increasing memory requirements. On the other hand continuous removal of data points could lead to a classifier model containing no information. We subsequently add several extensions to the kNN classifier to cope with the variable number of data points in the model during learning and classification.

To prevent a bias towards one class in the classification the sum of weights over all data points within the models $M_{c,t}$ of class c should be similar. An equalization mechanism stops learning, when the sum of weights of class c ($\sum_w M_{c,t}$) differs by $\alpha \in [0, 1]$ from the mean between the highest ($max_K(\sum_w M_{K,t})$) and the lowest ($min_K(\sum_w M_{K,t})$) sum of weights of all classes K.

The *correct* learning for a specific class c is stopped if the following condition is fulfilled:

$$\sum_w M_{c,t} > \frac{1}{2}(max_K(\sum_w M_{K,t}) + min_K(\sum_w M_{K,t})) \cdot (1 + \alpha) \quad (4.6)$$

The *error* learning for a specific class c is stopped if the following

condition is fulfilled:

$$\sum_w M_{c,t} < \frac{1}{2}(max_K(\sum_w M_{K,t}) + min_K(\sum_w M_{K,t})) \cdot (1-\alpha) \quad (4.7)$$

We also stop the *error* learning for class c when the number of data points present in its model ($\#M_{c,t}$) reaches an absolute lower limit δ:

$$\#M_{c,t} \leq \delta \quad (4.8)$$

The number k of NN to take into account for the classification of new instances is defining the neighborhood radius within the feature space. With a constant k this radius changes when the number of data points in the classifier model changes. The radius is kept constant by adapting k at every time step depending on the number of data points in the classifier model. We define k as the number of points being the fraction $R \in [0,1]$ of all data points in the classifier model:

$$k_t = R \cdot \#C_t \quad (4.9)$$

With $\#C_t$ denoting the number of all data points within the classifier model at time t.

Similarly the radius for learning, defined by l, depends on the number of points in the classifier model. To keep the learning radius constant we define a learning rate $LR \in [0,1]$ from which l is calculated at every time step as follows:

$$l_t = LR \cdot \#C_t \quad (4.10)$$

4.2.2. Characterization on artificial dataset

We characterize our incremental kNN learning based on simulations using an artificial dataset. In an initial training phase, all training instances $\mathbf{x}$ from an initial training set are added to the kNN classifier model, with weight $w_{i,0} = 1$. Each new instance $\mathbf{x}_t$ is then classified and the prediction $\tilde{c}_t$ together with the teacher signal s_t are used for learning. We generate the CE teacher signal s_t by comparing the result of the online classification to the ground truth. If the prediction $\tilde{c}_t$ matches ground truth, then $s_t = \textit{correct}$ - otherwise $s_t = \textit{error}$ is generated.

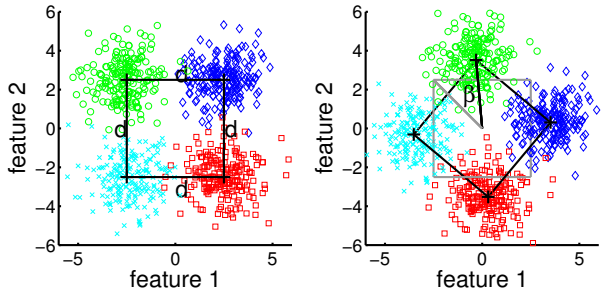


Figure 4.3: Artificial dataset consisting of four classes. Each class is represented by a two dimensional Gaussian distribution with the centers placed at the corners of a square. Left plot: the class overlap and therefore the optimally achievable classification accuracy can be adjusted by changing the class distance d. Right plot: The distributions can be rotated by an angle β to simulate a non-stationarity.

Artificial dataset generation

The artificial dataset we generate (Figure 4.3) allows us to characterize the behavior of our learning algorithm in case of a non-stationarity in the input data. It consists of four classes, each represented by a two dimensional Gaussian distribution with a diagonal covariance matrix $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The distribution centers for each class are placed at the edges of a square with side length d. This distance between classes is used to adjust the class overlap, and therefore the maximum accuracy an optimal classifier can achieve. To simulate a non-stationarity as it could appear e.g. at a sensor position change, we introduce an angle β by which the dataset can be rotated.

The distributions are sampled with an equal number of samples per class.

We generate three datasets with different parameters. The *initial training set* is used to train the initial classifier $C_{t=0}$. A non-stationarity is simulated by a rotation of the data by angle β_{adapt} in the *adaptation set*. The performance of the adapted classifier C_t is evaluated on the *adaptation test set*. The parameters d and β are changed according to the intended analysis. A summary of the generated datasets is provided in Table 4.1

dataset	d	β in °	# samples
initial train set	d_{sim}	0	100
adaptation set	d_{sim}	β_{adapt}	1000
adaptation test set	d_{sim}	β_{adapt}	800

Table 4.1: Summary of the generated artificial datasets with the according parameters.

Influence of initial accuracy

In our learning scenario according to Figure 4.1 each instance $\mathbf{x}_t$ is classified to prediction $\tilde{c}_t$ before it is used for incremental learning. If the prediction is according to the ground truth the teacher signal will be *correct* and *correct* learning will be applied (Algorithm 1). In case of a wrong prediction the teacher signal will be *error* and *error* learning will be applied (Algorithm 2). The performance of the classifier C_t therefore has a direct influence on the classifier learning.

For our simulations we empirically select the following learning and classification parameters relevant for Algorithms 1 and 2, and Equations 4.6 - 4.10:

$$R = 0.05 \quad \kappa = 0.1 \quad \alpha = 0.1 \quad LR = 0.05 \quad \delta = 13$$

In Figure 4.4 we show the adaptation of the classifier for β_{adapt} of 30°, 50° and 70°. We chose $d_{sim} = 3.95$ which results in a good separation between classes. The initial accuracies of classifier $C_{t=0}$ tested on the *adaptation test sets* are 75.0% (30°), 36.9% (50°) and 11.1% (70°) respectively. These are the accuracies before the adaptation is started. The simulation of the upper-bound accuracy (ub acc) the adapted classifier can reach resulted in 94.3% (determined by training a kNN classifier on the *adaptation set* and testing it on the *adaptation test set*).

In all three cases of β_{adapt} the adapted classifiers reach the upper-bound accuracy. The lower the initial accuracy for $C_{t=0}$, the longer the adaptation takes. During the adaptation phase, when the classifier is still improving, the number of data points in the model, as well as the weights, remain nearly constant. This indicates that the increase of weights, and the addition of new data points, are at similar levels as the decrease of weights and the removal of data points. When the performance of the adapted classifier gets close to the upper-bound

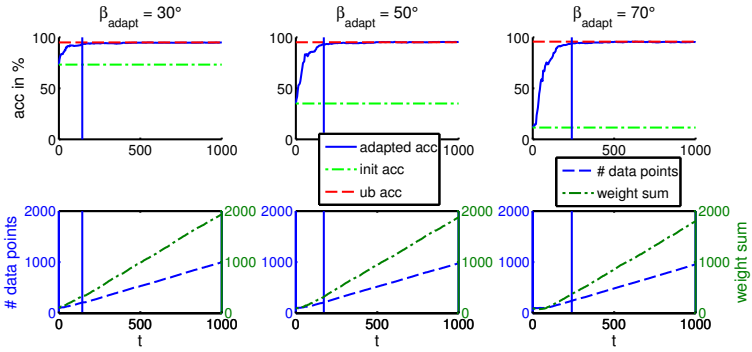


Figure 4.4: Top Row: Accuracy increase of the adapted classifier over sample index t for $d_{sim} = 3.95$ and three different β_{adapt}. The estimates of the initial accuracy (init acc) and the upper-bound accuracy (ub acc) are given as baselines. The indexes where the performance of the adapted classifier reaches 90% of the upper-bound accuracy are marked with a vertical line. The lower the initial accuracy the longer the learning takes to reach 90% of the upper-bound accuracy. Bottom Row: Number of data points in the kNN model as well as the sum of weights ($\sum_w M_{c,t}$) combined for all classes c over index t. Both values remain nearly constant until the performance of the adapted classifier is close to the upper-bound accuracy. Afterwards the number of points, as well as the sum of weights, increase, because the *correct* learning dominates.

accuracy the *correct* learning dominates and the number of data points in the model, as well as their weights, increase.

In Figure 4.5 we show the simulation results when the class separation is inferior ($d_{sim} = 1.85$). The initial accuracies in this case are 56.3% (30°), 40.8% (50°) and 27.3% (70°) respectively. The upper-bound accuracy in this case is 64.3%. The general behavior is similar to the one shown in Figure 4.4, with the adaptation being faster when the initial accuracy is higher. However, the increase rate of data points and weights in the model is lower. This is caused by the lower upper-bound accuracy which makes *correct* and *error* learning more even as more *error* instances are generated.

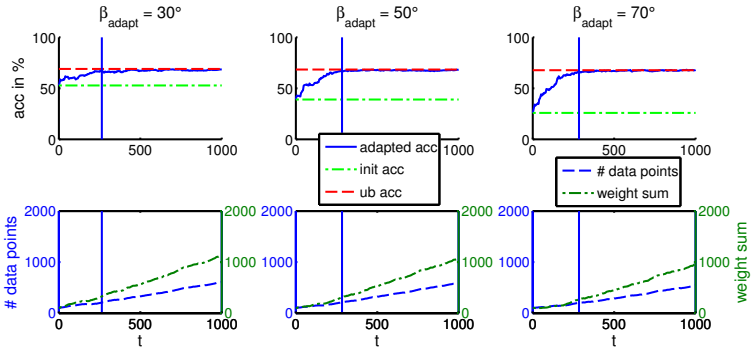


Figure 4.5: Similar to Figure 4.4 we show the results for $d_{sim} = 1.85$. The lower the initial accuracy, the longer the adaptation takes to reach the upper-bound accuracy. The increase in the number of data points and the sum of weights is slower compared to Figure 4.4 as more *error* learning is present.

Influence of learning rate

The learning rate is an important parameter in any adaptive learning application. A high learning rate allows for a fast adaptation to a non-stationarity but comes at the cost of a higher sensitivity to noise. A low learning rate is makes the learning more robust to noise but also reduces plasticity. In the following we investigate the effect of the learning rate LR (see Equation 4.10) on our incremental learning kNN approach.

In figure 4.6 we show the adaptation of the classifier for three different LR values, namely 0.005, 0.1 and 0.4. The other learning parameters are:

$$R = 0.05 \quad \kappa = 0.1 \quad \alpha = 0.1 \quad \delta = 13$$

The simulation datasets are configured with $d_{sim} = 3.85$ and $\beta_{adapt} = 70°$.

A higher learning rate, e.g. $LR = 0.1$ compared to $LR = 0.005$, leads to a faster increase of the weights. This leads to a faster learning so that the upper-bound accuracy is reached earlier. The change of the number of data points in the classifier model is also affected by the learning rate. With a higher LR the weights of more data points are adjusted.

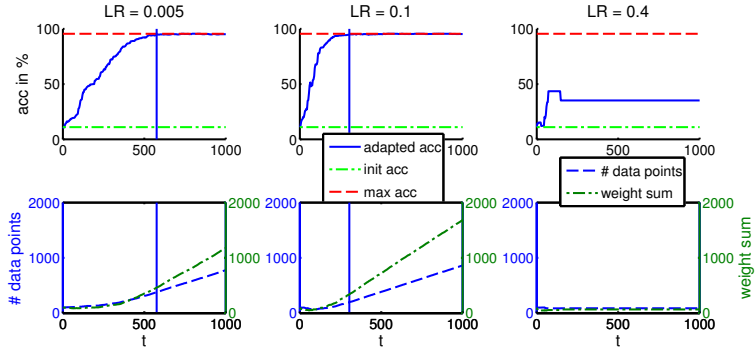


Figure 4.6: The plots illustrate the effect of the learning rate LR on the adaptation. The dataset is configured with $d_{sim} = 3.85$ and $\beta_{adapt} = 70°$. A higher learning rate LR makes the classifier more plastic and allows for faster learning (compare $LR = 0.005$ and $LR = 0.1$). This is also reflected by a stronger increase in the sum of weights when the learning rate is higher. Faster learning comes at a cost of stability though, as a too high learning rate leads to catastrophic forgetting (see $LR = 0.4$).

A high learning rate, e.g. $LR = 0.4$ in this case, results in a poor adaptation, with the adapted accuracy not reaching the upper-bound. This is due to a decrease of the number of data points in the classifier model due to *error* learning. This may eventually lead to the loss of one class, which can be seen as catastrophic forgetting. The inability to recognize one class leads to only *error* learning for this class and therefore a continuous removal of points until the balancing mechanism (Equation 4.8) stops the learning. In such a case the learning is never reactivated since no data points can be added for this class.

Our simulations have shown that the stability of the learning is not only affected by the learning rate but also depends on the initial accuracy, the number of points in the classifier model and the parameters of the equalization method. The higher the initial accuracy is, the less *error* learning is performed which reduces the risk of catastrophic forgetting. The more points a classifier model has, the more have to be forgotten before the result is catastrophic. This leads to a higher stability. The parameters of the equalization method (Equations 4.6, 4.7 and 4.8) regulate when the learning is stopped. The more conserva-

tively these parameters are chosen (i.e. the more balanced the number of data points for each class is, with large δ and small α), the more stable the learning is. The downside of this however, is that it may delay the learning.

Comparison to learning from ground truth

We compare our learning approach combining *correct* and *error* learning from a CE teacher with *correct* learning from a ground truth teacher. In Figure 4.7 we show the according learning curves for $\beta_{adapt} = 50°$ with the following learning parameters:

$$R = 0.08 \quad \kappa = 0.1 \quad \alpha = 0.02 \quad LR = 0.01 \quad \delta = 15$$

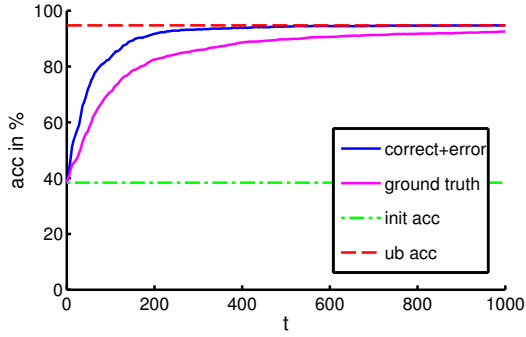


Figure 4.7: Comparison between learning from the CE teacher with *correct* and *error* learning and the ground truth teacher with only *correct* learning on the artificial dataset with $\beta_{adapt} = 50°$. Both variants converge to the estimated upper bound. The learning from the CE teacher allows for a faster convergence compared to the ground truth learning.

The combination of *correct* and *error* learning based on the CE teacher outperforms the *correct* only learning with ground truth in terms of learning speed. This can be explained through the *error* learning, which allows the active removal of those data points from the model, leading to misclassifications. Removal of data points with the *correct* only learning is not possible.

4.2.3. Validation on gesture recognition dataset

We validate our approach on a hand gesture dataset (see Section 5.3 for details of the recording setup). We simulate a new-user scenario, where the pretrained gesture recognition system is adapted to a new user. The dataset consists of five different hand gestures ($c = 1..5$) performed by seven subjects in a human computer interface scenario. The recognition is based on gesture instances ($\mathbf{x}_t$) containing six features calculated from the segmented hand acceleration (refer to Section 5.5.1 for details on the dataset preprocessing).

We generate an *initial train set* containing 480 gestures selected randomly from six of the seven subjects, equally distributed over all classes. The *adaptation set* and the *adaptation test set* are generated from the left out subject containing 2240 and 500 gestures respectively.

The simulations are performed in the same way as described in section 4.2.2. Each simulation is repeated 20 times with different dataset permutations to eliminate the influence of a specific instance order. The learning parameters were empirically chosen as follows:

$$R = 0.02 \quad \kappa = 0.1 \quad \alpha = 0.4 \quad LR = 0.002 \quad \delta = 48$$

The results, averaged over all subjects and all dataset permutations, are shown in Figure 4.8. In this case the incremental learning based on the CE teacher performs worse than the learning from ground truth. The estimated upper bound accuracy of 87.2% is not reached. This could be caused by certain classes being less good recognizable, and therefore also harder to learn since less *correct* instances are available for these classes. Nevertheless the accuracy is increased by 14.4% over the subject independent baseline of 72.1%, reaching 82.5%.

4.2.4. Discussion

With the presented incremental online kNN learning method it is possible to adapt a pretrained classifier to new input data characteristics based on a CE teacher. The validation on a real world gesture recognition dataset has shown the effectiveness of our approach. In average the adaptation through online learning from the CE teacher increased the accuracy by 10.3% over the subject independent baseline of 68.3%.

It has to be noted though, that our method is sensitive to the order of the input samples. Incrementally learning the classifier with a different order of the same instances will result in a slightly different classifier.

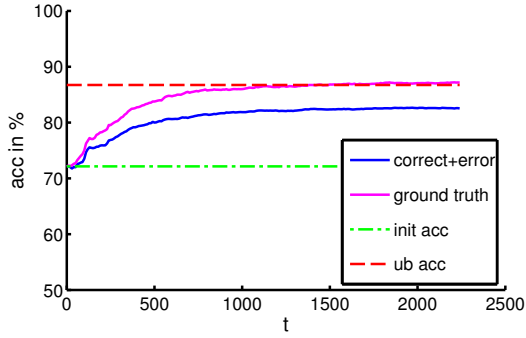


Figure 4.8: Learning from a CE teacher and from ground truth on the real world gesture dataset. With learning from the CE teacher the accuracy is improved by 14.4% over the initial subject independent accuracy. The increase is less compared to learning from ground truth and the estimated upper bound is not reached.

The number of data points in the classifier model continuously increases during the learning when the teacher provides more *correct* than *error* signals. This is likely to happen in recognition tasks with good class separation and therefore a high upper-bound accuracy. A mechanism which limits the *correct* learning, when enough data points are in the classifier model, would reduce the memory requirements, and make our approach applicable for life-long learning.

We propose quadratic weight adjustment functions designed to limit the weights to the range $[0, 2]$. This ensures a good stability-plasticity tradeoff and also showed good results on the dataset used. Other weight adjustment functions, e.g. exponential or linear, or altering the possible range of the weights, can change the learning behavior.

The teacher signal we simulated for the evaluation of our approach is free of errors. An imperfect teacher signal will affect the learning speed and the stability of the approach. In applications with a potentially imperfect teacher this should be taken into consideration.

4.3. Reinforcing a recognition system using negative rewards

RL is a machine learning principle based on rewards received from the environment. It is a bio-inspired approach related to the learning behavior of animals [54]. In this section the typical RL terminology is used which we map to our context in the following way:

- **State (S):** Activity instance represented by a feature vector calculated from motion signals
- **Agent:** Activity recognition system
- **Action (a):** Output action related to the recognized activity class
- **Reward (r):** RL specific supervision signal, for example extracted from error feedback

For a given state S_t the agent chooses an action a_t according to policy π which results in a reward r_{t+1} from the environment. This principle is depicted in Figure 4.9. In the course of time the agent explores possible actions for different observed states and receives the according rewards. Once the agent has gained some experience by exploring several state action pairs (SAP), it can optimize policy π to choose the action with the highest expected reward for a given state.

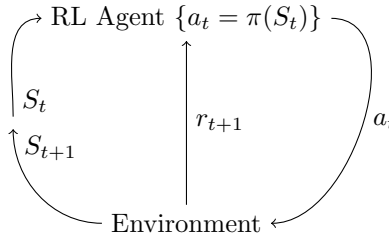


Figure 4.9: Principle of RL. The agent receives a state S_t from the environment and issues action a_t according to policy π. For the state action combination it receives a reward r_{t+1} and refines the policy accordingly. [54]

The CE signal generated from the user's feedback can be interpreted as a reward signal r to a RL agent. In this section we investigate the

learning and adaptation of a RL gesture recognition system, based on CE feedback.

4.3.1. Reinforcement learning background

In RL the agent performs an action a_t according to a given state S_t, with the goal of reward maximization. The function that performs the state action mapping is the so called policy π [54]:

$$a_t = \pi(S_t) \tag{4.11}$$

It is assumed that a new state S_{t+1} only depends on state S_t and action a_t. The environment the agent interacts with has to therefore fulfill the Markov property [84].

The policy π controls the selection of the best action for a given state, according to the Q-value. The Q-value is a measure for the expected reward of a state-action pair (SAP). It is calculated based on the Q-function, also known as Bellman equation [85], for a given SAP (S, a) and a policy π.

$$Q^{\pi}(S, a) = E\{r_{t+1} + \gamma R^{\pi}_{t+1}\} \tag{4.12}$$

The discount factor $\gamma \in [0, 1]$ is used to reduce the influence of future rewards R^{π}_{t+1}. In many applications it is assumed that the reward r_{t+1}, that directly follows a SAP, is more directly linked to the quality of the last action and therefore should be weighted higher then the following rewards.

The Q-function that results in the highest output for a SAP given all possible policies π is considered the optimal Q-function $Q^*(S, a)$.

$$Q^*(S, a) = \max_{\pi} Q^{\pi}(S, a) \tag{4.13}$$

The policy that is based on the optimal Q-function is the optimal policy π^*. It always selects the action with the maximum Q-value and is therefore called the *greedy policy*.

$$\pi^*(S) \in \arg\max_a Q^*(S, a) \tag{4.14}$$

To learn the policy for a given task there are two different procedures. In on-policy learning the RL system updates the policy incrementally with each new SAP that is experienced. The off-policy learning strategy consist of two stages. Initially a fixed preliminary policy is

applied (e.g. random action selection) to capture SAPs. Secondly these SAPs are used to learn a new policy which is then applied for actual use of the system.

Another important aspect regarding the policy π is the exploitation vs. exploration trade off. A good policy should be exploited as much as possible to maximize the reward. Nevertheless exploration of the state-action space is important during learning of the policy to find a globally optimal solution for the state-action mapping. In the following we describe three typical approaches to allow for exploration in the policy.

- **ϵ-greedy policy** [54]: With a probability of ϵ_t a random action is selected over the action with the highest Q-value for the SAP.

$$a_t \leftarrow \begin{cases} a = \arg\max_{\bar{a}} Q_t(S_t, \bar{a}) & p(\text{exploit}) = 1 - \varepsilon_t, \\ a = \text{random action in } \mathcal{A} & p(\text{explore}) = \varepsilon_t \end{cases} \tag{4.15}$$

  For learning a state-action mapping in a static environment ϵ_t can be decreased over time with e.g. $\epsilon_t = \frac{1}{t}$. This ensures high exploration in the beginning of the learning process. Finally exploitation dominates the policy, when the system has successfully learned a Q-function. In non-stationary environments, where the state-action space is subject to changes over time, a constant exploration factor can be set in order to explore novel behaviors.

- **Boltzmann exploitation** [54]: In this exploration scheme probabilities are calculated for each possible action a_t, given a state S_t. The action a_t is chosen randomly according to the probabilities $p(a|S_t)$ of all possible actions a given S_t.

$$p(a|S_t) = \frac{e^{Q_t(a,S_t)}/\tau_t}{\sum_{b=1}^{N} e^{Q_t(b,S_t)/\tau_t}} \tag{4.16}$$

  N is the total number of possible actions and $\tau_t \in \Re_0^+$ is the exploration rate. The higher τ_t the more exploration is performed. Similar to the ϵ-greedy policy, the exploration factor τ_t can be decreased over time to allow for more exploration in the early learning phase and more exploitation thereafter.

- **Optimistic initial value** [54]: With an optimistic initialization of the Q-function several possible actions are explored before the Q-function converges. Even with a greedy policy this leads to a fair amount of exploration. Since the amount of exploration becomes less with the system learning, the optimistic initialization is best used for stationary environments.

4.3.2. State of the art

One of the earliest applications where RL was successfully applied was Backgammon. The TD Gammon [86] RL approach is based on a neural network and discrete states, actions and rewards. In 1995, when TD Gammon was proposed, it was superior to other artificial intelligence players. In contrast to later RL methods this approach does not use a $Q(S, a)$ function to evaluate the expected reward for a SAP. Instead only the quality of the state is evaluated.

The Attention-Gated RL (AGRL) [87] is also based on a neural network, but in contrast to TD Gammon uses a $Q(S, a)$ function to estimate the expected reward for a SAP.

The SARSA [88, 54] RL approach has already been widely used and applied to applications like automatic spoken dialogue strategy optimization [89], adaptive music generation [90], and channel control in cellular networks [91]. This approach is named after the inputs of the Q-function update rule: state (S_t), action (a_t), reward r_{t+1}, state (S_{t+1}) and action (a_{t+1}). The Q-function is usually represented by a table of SAPs and therefore requires a finite number of discrete states and actions. An extended version of SARSA is the Least-Squares Policy Iteration (LSPI) [84]. In contrast to the traditional SARSA the Q-function is not represented by a table but modeled by basis functions which span the state space. This allows for a continuous state space representation. To find the right basis functions a priori knowledge of the state space is required. Furthermore, the state space has to remain constant over time.

An alternative state space representation is persued by Santamaria et al. In their Instance Based RL approach (INST)[92] new SAPs are stored as individual instances in the state space. This allows for continuous state space representation. Q-values for a new SAP are estimated by interpolation from stored SAPs close to the new SAP in the state space.

Q-Learning [93, 94] is a RL principle for off-policy learning. It is

based on a similar Q-value update rule as SARSA. As one main difference Q-value updates for a certain SAP are based on estimates for all possible next SAPs. This is necessary since future SAPs are not observable in off-policy learning

An alternative off-policy RL method is PEGASUS [95], a policy search algorithm. Based on a model of the environment, it searches for an optimal policy to be used later for the actual control tasks. This approach has been applied to autonomous helicopter flying [96].

4.3.3. Requirements posed by the learning scenario

The approaches just explained all have different properties. We focus on a gesture recognition scenario, similar to the one presented in Section 4.2.3, where the gesture recognition system is adapted to a new user. The learning is based on the same CE teacher signal generated from user feedback as described in Section 4.1. In the following we list the requirements our scenario poses on a RL method for gesture recognition:

- **Arbitrary continuous state space**
  The gestures captured are represented by a multidimensional feature vector of continuous real values. The characteristics of the state space are not known a priori and can vary depending on the environmental characteristics (e.g. the user behavior or sensor placement).
- **Discrete action space**
  For every state one discrete action is chosen.
- **Discrete reward** The feedback given by the user is binary, either the last action was *correct* or an *error*. This binary feedback is translated into a discrete reward.
- $Q(S, a)$**-function**
  A reward is always given for a SAP and does not solely depend on the state. Therefore a reward estimation is required, that takes into account both the state and the action.
- **On-policy**
  We are aiming at a system which continuously adapts and improves within the application. This requires on-policy RL, so that new knowledge can be incrementally integrated while the system is used.

In Table 4.2 we compare the RL approaches listed in Section 4.3.2 and investigate their applicability to our scenario based on the requirements posed.

Algorithm	Requirements				
	arbitrary continuous state space	discrete action space	$Q(S,a)$-function	discrete reward	on-policy
TD Gammon	x	x		x	x
AGRL	x	x	x	x	x
REINFORCE	x		x	x	x
SARSA		x	x	x	x
LSPI		x	x	x	x
INST	x	x	x	x	x
Q-Learning	x	x	x	x	
PEGASUS	x	x	x	x	

Table 4.2: Reinforcement learning algorithms overview. "x" indicates, which requirement is met by the algorithm

AGRL and INST both fulfill our requirements. AGRL is based on a neural network trained by back-propagation. This is known to require a high number of instance presentations and therefore a large amount of training data [97, 98]. As it is desirable within our target application to learn from a small number of instances, we focus for our further evaluation on INST, which is not limited by that.

4.3.4. Instance Based Reinforcement Learning method description

The instance based RL method we propose for our learning scenario is based on work by Santamaria et al. [92]. Q-values of possible actions a_t for an observed state s_t are interpolated from previously observed SAP. The observed SAPs are combined with their corresponding Q-value and

stored in the memory C as cases $c_i(S_i, a_i, Q_i)$. The Q-values of possible actions a_t for a new state S_t are interpolated from the cases c_i in the neighborhood NN_t of S_t. Neighboring cases are those which states S_i are within a predefined radius τ_k around S_t.

$$NN_t = \{c_i \in C | d(S_t, S_i) \leq \tau_k\} \tag{4.17}$$

As a distance measure $d(S_t, S_i)$ we use the Euclidean distance.

The estimate of a Q-value for a SAP $\hat{Q}(S_t, a_t)$ is derived by an interpolation based on the neighborhood NN_t using a kernel function $K(d(S_t, S_i)$:

$$\hat{Q}(S_t, a_t) = \sum_{\forall c_i(a_i = a_t) \in NN_t} \frac{K(d(S_i, S_t))}{\sum\limits_{c_j(a_i = a_t) \in NN_t} K(d(S_j, S_t))} Q_i \tag{4.18}$$

We make use of a Gaussian kernel function which has already shown good results in combination with the INST approach [92].

$$K(d) = exp(\frac{d}{\tau_k}) \tag{4.19}$$

As soon as the reward r_t for an observed SAP becomes available the Q-values of the cases in the memory are updated according to the following rule.

$$Q_i = Q_i + LRe_i(r_{t+1} - \hat{Q}(S_t, a_t)) \quad \forall c_i \in NN_t \tag{4.20}$$

Here $LR \in [0, 1]$ is the learning rate and e_i is a distance based weight parameter. The closer the state S_i of a case c_i is to the observed state S_t, the more c_i is affected by the update.

$$e_i = \frac{K(d(S_i, S_t))}{\sum\limits_{c_j(a_i = a_t) \in NN_t} K(d(S_j, S_t))} \tag{4.21}$$

The reward r_{t+1} is generated from the CE signal so that erroneous actions are punished while good actions are rewarded.

$$r = \begin{cases} 1 & \text{chosen action is } \textit{correct} \\ -1 & \text{chosen action is } \textit{error} \end{cases} \tag{4.22}$$

A new case is added to the memory when the distance between the newly observed state S_t and the nearest case c_i, with the same

action, is higher then a preset minimum distance τ_d. This ensures good coverage of the state space while at the same time limiting memory requirements.

In Algorithm 3 we give a summary of the learning and Q-value update for this approach.

Algorithm 3 INST with ε-greedy policy

1: **for all** time steps t **do**
2: observe state S_t
3: estimate $\hat{Q}(S_t, a_t)$ according to equ. 4.18
4: $a_t \leftarrow \begin{cases} a = \arg\max_{\bar{a}}(\hat{Q}(S_t, a_t)) & \text{exploit (prob. } 1-\varepsilon) \\ a \text{ random action} \in \mathcal{A} & \text{explore (prob. } \varepsilon) \end{cases}$
5: apply a_t and receive reward r_{t+1} from the user
6: update Q-values according to equ. 4.20
7: **if** $\min(d(S_i, S_t)) \geq \tau_d$ for all NN_i with action a_t **then**
8: add new case $c(S_t, a_t, Q_{init})$ to memory
9: **end if**
10: **end for**

4.3.5. Evaluation of RL methods on gesture dataset

We evaluate the INST RL approach on the same dataset as we did with the incremental kNN (see Section 4.2.3). We simulate the CE teacher by comparing for each input gesture the resulting action (predicted class) to the ground truth. If the action matches the ground truth *correct*, otherwise *error* is generated.

We consider two distinct learning cases:

LC1 **Learning from scratch:** A user starts using the recognition system which is randomly initialized and is not capable of recognizing the gestures yet. While the system is used and rewarded it learns the correct mapping from sensor data to gesture recognition output.

LC2 **Learning after user change:** The system is used by user A and learns the gestures of this user from scratch. At a later point the system is given to user B. Since user B is likely to perform the gestures slightly different, when compared with user A, the recognition performance drops. Therefore the system has to adapt, to match the behavior of the new user B, to continue with optimal

performance. The adaptation is based on the same reinforcement learning principle as in LC1.

The dataset is split in a training set (1155 gestures) and an evaluation set (100 gestures) for each subject. To simulate LC1, one gesture instance of the training set is presented to the RL gesture recognition system at each time step. The resulting action is compared to the ground truth and an error signal is generated if the action and the ground truth differ. The error signal is translated into a reward and fed back to the RL system.

At each time step the RL system is evaluated based on the evaluation set. This leads to an accuracy value at regular intervals, which allows to track the learning of the gesture recognition system.

The simulation of LC2 is analog to LC1. At first the learning and evaluation of the system is based on the dataset of Subject A. Afterwards the training gestures of Subject B are presented to the system. When training from gestures of Subject A the system is also tested on gestures from Subject A. This process is identical for Subject B.

For each subject or subject combination the simulations are repeated at least 10 times with different random data permutations. The presented values are averages taken from all simulation runs.

Algorithm parameter selection

The parameters τ_d, τ_k, LR and ϵ for the INST RL algorithm have to be chosen according to the application and the expected characteristics of the input data (see Equations 4.17, 4.20 and Algorithm 3).

We choose τ_d and τ_k based on parameter sweeps on the evaluation dataset, so that they fit the data distributions expected in the state space. In Figure 4.10 we show the results for the sweep of τ_d and τ_k in the LC1 setting at different time steps. For this simulation we chose $LR = 0.4$. The plots show optimal results for $\tau_d = 0.8$ and $\tau_k = 1.6$, at all time steps evaluated. For all further simulations we therefore use these values.

For the learning rate LR a trade off between stability and plasticity has to be found. In Figure 4.11 we show the result for the sweep of parameter LR in the LC1 and LC2 setting. A change of the learning rate has only a small effect on the learning behavior. This applies to learning from scratch (LC1) as well as to learning after user change (LC2). We choose $LR = 0.4$ for the following simulations.

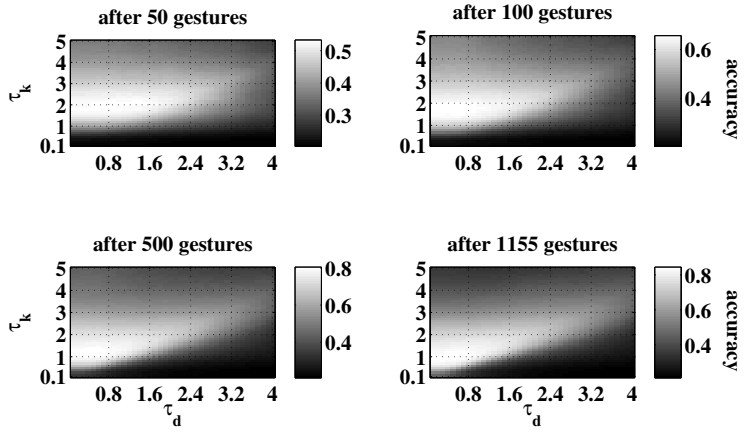


Figure 4.10: Parameter sweep for τ_d and τ_k. The bright area shows the parameter combinations with the best accuracy after learning from the given number of gestures. The parameter combination $\tau_d = 0.8$ and $\tau_k = 1.6$ leads to optimal results after learning from 50, 100, 500 and 1155 instances respectively.

For all simulations the exploration rate is empirically set to $\epsilon = 0.01$ (see Equation 4.15). A summary of the simulation parameters is given in Table 4.3.

Parameter	Value
τ_d	0.8
τ_k	1.6
LR	0.4
ϵ	0.01

Table 4.3: Selected learning parameters for simulating the learning behavior of INST on the gesture recognition dataset (see Equations 4.17, 4.20 and Algorithm 3).

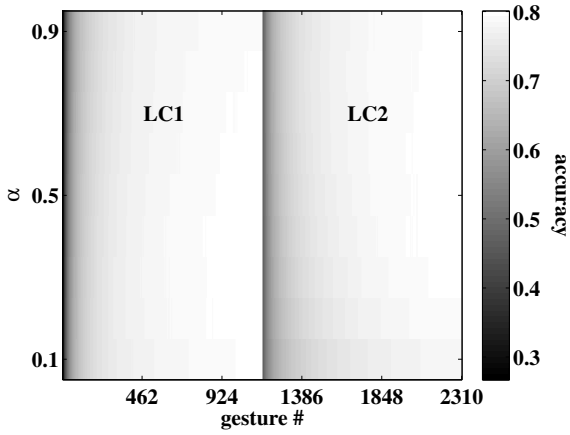


Figure 4.11: Parameter sweep for LR. The learning rate parameter only slightly influences the learning behavior with this dataset. This applies to the LC1 (up to gesture number 1155) as well as to LC2 (from gesture number 1156 onwards). The best result is achieved with $LR = 0.4$.

Simulation results

The simulation result of the learning behavior for LC1 and LC2 is shown in Figure 4.12. The first 1155 input gestures are based on LC1 while the following 1155 input gestures are based on LC2. After 175 gestures the system has learned enough to achieve an accuracy of 72% (90% of the maximum accuracy). The maximum accuracy is the accuracy after learning from 1155 gestures in the LC1 case, which results in 80%. When user B starts using the system (at gesture number 1156) there is a severe drop in accuracy, down to 46%. This indicates that user B performs the gestures differently compared to user A. The learning curve for LC2 has a similar shape compared to LC1 but rises slightly slower. It takes 205 input gestures after the user change to reach 90% of the maximum accuracy.

From this perspective there is no benefit when starting with a pre-trained system compared to starting from scratch. It must be noted however, that the accuracy drops to 46% when user B starts using the system, which is still better than guessing (20% accuracy) The RL improves the recognition accuracy for user B by 36%.

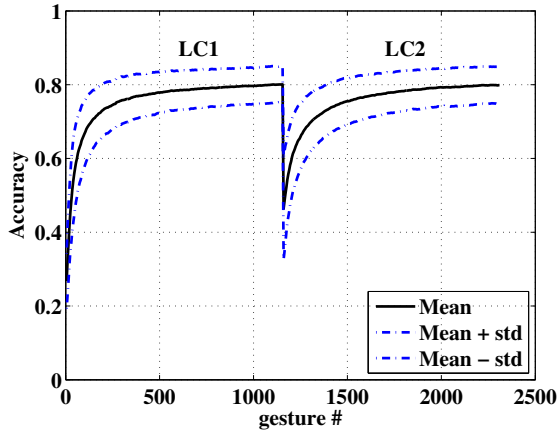


Figure 4.12: Simulation result for the learning cases LC1 and LC2. User A starts using the system which is so far untrained. As the user gestures and indicate errors, the system learns the state-action mapping (LC1). At gesture number 1155 user A passes the system over to user B (LC2). The recognition accuracy drops due to the fact that user B performs the gestures differently compared to user A. While user B continues to input gestures and rewards, the system adapts which leads to an improved recognition performance. The result is given as an average over all users and all data permutations with the according standard deviation.

4.3.6. Discussion

A recognition system based on the INST RL algorithm can be learned based on the input of gestures and the corresponding error feedback. We have demonstrated this in two scenarios; learning from scratch and learning after change of user. In both cases a maximum recognition accuracy of 80% has been reached. A reasonable performance (90% of the maximum accuracy) was achieved after learning from 175 and 205 instances respectively. The relative improvement of the recognition accuracy, achieved through adaptation, is 36% after the new user took over the system (LC2).

The parameter sweeps indicate, that the selection of correct learning parameters is essential for an optimal learning performance. Parameter values depend on the application and may also vary depending on

the user. This is a potential drawback since parameters obtained for a certain group of users may not fit for a new user.

Our simulation results show that learning from scratch is slightly faster compared to learning after user change. In the user change case the system has to unlearn some knowledge about the old user as well as learn the gestures of the new user. This unlearning step delays the system's learning of the new user behavior. Even though the learning after system hand-over is slightly slower compared to learning from scratch, it might still be beneficial for the new user, since the system performs initially better.

4.4. RL gesture recognition case study

We conduct a case study to validate the applicability of learning based on CE feedback in the gesture recognition scenario. A computer game is controlled by hand gestures which are recognized online. Gesture recognition is based on the INST RL algorithm described in Section 4.3.4.

4.4.1. System setup

The user plays a computer game where an object (a colored dot) has to be moved towards a target position (a black dot) within a two dimensional playing grid (see Figure 4.13). Four different input hand gestures are used to move the object in the four directions of the grid, which are the four output actions of the gesture recognition; namely left, right, up and down. Each input gesture moves the object to the next grid position in the according direction until the target position is reached.

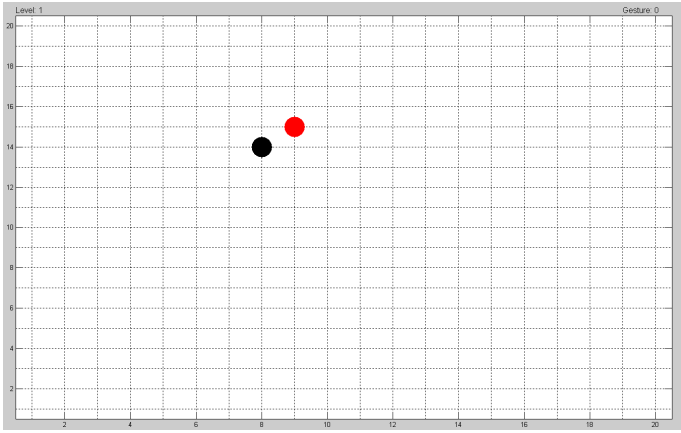


Figure 4.13: Screenshot of the computer game played during the case study. Goal of the game is to move the colored dot to the position of the black dot within the two dimensional grid. The game is controlled by four hand gestures for the four respective directions. Each gesture moves the colored dot in the relating direction to the next grid position.

To capture the hand movement a tri-axial acceleration sensor, integrated in a Texas Instruments ez430 [99] sports watch, is mounted

at the user's wrist. The sensor is sampled with a rate of 23.8 Hz. The user can give error feedback by pushing a button on the sports watch if an input gesture hasn't been recognized correctly. The feedback has to be given prior to performing the next input gesture. The acceleration data as well as the feedback information is transmitted wirelessly to the game computer. In Figure 4.14 we show the full setup comprising of the user, the game computer and the sensor watch.

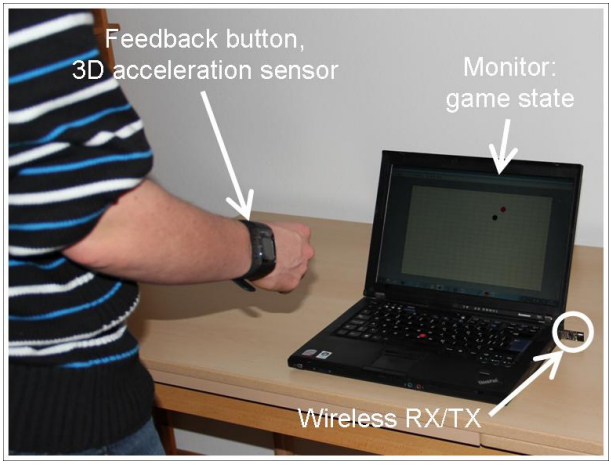


Figure 4.14: The user performs hand gestures to control the computer game. The hand movement is captured by a sports watch integrated acceleration sensor and transmitted wirelessly to the game computer. System recognition errors can be indicated by the user by pushing a button on the sports watch.

The acceleration signal is segmented based on the standard deviation of the low-pass filtered acceleration magnitude. All parts of the signal with a standard deviation below a preset threshold are omitted as depicted in Figure 4.15. Each remaining signal segment contains exactly one gesture.

For each signal segment mean and standard deviation are calculated on the signals of each sensor axis, forming a six dimensional state vector S_t. The action a_t according to the given state S_t is chosen by the gesture recognition system based on the RL agent operating the INST algorithm (see Figure 4.9 and Algorithm 3). The resulting action moves the colored dot on the game grid. The output action is observed by the

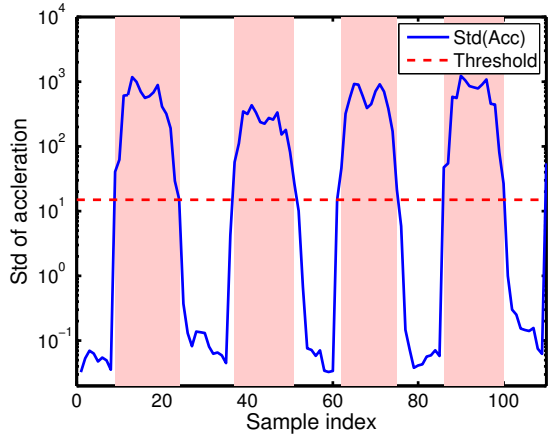


Figure 4.15: Segmentation of the continuous acceleration signal. The marked segments show the parts of the acceleration signal where the standard deviation is greater than a preset threshold. Each of these segments contains a gesture.

user so that system recognition errors can be identified by them and fed back into the system as reward r_{t+1}. Each new input gesture, in combination with the given error feedback, is used by the system to learn state-action mapping.

The learning parameters for the INST algorithm were empirically selected based on data recorded from two test subjects prior to the case study. The chosen parameters are listed in the following table.

Parameter	Value
LR	0.01
τ_k	20
τ_d	3
ε	0.01

4.4.2. Study protocol

Each participant received an instruction sheet prior to the study. The sheet explained in detail how to play the computer game, when to input gestures and when to push the error feedback button. The gestures were described as hand movements “left”, “right”, “up” and “down” without

further explanation. This gives the participants the freedom to perform the actions in the way most comfortable to them. The participants are free to choose if they want to perform the gestures with their left or right hand, and to mount the sensor watch on the according wrist.

Before the participants perform the actual study tasks, they play an introduction game. The system is untrained and learns the state-action mapping from scratch based on the input gestures and the error feedback. This first game allows the participants to get used to the system, the gestures, and the error feedback.

The study consists of the following three tasks. Each task lasts until 100 gestures are captured.

Task 1 **Learning from scratch:** This task is similar to the introduction game. The participant starts using the untrained system which learns the state-action mapping from the gesture input and the error feedback.

Task 2 **Learning after sensor displacement:** The participant removes and reattaches the sensor watch on the wrist. This is likely to induce a sensor displacement since the sensor is likely to be placed at a slightly different location compared to before. In this task the system is not learning from scratch but instead starts from the state-action mapping learned during Task 1. The system has to adapt to the readings of the displaced sensor.

Task 3 **Learning after sensor relocation:** In this setting the sensor is moved to the wrist of the alternate arm and the gestures are performed with the according hand. This induces a stronger change compared to Task 2. The system starts learning from the state-action mapping learned during Task 1. The system has to adapt to the readings of the relocated sensor. This includes the changes induced by the subject performing the gestures with the other hand.

For each subject the duration for all tasks together was between 35 and 50 minutes including an introduction and short breaks between individual tasks. The study was conducted with 18 subjects aged 21 to 57 years.

4.4.3. Evaluation

We evaluate the learning behavior of the RL gesture recognition system based on the error feedback given by the participants. The error feed-

back allows us to estimate the recognition accuracy $\hat{acc} = 1 - \frac{\#errors}{\#gestures}$. In Figure 4.16 we show the accuracy over the number of input gestures, averaged across all participants. The accuracy is estimated on a sliding window of 10 gestures.

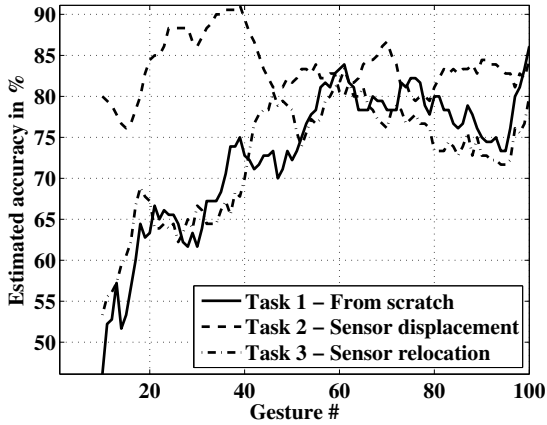


Figure 4.16: Estimated accuracy over the number of input gestures for the three case study tasks. The plot shows averages over all study participants. Task 1: The system is able to learn a state-action mapping from the input gestures and the error feedback. Task 2: Despite the slight sensor displacement the mapping learned in Task 1 still fits. Task 3: The sensor relocation induces a change that requires an adaptation of the state-action mapping.

For Task 1 a steep increase in accuracy can be observed for the first 60 input gestures. The curve flattens out at about 80% accuracy. The system is therefore able to learn state-action mapping from the gesture input and the user feedback. For Task 2 the accuracy is at the 80% level from the beginning. This indicates that the mapping learned in Task 1 also fits quite well in Task 2, despite the slight sensor displacement. For Task 3 the accuracy is below 55% in the beginning and rises to the 80% mark within the first 60 gestures. The change induced by the relocated sensor, in addition to a potentially different gesture execution with the alternate gesture hand, requires an adaptation of the state-action mapping. This adaptation process can be considered successful since the 80% accuracy level is reached again.

There seems to be a slight decrease of performance in the second half of each task. It is not clear if this is caused by the users changing their behavior or failure to provide feedback during the tasks (see Section 4.4.4). Furthermore it may be just an artifact caused by the small window size and the limited number of subjects.

4.4.4. Discussion

The results from our case study confirm our simulation results from Section 4.3. It is possible to learn and adapt a RL based gesture recognition system solely from the input gestures and a user generated reward. In our case study about 60 input gestures were necessary for the system to learn a state-action mapping that resulted in a gesture recognition accuracy of 80%.

One drawback of this approach is that failure to give feedback by the user may result in a less accurate system performance. In our case study we have observed that users forgot to input the feedback occasionally, especially when the system had reached a reasonable performance. This may also be caused by their attention being more drawn to playing the game than to identifying recognition errors.

We did not restrict the participants of the study in the way they performed the gestures. Despite the different gesture executions of various users, the system was still able to learn the gestures. This may increase the comfort of a gesture recognition system since the users are able to perform the gestures in a way comfortable to them and not in the way the system designer intended.

5

Implicit error feedback generation using brain signals

In this chapter we present a novel idea for unobtrusive error feedback generation from brain signals. We evaluate the potential of this approach on a dataset, containing brain signals and motion data, recorded in an extensive gesture recognition experiment. The challenges and adaptation potential, capitalizing on brain based error feedback, is investigated.

5.1. Introduction

The methods described in Chapter 4 require a specific error feedback signal from the user to guide the learning and adaptation. So far this feedback signal was generated explicitly, by the user pushing a button in case of a system recognition error.

In this chapter we capitalize on advances in electroencephalography (EEG) signal processing that allow for error related potential (ErrP) recognition as a source of error feedback. ErrP occurs when someone, i.e. a user, observes unexpected system behavior [100, 101, 102]. We evaluate a hand gesture recognition system that takes advantage of ErrP to adapt.

In Figure 5.1 we sketch the typical ErrP case. A hand gesture $\mathbf{x}_t$ is performed by the user. The output of the application, indicating the recognition result, is provided to the user right after the gesture input. If the recognition result is contrary to the user's expectation an ErrP follows in the user's EEG signal O_t $50 - 250$ ms after the recognition result is provided.

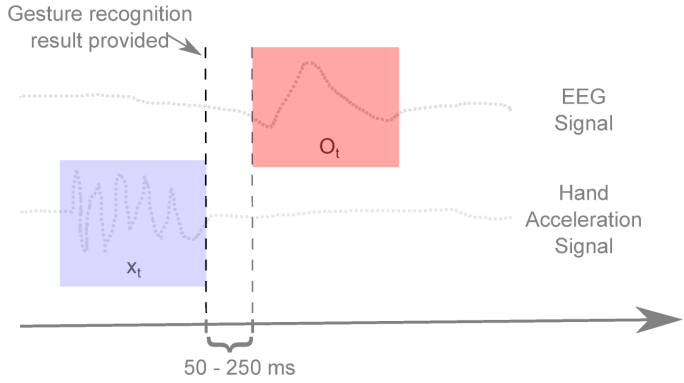


Figure 5.1: The user performs a hand gesture $\mathbf{x}_t$ which is not recognized correctly by the gesture recognition. Feedback about the recognition result is provided to the user. $50 - 250$ ms after the feedback an ErrP follows in the user's EEG signal O_t.

5.2. State of the art in ErrP recognition

Several studies have suggested the existence of a neural system in relation to error processing [103]. Specifically, stereotypical electrophysiological signals have been consistently reported to appear as a response to erroneous actions [104] or unexpected action outcomes [102]. These signals – termed error-related negativity (ERN) and feedback-related negativity (FRN)— are characterized by a negative deflection of the EEG signals in fronto-central areas of the scalp, followed by a centro-parietal positive peak. Typical signal latencies are 50 to 100ms in the case of ERNs and around 250ms for FRNs. Neurophysiological studies have provided evidence of error-based learning. Specifically, it has been suggested that these signals reflect conscious error processing, post-error adjustment of response strategies [104], and reward-based adaptive behavior [102].

Moreover, research on BCI has shown that it is possible to recognize EEG error-related signals (ErrP) on single occurrences better than guessing (two-class problem, $> 50\%$ accuracy) [105, 106, 107]. Based on this fact, these signals have been proposed to be used to correct erroneous motor action in speed-response human-computer interaction [107], as well as to increase the information transfer rate of EEG-based BCI systems [105]. Experimental measures taken over different time periods (up to two years apart) show that these potentials are stable over time, despite the time delay between recordings. Current protocols for EEG signal analysis require motionless subjects to avoid contaminating the subtler EEG signals (10-100μV) with EMG signals (1-30mV) from muscle activity [108]. In order to use EEG systems in naturalistic settings however, researchers have begun investigating limited subject mobility [109].

5.3. ErrP-based adaptive gesture recognition experiment

We conduct an experiment in order to investigate the ErrP based adaptation of a gesture recognition system to a new user. A game is controlled by HCI hand gestures. The gesture recognition system is trained in a user independent manner. The system is then given to a new user. Each gesture instance $\mathbf{x_t}$, performed by this user, is classified by the gesture recognition system to gesture class $\tilde{c}_t$. ErrP analysis indicates whether the action taken by the computer game, and thus the classifi-

cation of the gesture, was correct or erroneous (teacher signal s_t). Based on this information the gesture recognition system adapts to the new user. In Figure 5.2 we illustrate this concept. This scenario is based on

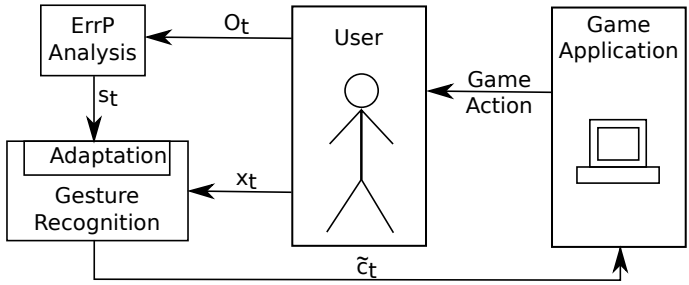


Figure 5.2: The user performs a hand gesture with the intention to control the game application. The hand acceleration is sensed and fed as gesture instance $\mathbf{x}_t$ into the pre-trained gesture recognition system. The recognized gesture class $\tilde{c}_t$ is used to control the game application. The user observes the game and the resulting action. If the action differs from the user's expectation the gesture was not correctly recognized from the hand acceleration. The user will perceive this deviation from the expected behavior resulting in an ErrP in the EEG signal instance O_t. The result s_t of the ErrP analysis, together with gesture instance $\mathbf{x}_t$, are used to adapt the gesture recognition.

a game to maintain the user's involvement during experimental sessions [82, 105]. The game is designed so that a typical gesture input speed of $30\frac{\text{gestures}}{min}$ is achieved. This allows for acquisition of a large number of gesture instances (> 2000) in a short amount of time (< 2 hours).

The goal of this experiment is to collect a dataset, containing hand gesture acceleration instances $\mathbf{x}_t$ and the according EEG signal instances O_t, within our scenario (Figure 5.2) to assess adaptation strategies in offline simulations. The online gesture recognition during the experiment is therefore not based on the acceleration data but on a reliable light barrier gesture recognition system (described in Section 5.3.2), that provides the gesture ground truth c_t. Since the light barrier based gesture recognition is deemed error free, recognition errors are artificially generated (resulting in $\tilde{c}_t$), to trigger ErrP events in the EEG signal O_t for later offline evaluation. The actual experimental setup is depicted in Figure 5.3.

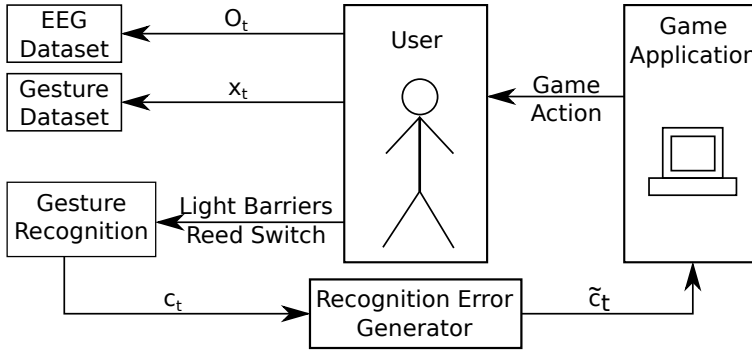


Figure 5.3: The user performs a hand gesture with the intention to control the game application. The gesture is reliably recognized based on a light barrier frame and a reed switch. Recognition errors are artificially induced in the gesture ground truth c_t. The resulting output $\tilde{c}_t$ is used to control the game application. For every induced recognition error it is assumed that an ErrP is triggered in O_t. During the experiment hand gesture acceleration instances $\mathbf{x}_t$ and EEG signal instances O_t are recorded for later offline analysis.

5.3.1. Gesture-controlled computer game

The subjects play a computerized version of a "memory game" consisting of 8 image pairs (Figure 5.4). The 16 images are randomly distributed in a four by four matrix and hidden behind question marks. The subjects have to find identical pairs of images, which are then removed from the screen. If two images are flipped and don't match, they are hidden again before new images can be selected. The game is finished when all image pairs were correctly found.

The game input interface is based on five hand gesture classes. Left, right, up and down hand movements shift the image selection cursor in the corresponding direction. Flipping an image is controlled by closing and opening the hand.

5.3.2. Measurement setup

The online recognition of the gestures during the data collection is based on light barriers and a reed switch. This ensures accurate gesture ground truth for the collection of a reference dataset. Three horizontal

and three vertical infrared light barriers return the sequence of hand positions (Figure 5.4) from which the gesture is inferred. The closing gesture is detected from a reed switch on the subjects hand activated by a magnet on the subjects fingers.

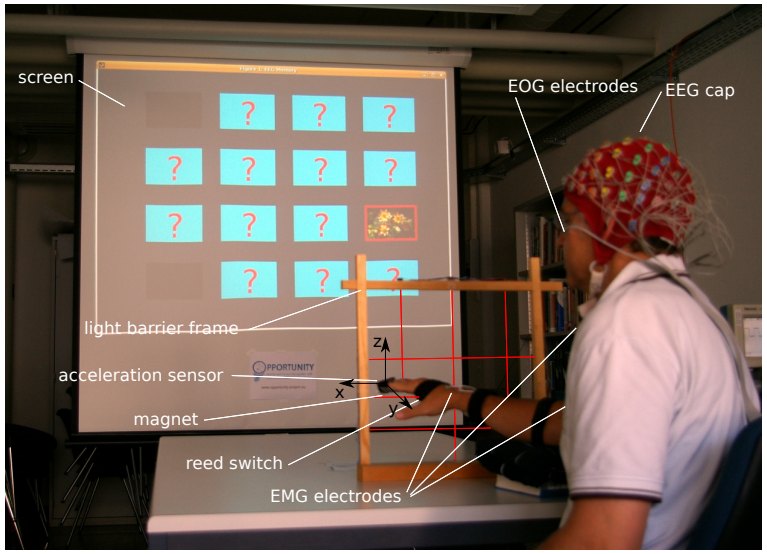


Figure 5.4: The computer game is presented on the screen; the light-barrier frame, magnet and reed switches capture game control gestures; the acceleration sensor (orientation relative to hand) and the EEG electrode cap stream data to a PC for recording and offline analysis.

A tri-axial acceleration sensor at the subjects fingertips records the motion of the hand for offline acceleration-based gesture recognition simulations. The acceleration sensor is sampled at 64 Hz and connected via USB to the experiment computer. This computer also runs the memory game. Another computer records EEG with the Biosemi ActiveTwo system and active electrodes. Both computers are interconnected using a shared data line to ensure a synchronized data recording.

5.3.3. Experimental protocol

Seven healthy male subjects aged 25 to 47 participated. For each subject we recorded 14 sessions with a duration of three to five minutes. One session corresponds to one “memory game”. Between recording

sessions the subjects could rest for one to two minutes. We recorded more than 2700 hand gestures per subject. The experiment lasted about two hours per subject including setup and introduction.

In each session we randomly induced between 5% and 33% of gesture recognition errors in c_t to provoke ErrP events. To induce an error the error generater (Figure 5.3) selects a random gesture $\tilde{c}_t$ instead of the actual recognized gesture c_t. For example if the subject closes their hand to turn a card, the card would not be turned but instead the cursor would be moved in a random direction. If no error is induced $\tilde{c}_t = c_t$.

5.4. EEG ErrP Recognition

In the EEG signal analysis (see Figure 5.2) we classify the EEG signal O_t into *error* ($s_t = 1$) and *correct* ($s_t = 1$) based on the presence of ErrP. An *error* detected from the EEG signal indicates, that the gesture performed by the user was not recognized correctly by the gesture recognition, and therefore the game output action did not meet the expectation of the user. We exploit the fronto-central distribution of ErrP [82, 105], using the time signals of EEG electrodes FCz and Cz (see Figure 5.5b for scalp positions) as input features for a Bayesian filter [110].

In order to remove the background brain activity and to enhance localized activity, raw EEG potentials are spatially filtered by computing the Common Average Reference, i.e. by subtracting, at each time step, the average potential of all electrodes from each electrode. We exclude from the analysis the most external EEG channels, since those are more likely affected by muscular artifacts related to facial or head movements. In addition, signals of selected electrodes exceeding $80\mu V$ amplitude are discarded. Following previous studies in this type of potentials, signals are then filtered (1–10 Hz bandpass) and subsampled with a sampling rate of 64 Hz before classification. The input vector O_t for the classifier (see Section 5.4.1) is composed by the time samples on electrodes FCz and Cz within the [200 400] ms time window after the end of the hand gesture.

5.4.1. Classification based on Bayesian filtering

The Bayesian filter estimates the state probabilities at each sampling time step according to the observations and the previous state estimations [111]. Through discrete observations of a continuous EEG signal we want to find out, if the output action on the screen matches the input gesture intended by the user.

To build the Bayesian filter, two possible states are defined at each time t: $s_t \in \{1, 0\}$ for erroneous and correct gesture recognition, respectively. At each sampling time step t observations O_t are given by a vector with components FCz and Cz corresponding to the electrodes of the same name: $O_t = [FCz_t, Cz_t]$. Observations and states from time zero to T are respectively noted $O_{0:T}$ and $s_{0:T}$.

A transition model is defined by a first order Markov hypothesis for states over time: $P(s_t|s_{0:t-1}) = P(s_t|s_{t-1})$ for $t = 0 \dots T$. Since

the state during a potential ErrP instance doesn't change, the transition model corresponds to the identity matrix: $P(s_t|s_{t-1}) = 1$ if $s_t = s_{t-1}$ and zero otherwise.

The sensor model is given by the probability distribution $P(O_t|s_t)$ which predicts observations given the state. Then the decomposition of the joint probability is given by:

$$P(s_{0:T}O_{0:T}) = P(s_0)P(O_0|s_0)\prod_{t=1}^{T}(P(s_t|s_{t-1})P(O_t|s_t)) \tag{5.1}$$

The classification consists in estimating $P(s_t|O_{0:t})$, i.e. the probability of the state (*error* or *correct*) knowing the observations (EEG activity). It can be obtained in a recurrent manner. First the state is predicted (Equation 5.2) based on the transition model. Secondly the state estimation (Equation 5.3) is computed based on the sensor model.

$$P(s_t|O_{0:t-1}) = \sum_{s_{t-1}}(P(s_t|s_{t-1})P(s_{t-1}|O_{0:t-1})) \tag{5.2}$$

$$P(s_t|O_{0:t}) \propto P(O_t|s_t)P(s_t|O_{0:t-1}) \tag{5.3}$$

Since the state doesn't change within one signal occurrence the transition model corresponds to the identity matrix. the *prediction–estimation* recurrent calculus is simplified:

$$P(s_t|s_{t-1}) = 1 \text{ if } s_t = s_{t-1} \text{ and zero otherwise} \tag{5.4}$$

$$P(s_t = 1\,|O_{1:t}) \propto \frac{1}{C}P(O_t|s_t)P(s_{t-1} = 1\,|O_{1:t-1}) \tag{5.5}$$

$$P(s_t = 0\,|O_{1:t}) \propto \frac{1}{C}P(O_t|s_t)P(s_{t-1} = 0\,|O_{1:t-1}) \tag{5.6}$$

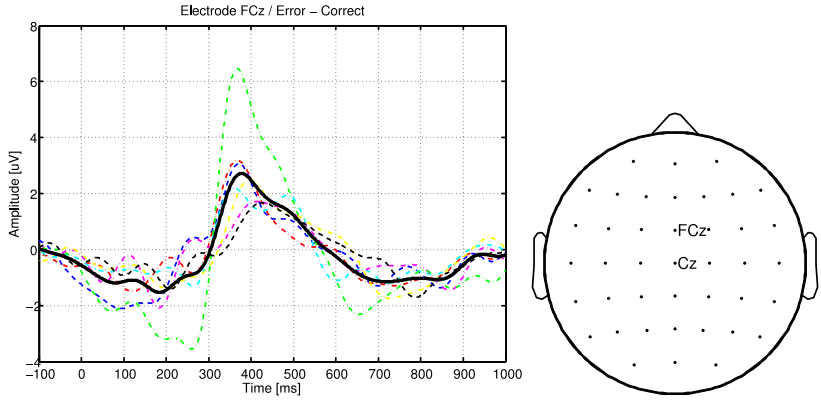
with C being a normalization factor.
At the end of the EEG occurrence at time $t = T$, an erroneous trial is detected if

$$P(s_T = 1|O_{1:T}) > \psi \tag{5.7}$$

where ψ is our current decision threshold and $T = 400ms$. Estimations from both channels are combined using a naive fusion.

$$P(O_t|s_t) = P(FCz_t|s_t)P(Cz_t|s_t) \tag{5.8}$$

The sensor model $P(O_t|s_t)$ is defined by a mono-dimensional Gaussian distribution with a mean μ_t and a variance σ_t^2. Having two input channels and two possible states, there are four Gaussian distributions at each time t, and eight parameters to identify. This approach updates the estimated state probability as new samples are available. Since the prediction-estimation update multiplies two probabilities, the next state estimate would be close to zero if one of the multiplied values would be close to zero. This would effectively stop the recursive Bayesian estimation. To avoid this effect, a lower limit for probabilities was introduced, so that any $P(s_t\ |O_t) < 0.01$ is forced to be equal to 0.01. Figure 5.5a shows the average EEG activity (*error* minus *correct* condition) for all subjects.



(a) (b)

Figure 5.5: (a) Average ErrP on the FCz electrode for the different subjects. The difference signal between the signal with *error* condition and the signal with *correct* condition is shown. *thick line*: average over all subjects; *thin dashed lines* individual averages for each of the seven subjects. The time (t=0) refers to the end of the input gesture after which an ErrP might occur.
(b) Electrode positions shown over the scalp.

5.4.2. ErrP Classification

As shown in Section 5.4.1, the Bayesian Filtering based classifier allows for classification of individual ErrP occurrences. We trained our

classifier subject dependent for each subject on eight of the fourteen recorded memory game sessions, estimating μ_t and σ_t^2 for the two states s_t. We tested on the remaining six recordings. We consider the activity of electrodes in the [200,400] ms time windows after the feedback presentation, estimating the state probabilities according to these observations. Figure 5.6 shows receiver operating characteristics (ROC) for all subjects, where *sensitivity* represents the amount of true positives (*error* class) and *specificity* represents the amount of true negatives (*correct* class). Different *sensitivity-specificity* combinations were computed varying the decision threshold ψ (as defined in eq. 5.7).
It is important to notice that the particular task performed by subjects may induce EMG artifacts due to arm and facial movements [112]. Nevertheless, in the current experimental protocol we reduce this aspect as gesture recognition feedback is only provided once the gesture has been completed. The classification of ErrP occurrences is better than guessing. In the following section we investigate the benefit of using this ErrP classification result for adapting the acceleration based gesture recognition system.

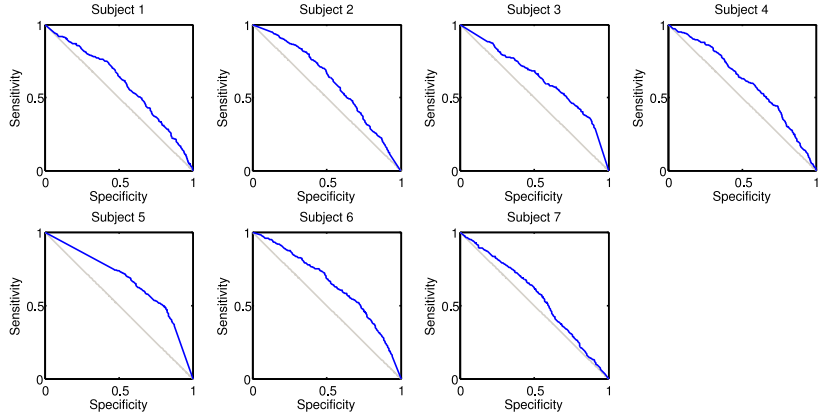


Figure 5.6: Receiver operating characteristics (ROC) curves of ErrP classification for all subjects. For each subject the classification of ErrP occurrences performs better than guessing.

5.5. Adaptive gesture recognition guided by ErrP

We investigate how the error feedback signal s_t, as provided by the ErrP classification, can be used to adapt the gesture recognition system. This analysis is based on the gesture dataset recorded during the experiment (Section 5.3), which contains the subject's hand acceleration.

5.5.1. Gesture classification

We distinguish the five game control gestures based on the hand acceleration. We segment the signal using the gesture-start and gesture-end signal provided by the light-barrier frame. During initial training of classifiers prior to adaptation, the ground truth label c_t of gesture instances $\mathbf{x}_t$ is provided by the light-barrier frame. We did no dataset cleaning or outlier removal as this would not be possible in the real application of such a system. We simulate the adaptation process based on the pre-recorded dataset.

For the training and during the operation, we calculate the following acceleration features on three windows (full gesture, first and second half of the gesture): mean, standard deviation, minimum, maximum and energy. We do this on the three axes of the acceleration signal as well as on its magnitude. In addition the correlation for each axes pair xy, xz and yz is calculated. This yields 63 features. We perform a probabilistic feature selection [113] combined with a scatter search [114] to select a feature subset [115]. This yields a six-dimensional feature vector $\mathbf{x_t}$ containing: the mean on the y-axis, the first half on the y-axis and on the magnitude, the minimum on the magnitude, the mean of the first half of the z-axis and the standard deviation on the first half of the x-axis.

We classify the gestures $\mathbf{x}_t$ with the incremental kNN classifier described in Section 4.2 since it supports learning from a CE teacher signal s_t. The learning of the initial classifier is based on standard kNN learning by adding the training instances with a weight of 1 to the classifier model. The incremental online adaptation is based on the correct and error learning strategies. The learning parameters were empirically set to the values listed in Table 5.1.

To train the user independent classifier $C_{t=0}$ on the recorded dataset we combine the data of all subjects, leaving out the subject we want to adapt to. From this combined dataset we randomly select 480 training instances, evenly distributed over all classes. An initial training set of

parameter	value
LR	0.002
R	0.02
κ	0.1
α	0.4
δ	48

Table 5.1: Parameters for the incremental kNN classier as used in the simulations of the ErrP based adaptation (see Algorithms 1, 2 and Equations 4.6, 4.7, 4.8, 4.9, 4.10).

this size results in an initial classifier while limiting the computational complexity for classification.

The data of the left out subject is split into an *adaptation set* and a *test set*. The *adaptation set* contains 2248 instances while the *test set* contains 500 instances. During operation, the instances in the *adaptation set* are iteratively (one at a time) presented to the system for classification and adaptation. The initial user independent classifier C_0 and the resulting user adapted classifier C_T (after presenting the *adaptation set*) are tested on the *test set*.

For the following evaluations we simulate the ErrP based teacher signal s_t in the following way. A gesture instance $\mathbf{x}_t$ is classified by C_t. If the predicted gesture class $\tilde{c}_t$ matches the ground truth ($\tilde{c}_t = c_t$), the recognition was correct and the teacher signal is set to *correct*($s_t = 0$). If $\tilde{c}_t$ does not match the ground truth ($(\tilde{c}_t \neq c_t)$ the recognition was wrong and the teacher signal is set to *error* ($s_t = 1$). Furthermore we simulate different ErrP recognition performances by randomly inverting the teacher signal s_t according to the probabilities given by the targeted ErrP recognition sensitivity and specificity.

5.5.2. Influence of the ErrP detection accuracy on the adaptation

The ErrP recognition performance (see Section 5.4) is a key parameter for successful user adaptation of the gesture recognition system. We investigate how the sensitivity and the specificity of ErrP recognition can influence the adaptation of the gesture recognition system. In Figure 5.7 we show the adaptation result for different simulated ErrP

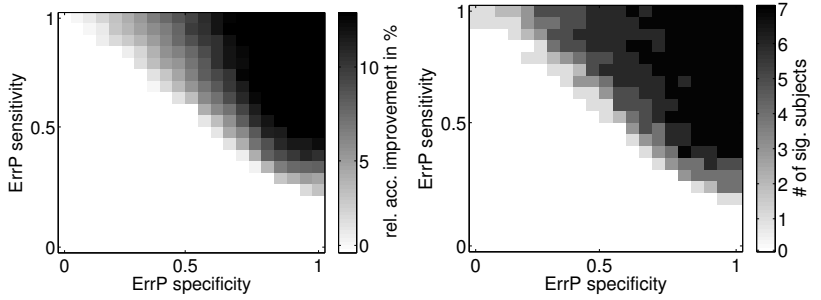


Figure 5.7: ROC analysis of the gesture recognition system adaptation behavior for all possible combinations of ErrP sensitivity and specificity. Average over 20 simulations and 7 subjects from different random data permutations. Left plot: Number of subjects for which the adaptation leads to a significant (one sided t-test with $\alpha = 0.05$) improvement of the gesture recognition. Right plot: Average relative improvement of the gesture recognition accuracy.

sensitivity-specificity combinations. The average subject independent baseline accuracy of the gesture recognition is 74.7%.

In case of a perfect ErrP signal ($sensitivity = specificity = 1$) we achieve an increase of 17.1% in accuracy compared to our baseline. A low sensitivity affects the adaptation slightly more than a low specificity. This is also reflected by the average relative improvement of the adapted classifier. The increase is less with a lower specificity than with a lower sensitivity. In general with an ErrP detection performing better than chance ($sensitivity + specificity \geq 1$) an improvement of the gesture recognition system can be expected.

5.5.3. Adaptation assuming the ErrP recognition performance resulting from the HCI gesture experiment

The ErrP recognition can be tuned to any working point on its ROC curve (see Section 5.4). In the following analysis we choose the optimal working point based on the ErrP analysis of the EEG data recorded from the HCI gesture experiment. The ErrP recognition performance varies between subjects. Therefore we choose the working points for each subject individually.

For each subject we calculate the accuracy gains, when adapting

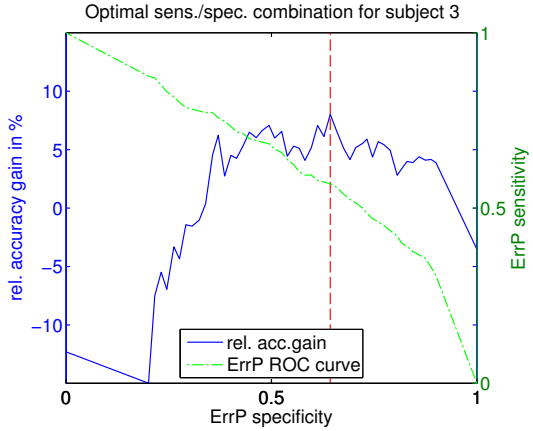


Figure 5.8: The curve shows the expected accuracy gain for the adaptation of the gesture recognition system, relative to the subject independent baseline, for all points on the ErrP ROC curve of this subject. The vertical line marks the maximum of the accuracy gain. Its intersection with the ErrP ROC curve marks the optimal ErrP sensitivity/specificity combination.

the gesture recognition with the possible sensitivity/specificity combinations for this subject, according to the ROC curves in Figure 5.6. The optimal sensitivity/specificity combination for each subject is that, where the accuracy gain by adaptation, compared to the subject independent baseline, is maximal. An example of such an optimization is shown in Figure 5.8.

In Figure 5.9 we show the selected optimal working points for all subjects. With the exception of one subject, they all fall in the region where we can expect an improved adapted gesture recognition system. The exact ErrP sensitivity and specificity values, together with the expected improvement and the resulting accuracy of the gesture recognition, are listed in Table 5.2 for each subject individually. For six of the seven subjects a significant (one sided t-test with $\alpha = 0.05$) gain in accuracy can be expected through adaptation.

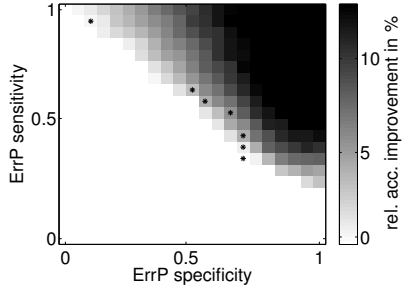


Figure 5.9: The * mark the ErrP sensitivity/specificity combinations for each subject which lead to the, in this case, optimal user adapted gesture recognition system.

Subject	1	2	3	4	5	6	7	AVG
SI accuracy (C_0)	74.8	79.4	77.2	89.7	77.0	64.6	59.8	74.7
ErrP threshold	0.87	0.16	0.09	0.14	0.80	0.35	0.71	0.45
ErrP sensitivity	0.76	0.58	0.57	0.13	0.70	0.73	0.76	0.60
ErrP specificity	0.38	0.58	0.64	0.93	0.56	0.47	0.34	0.56
Adapt. acc. (C_T)	81.3	81.5	83.4	89.9	80.0	72.0	67.2	79.3
Acc. gain in %	8.8*	2.7*	8.0*	0.4	4.0*	11.4*	12.2*	6.8

Table 5.2: Detailed simulation results for the gesture recognition adaptation based on ErrP. The subject independent (SI) recognition baseline for classifier C_0 and the accuracy after adaptation for C_T are given together with the according ErrP recognition parameters. The * indicates a significant improvement (one sided t-test with $\alpha = 0.05$). All results are averages of 20 simulations with different random data permutations.

5.6. Discussion

In this work we assess, for the first time, the recognition of EEG error-related potential occurrences in a complex and realistic task. Compared to previously reported experiments [82, 105, 110], the visual stimuli, by the "memory game" application, were more complex. Furthermore the subjects were cognitively involved in game playing and explicitly allowed movements, unlike in the other experiments. The differences in the experimental protocol (i.e. subject moving during the recording, complex visual feedback, different cognitive demand of the experimental task) together with the intrinsic variability, noise, and non-stationarities of brain signals, explain the low ErrP recognition accura-

cies obtained in the current study (see Figure 5.6) compared to previous studies. Nevertheless, it should be recognized that it is not possible to achieve 100% ErrP recognition accuracy from EEG signals due to the low signal-to-noise ratio, EEG non-stationarity and EMG contamination. Indeed, best classification performances for ErrP recognition in previous, controlled experiments lies generally around 80% [82, 105]. Therefore our ErrP recognition results are encouraging.

Despite the low ErrP recognition performance, it was still possible to use the ErrP information to successfully adapt the gesture recognition system towards a specific user. The gain in performance of the adapted gesture recognition system is dependent on the ErrP recognition. The better the ErrP recognition performs the larger the improvement in gesture recognition that can be expected.

As we rely on a subject independent gesture recognition system as a basis for our adaptation it is important that the initial gesture recognition reaches a sufficient recognition performance.

In our experimental setup for data collection we assume that the subjects intention is correctly captured by the gesture recognition. There might still be cases where the subject performs a gesture contrary to their intention by mistake. This mistake might also be reflected in the brain signal as an error. As we do not capture the users intention directly we can not assess the influence of user mistakes.

During the data recording experiment the gesture recognition errors are added artificially and randomly, so that the subjects can not adapt their movement strategies to improve the gesture recognition. Therefore the simulated improvements of the gesture recognition are independent of potential changes in the user behavior.

In the proposed adaptation schemes the ErrP recognition is interpreted as a binary value, and all occurrences are used for adaptation. However, the output of the Bayesian filter based classifier provides a posterior probability of the state class and can therefore be also used as a reliability measure of the classification. Previous studies in BCI applications have shown that rejection mechanisms based on the probability values may lead to improvements in the overall information transfer rate provided by the EEG decoding systems [116]. Similarly the gesture recognition adaptation could take only those gesture instances into account, which are clearly identified as *error* or *correct*. This might improve the gesture recognition adaptation.

5.7. Conclusion

We have investigated online user adaptation guided by a ErrP based error feedback signal. We chose a gesture based HCI scenario to evaluate our approach. We tested the system using perfect error feedback signals and also evaluated the performance when this feedback is implicitly provided by the user through decoding of the user's brain activity. To our knowledge, this is the first attempt to use brain signals related to the perception of errors for the improvement of activity recognition systems. Simulations of perfect decoding of such signals show that theoretically the recognition accuracy can be increased by up to 17.1% over the user independent classifier. Using single-trial recognition of actual EEG data recorded during the gesture based HCI experiment, the accuracy increase for the adapted gesture recognition reaches 6.8% in average. This shows that brain signals (i.e. EEG) generated during real human-computer interaction provide information that can be integrated into the activity recognition chain so as to improve its performance.

In the short term EEG-based user adaptation remains unlikely in real-world scenarios given the current state of the sensing technology, its sensitivity to motion artifacts, and the desire for invisible wearables. However portable sensing platforms are now becoming available, and there is an active research effort along these lines [117].

6

Adaptive recognition and user behavior

In this chapter we focus on the influence of system adaptation on user behavior. We investigate how the user changes their behavior when interacting with an adaptive gesture recognition system compared to a static system. Furthermore the effect of a change in user behavior on the adaptation is studied.

6.1. Introduction

In the previous chapters we have presented different methods for adapting a recognition system to the user (system→user adaptation). Whenever a user interacts with an activity aware system that provides an immediate feedback to the user (user in the loop), there is also the potential for the user to adapt their behavior according to the feedback they receive. It is known, that humans adapt their movement strategies to achieve certain goals in varying environmental conditions, for example during grasping tasks [118, 119]. Also in handwriting recognition an adaptation of the user to the system, in order to improve the recognition, has been observed [120]. A co-adaptation between user and system has also been investigated for BCIs [74] and a robotic prosthetic hand controlled via EMG signals [75]. Such a co-adaptation may also be present when interacting with an activity recognition system, with the user adapting the way gestures or activities are performed. The goals pursued by the adapting user could be a better recognition, more convenient or less tiring movements, or a faster input of commands (e.g. in HCI applications). Thus, the user also adapts to the system (user→system adaptation).

6.1.1. Issues of co-adaptation

To design better activity or gesture recognition systems, with the user in the loop, we have to take into account the user's ability to adapt. The user↔system co-adaptation has to be understood, especially with systems that provide adaptive mechanisms.

The two adaptive processes, the user and the system, may influence each other, which may result in the adaptation not being possible or beneficial. For example when one process adapts faster than the other, the faster process might continuously change conditions that the slower process is not capable of following. This is linked to stability issues [121], when two systems (or the system and the user) adapt to each other concurrently. It is also not clear how useful system adaptation is, when user adaptation is already present. The user might have more freedom in adapting their behavior than the system does. On the other hand, the adaptive recognition system may give the user more comfort, e.g. by providing a higher recognition rate, or enabling a more efficient behavior by giving more freedom in the movement execution.

We investigate these aspects on a gesture recognition experiment. The experiment is designed so that it allows for evaluation of the sys-

tem's adaptation behavior according to the gesture recognition accuracy. Furthermore the user behavior can be investigated by analyzing the users' hand gesture trajectories.

6.1.2. Gesture recognition scenario

We consider an experimental setup, comprising of the key elements of an activity aware assistant (activity sensing, provision of a user feedback based on the sensed activity, user can communicate with the assistant). Such a setup allows for the study of co-adaptation dynamics.

Our scenario combines the characteristics outlined above. Hand gestures are recognized from the user's hand movements, captured by an optical motion tracker. The actions according to the input gestures are observed by the user via a computer screen. Gesture ground truth can be indicated by the user to the system.

The gesture recognition system is pre-trained in a user-independent fashion. Initially it is not optimized for a specific new user. Nevertheless it can recognize gestures with reasonably high accuracy if they are performed according to the system's model.

We investigate two different adaptation processes; the adaptation of the user to the system (user→system) and the adaptation of the system to the user (system→user). In the first case the user uses the feedback given by the system to adapt their movement strategy in a way that is most compliant with the system's model, effectively improving the recognition. In the second case the system uses the feedback, given by the user, to adapt it's model to best reflect this user's properties.

The key point is to understand the user↔system co-adaptation dynamics. Users adapt to the characteristics of the system they interact with [122]. Here, at the same time, the recognition system attempts to adapt to the user. We investigate whether a better performance in the system, observed after a long period of interaction, is the result of the system's adaptive mechanism or due to the natural adaptation of the user as they become more used to the system.

We also make the hypothesis, that the system's adaptive behavior affects the user's behavior and comfort, by allowing for more freedom (e.g. in gesture execution) together with a higher recognition performance.

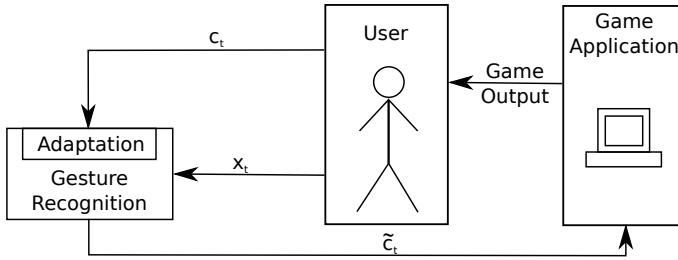


Figure 6.1: The user performs a hand gesture to control the application. The sensed hand trajectory $\mathbf{x}_t$ is fed into the pretrained gesture recognition system. The recognized gesture class $\tilde{c}_t$ controls the game application. The user observes the game and the action resulting. They input the ground truth c_t of the intended gesture. The ground truth c_t, together with the gesture input trajectory $\mathbf{x}_t$, are used to adapt the recognition system.

6.2. Online adaptation experiment setup

We investigate user adaptation within a scenario where a game is controlled by HCI hand gestures. The gesture recognition system is initially trained in a user-independent manner. The system is then given to a new, so far unseen, user. Each gesture $\mathbf{x_t}$ performed by this user is classified by the gesture recognition system to $\tilde{c}_t$, with the resulting action outputted on the computer screen. The user indicates the intended ground truth c_t of their last gesture to the system. Based on this information the gesture recognition system is adapted to the new user. In Figure 6.1 we illustrate this concept.

We decouple the problem of adaptation from the problem of user feedback, and focus on the former aspect, to study the user↔system co-adaptation dynamics. Thus, ground truth feedback is inputted by the user via a computer keyboard directly after every gesture they perform.

6.2.1. Gesture-controlled computer game

The subjects play a computerized version of a “memory game similar” to the one described in Section 5.3. This time 18 image pairs have to be found (Figure 6.2).

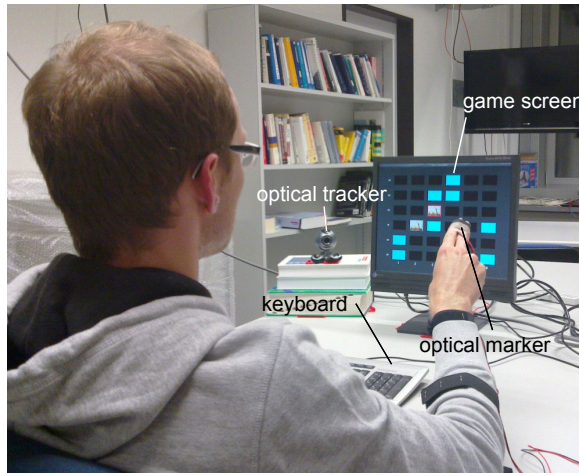


Figure 6.2: Experimental setup. The computer game is presented on the screen. The optical motion tracker, based on a camera and an optical marker, captures the input gestures. Ground truth is inputted via the keyboard.

The game input interface differs from the one described in Section 5.3. Here it is based on six right hand gestures, namely the numbers one to six written with the right hand in the air. The gestures were inspired by [123]. The gestures are defined as single stroke gestures as depicted in Figure 6.3. An image at position (x, y) in the game matrix is selected by entering the according coordinates with two adjacent hand gestures. The first gesture selects the column, the second gesture the row. A selection of column or row can not be reverted or changed. The selected column is highlighted on the screen by drawing red frames around every image placeholder within this column. After the user enters the row number, only the selected image is highlighted with a red frame and revealed. By highlighting the chosen column and row the user gets feedback on the recognition result of the inputted gestures.

The movement of the user's hand is captured by an optical tracking system. An optical marker is placed on the subject's right hand and tracked with a sampling rate of 25 Hz and a resolution of 640x480 pixels. The ground truth of a gesture is inputted by the subject via the numbers 1 to 6 on a computer keyboard, operated with the left hand. A new gesture can only be inputted, if the ground truth has

Figure 6.3: Single stroke gestures $c = 1, \ldots, 6$. The dot marks the start and finish-point of the trajectory described by the hand movement when drawing the number in the air. The dashed lines mark the section of the trajectory necessary to achieve an identical start and finish-point. The arrow indicates the direction of the movement.

been indicated prior to the start of the movement. This ensures that ground truth input is given for each gesture performed.

6.2.2. Online adaptive gesture recognition

During system use, the user's gestures are recognized online. The motion trajectory of the hand is captured by an optical motion tracking system. The continuous trajectory is segmented automatically based on the movement speed. This is possible because the hand remains mostly static between gestures, while it moves faster while performing a gesture. The speed is calculated based on the distance traveled between two location samples. The resulting speed signal is low pass filtered to eliminate noise. A movement speed of more than 10 pixels per second indicates a gesture. A plausibility analysis eliminates all segments shorter than 14 and longer than 100 samples. All resulting segments $\mathbf{x}_t$ are assumed to contain a valid gesture, therefore this segmentation also acts as a NULL class rejection.

Gesture classification based on Active Shape Models

The classification of the gestures is based on Active Shape Models (ASM) [124]. ASMs allow for an investigation and comparison of gestures according to motion trajectories. Motion trajectories can be better interpreted compared to abstract features typically used with other classification methods.

For each class c a ASM is built from training examples. Prior to model generation the training trajectory instances $\mathbf{x}_{i,c}$ are aligned and resampled to contain $n = 30$ equidistant points each. Since each point

is represented by a x- and a y-coordinate, each trajectory instance $\mathbf{x}_i, c$ is $2n$ dimensional. The ASM consists of the mean trajectory $\bar{\mathbf{x}}_c$ of the training instances and eigenvectors $\mathbf{p}_{k,c}$ that capture the variation within the training set. For N training trajectory instances of class c $\bar{\mathbf{x}}_c$ is calculated using

$$\bar{\mathbf{x}}_c = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_{i,c} \tag{6.1}$$

To calculate the eigenvectors $\mathbf{p}_{k,c}$ we first calculate for each trajectory instance $\mathbf{x}_{i,c}$ the deviation $d\mathbf{x}_{i,x}$ from the mean trajectory $\bar{\mathbf{x}}_c$,

$$d\mathbf{x}_{i,c} = \mathbf{x}_{i,c} - \bar{\mathbf{x}}_c \tag{6.2}$$

and then the $2n \times 2n$ covariance matrix $\mathbf{\Sigma}_c$.

$$\mathbf{\Sigma}_c = \frac{1}{N} \sum_{i=1}^{N} d\mathbf{x}_{i,c} d\mathbf{x}_{i,c}^T \tag{6.3}$$

The variation in the training set is described by the eigenvectors $\mathbf{p}_{k,c}$ of $\mathbf{\Sigma}_c$, such that

$$\mathbf{\Sigma}_c \mathbf{p}_{k,c} = \lambda_{k,c} \mathbf{p}_{k,c} \tag{6.4}$$

with $\lambda_{k,c}$ being th kth eigenvalue ($k = 1, \ldots, 2n$) of $\mathbf{\Sigma}_c$. To cope with noise within the training set we keep all l eigenvectors $\mathbf{p}_{l,c}$ in the ASM of each class c, which are in the direction of the strongest variations and capture 95% of it.

$$\sum_{k=1}^{l} \lambda_{k,c} = 0.95 \tag{6.5}$$

The training examples were recorded from five different users prior to the experiment. The setup of the training data recording was identical to the experiment setup. The instruction given to the training subjects, on how to perform the gestures, was done in the same way as later with the experiment subjects.

20 training examples per class were used to build the user independent classifier model. The ASMs for each class are shown in Figure 6.4. The eigenvectors cover different trajectory sizes or start/end-point variations, to support robust classification.

A new input gesture is classified to one of the gesture classes (Figure 6.3) in the following way. The segmented gesture trajectory is aligned and resampled in the same way as the training trajectory resulting in

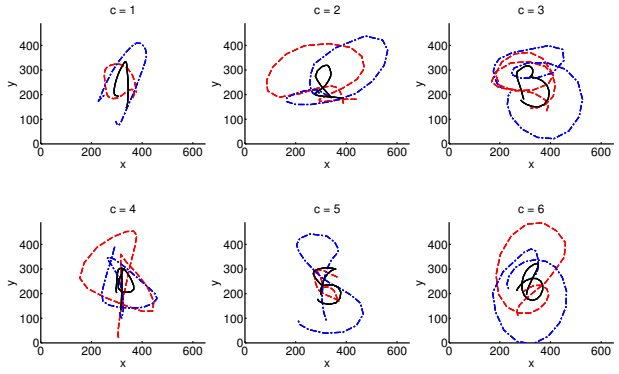


Figure 6.4: Active Shape Models for the six classes. The black line is the mean trajectory $\bar{\mathbf{x}}_\mathbf{c}$. Both the dashed, and the dash-dotted line illustrate the variations captured by the first eigenvector $\mathbf{p}_{1,c}$ for $\pm 3\sqrt{\lambda_{1,c}}$.

a gesture trajectory instance $\mathbf{x}_j$ to classify. $\mathbf{x}_j$ is fitted to the mean trajectories $\bar{\mathbf{x}}_c$ of all classes c along the directions of variation given by $\mathbf{p}_{k,c}$.

At first the difference between the trajectory $\mathbf{x}_j$ to classify and the mean trajectory $\bar{\mathbf{x}}_c$ is calculated.

$$d\mathbf{x}_{j,c} = \mathbf{x}_j - \bar{\mathbf{x}}_c \tag{6.6}$$

Then the amount of variation in direction of the eigenvectors $\mathbf{p}_{k,c}$ is determined.

$$\lambda_{j,k,c} = d\mathbf{x}_{j,c}\mathbf{p}_{k,c} \quad \text{for} \quad k = 1, \ldots, l \tag{6.7}$$

With $\mathbf{p}_{k,c}$ and $\lambda_{j,k,c}$ $\mathbf{x}_j$ can be fitted to $\bar{\mathbf{x}}_c$ resulting in a fitted trajectory $\mathbf{x}_{j,c}$.

$$\mathbf{x}_{j,c} = \bar{\mathbf{x}}_c + (\lambda_{j,1,c}\mathbf{p}_{1,c} + \lambda_{j,2,c}\mathbf{p}_{2,c}, \ldots, \lambda_{j,k,c}\mathbf{p}_{k,c}) \quad \text{for} \quad k = 1, \ldots, l \tag{6.8}$$

This results in six trajectories $\mathbf{x}_{j,1}, \ldots, \mathbf{x}_{j,6}$, one for each class c, which best fit the according ASMs of the six classes. For each fitted trajectory, the Euclidean distance to the mean trajectory of the according model, is calculated. The gesture is classified to prediction $\tilde{c}$ where the fitted trajectory $\mathbf{x}_{j,c}$ has the minimum distance to the mean trajectory $\bar{\mathbf{x}}_c$ of the model.

$$\tilde{c} = argmin_c(\sqrt{(\mathbf{x}_{j,c} - \bar{\mathbf{x}}_c)(\mathbf{x}_{j,c} - \bar{\mathbf{x}}_c)^T}) \tag{6.9}$$

Adaptive classification

The gesture classification can either be used in static or in adaptive mode. Adaptation of the gesture classifier is accomplished by a batch approach. The 30 training examples per class are kept in memory. When a new gesture instance with label c becomes available, it replaces the oldest example in the training memory of class c. A new ASM is built from the updated training data of class c. This continuously and incrementally adapts the gesture recognition to the characteristics of the user inputting the new gestures.

6.2.3. Experimental protocol

At the start of the experiment each subject was introduced to the system. We explained the operational principle of the motion tracking device, and stated that the gesture recognition is based on hand trajectories. The subjects were shown a printout of Figure 6.3 and given the chance to practice the gestures several times. The printout was removed before the experiment started. To allow for accurate segmentation, the subjects were instructed to perform the gestures at a reasonable speed and hold their hand still between gestures. The subjects were not informed about any static or adaptive behavior of the system.

For each subject the experiment was divided into the following four tasks T.

T1: **Offline data acquisition -** The subject randomly performs 30 gestures from each of the six classes outside the actual game application. The ground truth from each gesture is recorded together with the gesture trajectory. This task is a plain data recording without online gesture recognition and without feedback to the user. It is identical to the data recording from the initial training subjects.

T2: **Online use of the non-adaptive system -** The subject plays the memory game. Input gestures are recognized online based on the initially trained user independent classifier. After each gesture the subject indicates the ground truth of their last gesture. The gesture trajectories, the ground truth and the recognition results, are stored for later evaluation.

T3: **Online use of the adaptive system -** This task is similar to T2, except that the gesture recognition is adaptive. The ground truth

	Subject									
Task	1	2	3	4	5	6	7	8	AVG	STD
T1	93.9	69.4	69.4	71.1	85.6	80.6	60.6	78.3	76.1	10.6
T2	92.9	76.1	70.4	69.1	84.7	54.5	47.0	79.6	71.8	15.2
T3	93.8	95.8	94.8	93.0	98.9	93.7	86.0	91.1	93.4	3.7
T4	88.2	85.1	81.7	76.8	84.0	46.8	44.6	81.5	73.6	17.5

Table 6.1: Gesture recognition accuracies for all subjects and all experiment tasks in percentages.

information provided by the subject is used to continuously adapt the gesture recognition to the new user. The gesture trajectories, the ground truth and the recognition results, are stored for later evaluation.

T4: **Online use of the non-adaptive system -** This task is identical to T2. It is used as a verification task to ensure that potential improvements observed in T3 are not solely caused by user→system adaptation but actually the result of system→user adaptation. The recognition performance in this task is therefore expected to be similar compared to T2.

After T2, T3 and T4, the subjects fill in a questionnaire about their perception of the system's performance and their own behavior. In each of these tasks at least 150 gestures are performed per subject (6500 gestures in total for all subjects). Between all tasks the subjects took a short break of three to four minutes to recover.

We conducted the experiment with eight right-handed subjects aged 25 to 60. The experiment subjects are different from the subjects used for creating the initial ASM classifier. The duration of the whole procedure was about two hours per subject, including 15 minutes of system setup and procedure explanation.

6.3. Analysis of user→system adaptation

The recognition system can only recognize gesture trajectories that match the initially trained ASMs. We investigate in how far the user is capable of adapting their gesture execution in a way that is compliant with the recognition system. We refer to this as the user→system adaptation.

In T1 of the experiment the subject performs the gestures without obtaining feedback about how well their gestures match the gesture classifier. After the experiment the recorded gestures were classified offline, based on the user-independent classifier. The resulting accuracy acts as a baseline of the gesture recognition performance, when no adaptation, neither from the user nor from the system, is present.

In T2 and T4 the subject can see on the computer screen if the input gesture resulted in the intended action, and therefore if the gesture was correctly recognized. This information enables the subject to explore different gesture executions, in case gestures are not correctly recognized. The resulting gesture recognition accuracy for these tasks can be compared with the accuracy obtained from T1 to investigate the effect of user→system adaptation. Gesture recognition accuracy is calculated using

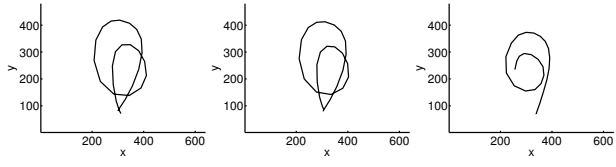
$$\text{accuracy} = \frac{\#\text{ gestures correctly classified}}{\#\text{ gestures performed}} \tag{6.10}$$

In Table 6.1 we list the accuracies for the gesture recognition for all subjects and all tasks. In average over all subjects the gesture recognition accuracy for T2 and T4 (71.8% and 73.6%) is lower compared to the T1 baseline (76.1%). A more detailed look reveals that this accuracy difference is not a trend valid for all subjects. Some subjects clearly benefit from the feedback and manage to adapt their gesture execution (e.g. Subject 2). For others the feedback doesn't have a noticeable effect (e.g. Subject 8) or even leads to a less system compliant gesture execution and worse recognition (e.g. Subject 7). This indicates that the gesture exploration, and the use of the feedback, varies between subjects. To further investigate this we focus on extreme examples of specific subjects.

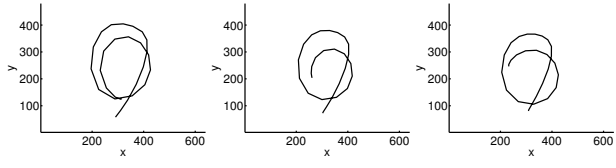
Examples for user→system adaptation

To get a better understanding of the user→system adaptation we look at two extreme cases in more detail (Subject 2 and Subject 7).

The gestures of class 6 performed by Subject 2 in T1 of the experiment were hardly recognized by the user-independent classifier (10.0% correctly classified). In contrast, the gestures of the same class were recognized much better in T2 (64.8% correctly classified). In Figure 6.5 we show the average hand trajectories for this case. For T1 there is a clear difference between the trajectories of the gestures incorrectly and



(a) T1: wrong (b) T1: all (c) T1: correct

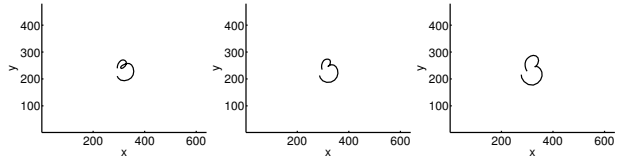


(d) T2: wrong (e) T2: all (f) T2: correct

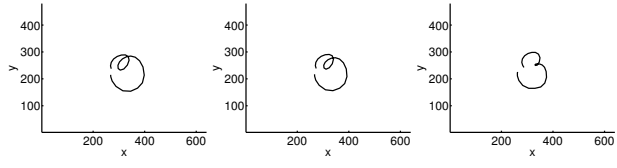
Figure 6.5: Trajectories for gesture class $c = 6$ of Subject 2 performed in T1 and T2. The trajectories shown are averages over all incorrectly classified gestures, all gestures and all correctly classified gestures.

correctly recognized (Figures 6.5 (a) and (c)). The main difference is the trajectory end point which is much lower for the incorrectly recognized gestures compared with those correctly recognized. In T2 the subject uses the feedback to learn a gesture execution that leads to a better recognition of class 6. The correctly recognized gestures still have a high trajectory end point compared to the incorrectly classified ones (Figures 6.5 (d) and (f)). For the incorrectly recognized gestures the trajectory end point in T2 is higher than it is in T1 (Figure 6.5 (a) and (d)). The average trajectory of all class 6 gestures has a higher trajectory end point in T2 compared to T1 (Figures 6.5 (e) and (b)). This indicates that the user adapted to the system's expectation of a higher trajectory end point.

The gestures of class 3 performed by Subject 7 in T1 were better recognized (43.3%) than the gestures of the same class during T2 (18.6%). In Figure 6.6 we show the average hand trajectories for this case. The average trajectory of the incorrectly classified gestures shows a loop between the upper and lower curve of the digit 3 (Figures 6.6 (a) and (d)). This loop is not present in the trajectories of the gestures that were correctly classified (Figures 6.6 (c) and (f)). Furthermore the incorrectly classified gestures have a smaller upper curve in the trajectory than the correctly classified ones. The lower recognition rate in



(a) T1: wrong (b) T1: all (c) T1: correct



(d) T2: wrong (e) T2: all (f) T2: correct

Figure 6.6: Trajectories for gesture class $c = 3$ of Subject 7 performed in T1 and T2. The trajectories shown are averages across all incorrectly classified gestures, all gestures (correct and incorrect) and all correctly classified gestures.

T2 indicates that the subject was not able to use the feedback from the system to adjust their movement strategy, to match the system's model. In contrast they performed the gestures even more in the way that was less recognizable, as can be seen on the average trajectory over all gestures of T1 and T2 (Figures 6.6 (b) and (e)). This might be caused by the subject exploring different gesture execution strategies without being able to find one that leads to a robust recognition.

6.4. Analysis of system→user adaptation

We investigate the online adaptation of the gesture recognition system towards the actual user. With each new gesture input the recognition system adapts incrementally to the user (T3 of the experiment).

The results listed in Table 6.1 clearly show the benefit of the system adaptation. For all subjects the adaptive system (T3) achieves a better recognition performance compared to the non-adaptive cases (T1, T2 and T4). In average the adaptive recognition system outperforms the non-adaptive system by about 20% in accuracy.

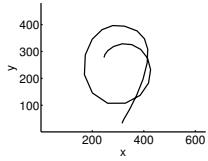
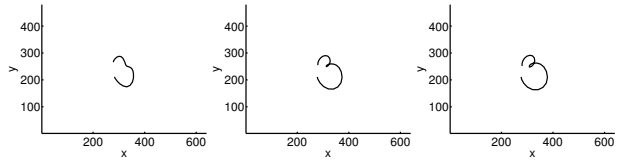


Figure 6.7: Gesture trajectory for Subject 2 and gesture 6 performed in T3. The trajectory shown is the average over all correctly classified gestures. For this subject all gestures were classified correctly in T3.



(a) T3: wrong (b) T3: all (c) T3: correct

Figure 6.8: Gesture trajectories for Subject 7 and gesture 3 performed in T3. The trajectories shown are averages over all incorrectly classified gestures, all gestures and all correctly classified gestures.

Examples for system→user adaptation

We look again at the two cases we already investigated in Section 6.3. Namely the gestures of class 6 from Subject 2 and the gestures of class 3 from Subject 7

In Figure 6.7 we show the average gesture trajectory for T3 of Subject 2. All gestures of class 6 were correctly recognized in this setting. The average correct gesture trajectory from T3 (Figure 6.7) is very similar to the average correct trajectory from T2 (Figure 6.5 (f)). This indicates that Subject 2 has learned to perform the gesture in a system compliant way during T2 and maintained this knowledge during T3 despite the system being adaptive.

The average trajectories of the class 3 gestures of Subject 7 performed during T3 of the experiment are shown in Figure 6.8. In this setting 79.0% of all gestures of class 3 were correctly recognized. The average trajectory for the correctly recognized gestures (Figure 6.8 (c)) shows a small loop between the upper and lower curve of the digit and a small upper curve. This is similar to the gesture trajectories of the

incorrectly classified gestures during T1 and T2 (Figures 6.6 (a) and (d)). The adapted recognition system of T3 was therefore capable of correctly classifying the gestures which were formerly misclassified. The average trajectory of all gestures of class 3 during T3 (Figure 6.8 (b)) is also similar to the average trajectory of T1 (Figure 6.6 (b)). This indicates that the adapted recognition system enables the user to fall back to their way of performing the gestures, while the system maintains a better recognition performance compared to T1 and T2.

6.5. Discussion

We carefully chose the application scenario and designed the experiment in a way, that eliminates potential error sources, and reduces the likelihood of misinterpretations. Nevertheless we discuss potential issues with our setup and our results.

6.5.1. On the experiment

The experiment design we chose in this chapter is not focused on an evaluation of a novel gesture recognition approach. Conversely, we intend to show the differences between the evaluation of an adaptive online recognition system on pre-recorded data and the evaluation within the actual online application. Our special focus hereby is on adaptation effects, from the user as well as from the system. Therefore we chose the optical hand trajectory tracking together with the ASM approach, allowing for a visualization of the gestures in 2D space. Our findings are not limited to optical motion tracking and should also be applicable to other sensor modalities that capture a form of motion or movements (for example an inertial measurement unit (IMU)).

The system adaptation is based on ground truth information provided by the user. Even though this approach works well for our purpose, it is not appropriate for an actual gesture recognition application. The ground truth feedback supersedes the purpose of the online recognition. Within an actual application a minimally obtrusive form of user feedback has to be chosen to guide the adaptation (see e.g. Chapter 5).

6.5.2. On user→system adaptation

The comparison of the gesture recognition accuracies in experiment T1 and T2 show major differences for several subjects, despite us taking special care to perform T1 and T2 as similarly to one another.

This shows that evaluations based on data recordings in a setting only slightly different from the actual application (e.g. without feedback or without involvement of the user in the task) may not apply to the actual application. When the user gets feedback from the system about the recognition, they may learn a specific behavior to meet the system's expectation. We observed such a user adaptation for several users. The effect and impact of the user adaptation is diverse throughout the subjects though. Even for the same subject we can observe different behaviors for different gesture classes.

When evaluating the system online within the target application other factors may also come into play. The memory game for example, played during T2 to T4, takes the attention of the user away from the gesture execution. This may lead to the user focusing less on how they perform the gestures and more on the actual game playing. This is an important aspect for such systems, as gesture execution should be intuitive, and not require much attention. The user may also get tired during the experiment due to its physical and mental demand. Lesser attention on the gesture execution or tiredness may result in a change of the gesture trajectory. We aimed at minimizing the effect of tiredness by giving subjects rest periods between experiment tasks.

6.5.3. On system→user adaptation

The adaptive recognition system, used in T3 of the experiment, provides a benefit for the user in form of a better gesture recognition. Furthermore, it allows the user to retain their own gesture execution style minimizing the effort of exploring other gesture movements which might be better recognized by the system. The adaptive behavior of the system can therefore increase the comfort for the user during system interaction. This finding is also supported by the questionnaire results. Six out of the eight subjects reported that they were able to focus more on the actual game playing during T3 compared to T2 and T4.

It is not possible to completely separate the system adaptation from the user adaptation though. The improvements observed in T3 of the experiment might be a result of combined adaptation efforts, from the system as well as from the user. Nevertheless, in our case there is evidence that the main increase in recognition accuracy is caused by the system adaptation. This is indicated by the fact that the recognition accuracy in T3 is higher compared to T4, even though the subject has more practice when performing T4, compared to T3. A system→user

adaptation has therefore clear benefits, even when the user is capable of adapting, too.

6.6. Conclusion

We investigated the user↔system co-adaptation of a gesture recognition system from the user as well as the system perspective. The adaptive gesture recognition is based on optical hand trajectory tracking combined with an ASM based classifier. In an online gesture recognition experiment we recorded more than 6500 gestures from eight subjects. The conditions covered by the experiment are:

1. A gesture data recording without feedback
2. A gesture controlled game playing with a static recognition system
3. A gesture controlled game playing with an adaptive recognition system

User→system adaptation

We observed that the subjects change their gesture execution when they become aware of the recognition result provided by the game application via the computer screen. The gestures recorded during the task, where no feedback was given to the users, were recognized with an accuracy of 76.1%. When the users get feedback on their gestures the average accuracies are 71.8% and 73.6%. It is a surprising result that the subjects are in average not able to benefit from the provided feedback, which contradicts our expectations. One reason for this could be, that in the feedback task the subjects focus mainly on the game playing, and concentrate less on the gesture execution compared to the no feedback task.

System→user adaptation

When the recognition system adapts to the user, based on ground truth provided, the gesture recognition accuracy increases to 93.4%. This increase in performance is likely to be caused by the system adapting to the user and confirms our expectation on an adaptive recognition system. This result is even more outstanding when taking into account

that the user→system adaptation, on average, did not lead to an improvement. From the user's point of view the benefit is not only in the increased recognition rate but also in the higher comfort, as they can perform gestures in ways most natural to them.

7

Conclusion

In this chapter we summarize and discuss the achievements of this thesis. In addition we provide an outlook for potential further research in the field of adaptive activity recognition.

7.1. Summary of achievements

Activity recognition is an important building block for context aware computing. The typical approach, inferring activities from motion sensor signals based on a static classifier model, has it's limitations. It cannot cope with non-stationary sensor signals caused by a different user behavior or novel characteristics of the environment. An activity recognition system that is intended to be used over long periods of time, in different environments, and by different users, has to be capable of adapting to such non-stationarities.

One possible solution we evaluated in this thesis is the use of incremental learning techniques. These techniques allow for the adaptation of a classifier by integrating new knowledge when it becomes available. The following achievements are part of our contributions to enable adaptive activity recognition and to make it more applicable to the real world.

- We proposed and evaluated three different incremental adaptation methods for activity recognition. They differ in the amount of supervision they require as well as in the learning principle they are based on.

  - The first approach (Chapter 3) is an unsupervised classifier self-calibration. It does not require any supervision from the user to adapt the classifier but instead capitalizes on structures in the input data. If the recognition accuracy before calibration is better than chance, an improvement by self-calibration can be expected. We have validated this finding for adaptation towards changes in the sensor system, namely the displacement of acceleration sensors on one limb segment. Simulations of the self-calibration on two datasets have shown an accuracy improvement of 33.3% and 13.4%, resulting in 63.5% and 74.4% of accuracy in case of a sligthly displaced sensor.
  - The second method (Section 4.2) is based on an incremental learning kNN classifier. We developed a novel incremental learning mechanism for a special form of supervision, namely error feedback. This feedback is given by the user whenever an activity instance is not classified correctly. In simulations on a real world gesture recognition dataset an improvement

of the recognition accuracy by 10.3% was achieved by adapting to a new user, compared to the user independent baseline accuracy of 68.3%.

  - The third method (Section 4.3) is based on Reinforcement Learning (RL) and also capitalizes on error feedback. In this case the error feedback from the user is translated into a reward signal. The RL aims to maximize the future expected reward by learning from past experiences. In contrast to the kNN approach the system based on RL can not only adapt to a new user but also learn the behavior of a user from scratch. We have shown this on a real world gesture recognition dataset. After 175 input gestures the system adapted to the new user achieving 90% of the maximum accuracy. Learning the behavior of a new user from scratch took a similar amount of input gestures and resulted in a similar recognition performance in comparison to the adaptation from a pre-trained system. The adaptation to a new user from a pretrained state increased the recognition performance by 36% compared to the baseline accuray of 46%.

- We conducted an online gesture recognition case study (Section 4.4) to validate the potential of incremental learning from error feedback. The participants of the study controlled a computer game by hand gestures which were recognized online. The adaptation of the gesture recognition was supervised by the user giving error feedback. We investigated three different learning scenarios: learning a new user behavior from scratch, adaptation to a displaced sensor and adaptation to a relocated sensor. In all three scenarios the learning and adaptation, based on the error feedback provided by the user, was successful and beneficial.

- We proposed a novel way for user feedback generation capitalizing on advances in EEG signal processing (Chapter 5). The user doesn't have to explicitly indicate a recognition error. Instead this information is extracted from the user's brain signals. This implicit user feedback is less obtrusive for the user since it doesn't require their attention. We conducted an online gesture recognition experiment to evaluate the potential of brain based error feedback generation. In simulations on the recorded brain signals it was possible to detect gesture recognition errors with above random accuracy. The average sensitivity and specificity

achieved were 0.60 and 0.56 respectively. Despite the imperfect error feedback it was still possible to adapt the user independent gesture classifier to a new user. The average gain in gesture recognition accuracy was 6.8%.

- We investigated the influence of an adaptive recognition system on the user's behavior in an online gesture recognition experiment (Chapter 6). The experiment covers three conditions: a gesture data recording, gesture controlled game playing with a static recognition system, and game playing with an adaptive recognition system. We observed that users changed their gesture execution when they became aware of the gesture recognition results via the game application. The users were not able to adapt their behavior to match the expectation of the static recognition system. In contrast the users' interaction with the adaptive system led to an improvement in the gesture recognition. There are indications that the adaptive system also enabled the users to perform the gestures in a way most convenient and natural to them.

7.2. Discussion

The adaptive methods we proposed and evaluated showed potential for the application of activity and gesture recognition. It must be noted though, that unsupervised adaptation has it's limitations since an improvement can not be guaranteed. In contrast the supervision provided by the error feedback reflects the user's intention and actively guides the adaptation.

The error feedback contains less information compared to ground truth feedback. This also influences the adaptation which might be slower or less accurate with error feedback. Nevertheless error feedback is easier to obtain and less obtrusive for the user, especially when the feedback is generated implicitly by the user.

With the proposed usage of EEG error related potentials for automatic implicit error feedback generation the subject doesn't have to explicitly focus on giving feedback. From this point of view it is even less obtrusive than an error button which has to be explicitly pushed. Considering the extensive measurement system required for capturing the user's brain signal, this approach is still not ready for every day use though.

When designing adaptive recognition systems the behavior of the user should also be taken into account. In a gesture recognition task we have seen that users adapt to the system they use. This leads to a user-system co-adaptation which is hard to predict and dependent on the application, the system setup, and the user.

7.3. Outlook

With the insights gained in this thesis we formulate further research directions of interest.

- The incremental learning algorithms we proposed assume that the error information provided for each instance is correct. We have already seen that imperfect error feedback, for example when it is extracted from brain signals, affects the adaptation performance. A learning method more robust to such imperfect feedback would lead to a better and also potentially faster adaptation.

- The error feedback could be combined with a confidence or quality value. For example this value could indicate how confident the system is, that the error feedback is actually correct. This confidence could be taken into account when learning and adapting, for example by weighting the instances accordingly, or by omitting the feedback when the confidence is too low.

- The supervision required could also be further reduced by integrating active learning methods [125]. In this case the idea is not to learn from all instances that become available but to choose those which seem to be most beneficial. Usually a label is requested for the instances that were selected for learning. These labels might also be replaced by error information.

- EEG signals are affected by contaminations from motion artifacts. In applications like activity or gesture recognition, where movements are required and wanted, this can lead to a reduced error recognition performance. An additional EMG measurement could be used to remove motion artifacts from the EEG signal to improve the error feedback.

- The implicit error feedback generation from brain signals was investigated with a complex and obtrusive EEG measurement device. Less obtrusive and even fully wearable EEG measurement

systems started to appear on the market lately. Such system could improve the comfort and make this approach usable in everyday life.

- So far we explored brain signals for an implicit generation of the error feedback from the user. Other physiological signals like the heart rate or the electrodermal activity could also provide valuable information for adaptation. A combination of these physiological signals could improve the quality of the error information provided.

- We investigated the co-adaptation behavior between an adaptive gesture recognition system and the user. In our case the recognition system operated on one fixed set of learning rate parameters. To date it is not clear how the user and the system behavior influence each other when a different learning rate is used, and therefore a different plasticity and stability of the system is given. A general understanding of this co-adaptation would be beneficial for the future design of any adaptive activity or gesture recognition system.

Glossary

Notation	Description
a	action
$a\hat{c}c$	accuracy estimate
α	balance parameter
β	rotation angle
c	activity/gesture class
$\tilde{c}$	class prediction
CC	class center
d	side length of square
$d(.)$	Euclidean distance
δ	stop learning parameter
e	distance based weight parameter
ε	exploration rate
$\mathbf{f}$	feature vector
i	index
K	set of all possible classes c
k	number of nearest neighbors
κ	removal weight threshold
$K(.)$	kernel function
l	learning neighborhood radius
λ	eigenvalue
LR	learning rate
M	classifier model
μ	mean
NN	neighborhood
O	EEG signal instance
$\mathbf{p}$	eigenvector
π	policy
$Q(.)$	Q-function
R	classification neighborhood radius
r	reward
S	state

Notation	Description
s	teacher signal
w	weight
Σ	covariance matrix
ST	stop learning threshold
SW	sum of weights
t	time
τ	time difference
τ_d	maximum density parameter
τ_k	kernel radius parameter
u	sensor position for training
v	sensor position for testing
W_{dec}	weight decrease function
W_{inc}	weight increase function
$\mathbf{x}$	activity/gesture instance

Abbreviations

Notation	Description
2D	2-dimensional
AVG	average
BCI	brain-computer interface
CE	*correct / error*
EEG	electroencephalography
EMG	electromyography
ERN	error related negativity
ErrP	error related potential
FRN	feedback related negativity
GNG	growing neural gas
HCI	human-computer interface
IMU	inertial measurement unit
kNN	k-Nearest-Neighbor
LC	learning case
NCC	nearest centroid classifier
NGN	neural gas network
RL	reinforcement learning
ROC	receiver operating characteristics
SAP	state-action pair
SI	subject independent
SOM	self organizing map
std	standard deviation
USB	universal serial bus

Bibliography

[1] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*, volume 4, pages 1–6. Citeseer, 2000.

[2] Jun Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, IMCE '09, pages 1–10, New York, NY, USA, 2009. ACM.

[3] Antonio Camurri and Volbe Gualtiero. *Gesture-Based Communication in Human-Computer Interaction*, volume 2915. Springer-Verlag, 2003.

[4] Christophe Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, 13(4):215–223, December 2000.

[5] Holger Junker. *Human Activity Recognition and Gesture Spotting with Body-Worn Sensors*. PhD thesis, ETH Zurich, 2005.

[6] E. Parzen. *Stochastic processes*, volume 24. Society for Industrial Mathematics, 1999.

[7] Mark G. Kelly, David J. Hand, and Niall M. Adams. The impact of changing populations on classifier performance. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 367–371, New York, NY, USA, 1999. ACM.

[8] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, 1996.

[9] Chunyu Yang and Jie Zhou. Non-stationary data sequence classification using online class priors estimation. *Pattern Recogn.*, 41(8):2656–2664, 2008.

[10] Alexey Tsymbal. The problem of concept drift: Definitions and related work. Technical report, Department of Computer Science, Trinity College, 2004.

[11] Ludmila I. Kuncheva. Classifier ensembles for changing environments. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *Multiple Classifier Systems*, volume 3077 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 0 2004.

[12] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vision Graph. Image Process.*, 37(1):54–115, 1987.

[13] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. *American Association for Artificial Intelligence*, 2005.

[14] Thomas G. Dietterich, Gerhard Widmer, and Miroslav Kubat. Special issue on context sensitivity and concept drift. *Mach. Learn.*, 32(2), 1998.

[15] J. del R. Millan. On the Need for On-Line Learning in Brain-Computer interfaces. In *Proc. of the Int. Joint Conf. on Neural Networks*, 2004.

[16] P. Shenoy, Matthias Krauledat, B. Blankertz, R. P. N. Rao, and K. R. Müller. TUTORIAL: Towards adaptive classification for BCI. *Journal of Neural Engineering*, 3, 2006.

[17] A. Buttfield, P. W. Ferrez, and J. del R. Millan. Towards a robust bci: error potentials and online learning. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):164–168, 2006.

[18] Matthias Krauledat. *Analysis of Nonstationarities in EEG Signals for Improving Brain-Computer Interface Performance*. PhD thesis, Technischen Universität Berlin, 2008.

[19] John C. Platt and Nada P. Mati'c. A constructive rbf network for writer adaptation. In *Advances in Neural Information Processing Systems*. MIT Press, 1997.

[20] Anja Brakensiek, Andreas Kosmala, and Gerhard Rigoll. Comparing adaptation techniques for on-line handwriting recognition. *icdar*, 00:0486, 0 2001.

[21] Alessandro Vinciarelli and Samy Bengio. Writer adaptation techniques in hmm based off-line cursive script recognition. *Pattern Recogn. Lett.*, 23(8):905–916, 2002.

[22] Scott D. Connell and Anil K. Jain. Writer adaptation for online handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):329–346, 0 2002.

[23] Randy Gomez, Akinobu Lee, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Improving Rapid Unsupervised Speaker Adaptation Based on HMM-Sufficient Statistics in Noisy Environments Using Multi-Template Models. *IEICE Trans Inf Syst*, E89-D(3):998–1005, 2006.

[24] Gyucheol Jang, Minho Jin, and Chgang D. Yoo. Speaker adaptation based on confidence-weighted training. In *Eurospeech*, pages 1617–1620, 2003.

[25] J. Neto, C. Martins, and L. Almeida. Unsupervised speaker-adaptation for hybrid hmm-mlp continuous speech recognition system, 1995.

[26] Yun Tang and R. Rose. Rapid speaker adaptation using clustered maximum-likelihood linear basis with sparse training data. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):607–616, 2008.

[27] Puming Zhan, Martin Westphal, Michael Finke, and Alex Waibel. Speaker normalization and speaker adaptation - a combination for conversational speech recognition. In *Proceedings of Eurospeech Conference*, pages 2087–2090, 1997.

[28] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Proc. of the 2nd Int Pervasive Computing Conference*, pages 1–17, 2004.

[29] Norbert Olivier and Ulrike Rockmann. *Grundlagen der Bewegunswissenschaft und -lehre.* Grundlagen der Sportwissenschaft. Verlag Karl Hofmann, 2003.

[30] David A Winter, Aftab E Patla, James S Frank, and Sharon E Walt. Biomechanical walking pattern changes in the fit and healthy elderly. *Phys Ther (United states)*, 70(6):340–347, 0 1990.

[31] Réjean Hébert, Carol Brayne, and David Spiegelhalter. Incidence of functional decline and improvement in a community-dwelling, very elderly population. *American Journal of Epidemiology*, 145(10):935–944, 1997.

[32] Jeffrey M Hausdorff, Chung-Kang Peng, Ary L. Goldberger, and Andrew L Stoll. Gait unsteadiness and fall risk in two affective disorders: a preliminary study. *BMC Psychiatry*, 4(39), 2004.

[33] Farid Parvini and Cyrus Shahabi. Utilizing bio-mechanical characteristics for user-independent gesture recognition. In *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, page 1170, Washington, DC, USA, 2005. IEEE Computer Society.

[34] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. In *Lecture Notes in Computer Science : Pervasive Computing*, pages 1–16, 2006.

[35] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge university press, 2006.

[36] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[37] C. Alippi and M. Roveri. Just-in-time adaptive classifiers in non-stationary conditions. In *International Joint Conference on Neural Networks*, pages 1014–1019, 2007.

[38] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[39] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[40] T. Martinetz and K. Schulten. A "neural gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier, Amsterdam, 1991.

[41] S. Mika, C. Schäfer, P. Laskov, Tax D., and K. R. Müller. *Handbook of Computational Statistics*, chapter 15, pages 841–876. Springer, Berlin, 2004.

[42] Qiang Huo and Chin-Hui Lee. On-line adaptive learning of the continuous density hidden markov model based on approximate recursive bayes estimate. *Speech and Audio Processing, IEEE Transactions on*, 5(2):161–172, 1997.

[43] Gianluigi Mongillo and Sophie Deneve. Online learning with hidden markov models. *Neural Computation*, 20:1706–1716, 2008.

[44] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. M. Buhmann. Topology free hidden markov models: application to background modeling. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 1:294–301 vol.1, 2001.

[45] J. C. Stiller and G. Radons. Online estimation of hidden markov models. *Signal Processing Letters, IEEE*, 6(8):213–215, 1999.

[46] Jun Mizuno, Tasuya Watanabe, Kazuya Ueki, Kazuyuki Amano, Eiji Takimoto, and Akira Maruoka. On-line estimation of hidden markov model parameters. In *DS '00: Proceedings of the Third International Conference on Discovery Science*, pages 155–169, London, UK, 2000. Springer-Verlag.

[47] L Cohen, G Avrahami, M Last, A Kandel, and O Kipersztok. Incremental classification of nonstationary data streams. In *Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, pages 117–124, 2005.

[48] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, us ed edition, May 2005.

[49] Bernd Fritzke. A self-organizing network that can follow nonstationary distributions. In *ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks*, pages 613–618, 1997.

[50] R. Salas, S. Moreno, H. Allende, and C. Moraga. A robust and flexible model of hierarchical self-organizing maps for non-stationary environments. *Neurocomput.*, 70(16-18):2744–2757, 2007.

[51] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[52] V Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[53] R. Dara, S. C. Kremer, and D. A. Stacey. Clustering unlabeled data with soms improves classification of labeled real-world data. *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, 3:2237–2242, 2002.

[54] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. The MIT press, 1998.

[55] T. Artieres and P. Gallinari. Stroke level hmms for on-line handwriting recognition. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 227 – 232, 2002.

[56] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. On-line adaptation in recognition of handwritten alphanumeric characters. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 792 –795, sep 1999.

[57] Ali Nosary, Laurent Heutte, and Thierry Paquet. Unsupervised writer adaptation applied to handwritten text recognition. *Pattern Recognition*, 37(2):385 – 388, 2004.

[58] S. D. Connell and N. K. Jain. Writer adaptation of online handwriting models. In *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pages 434 –437, 1999.

[59] G. G. Molina. Bci adaptation using incremental-svm learning. In *Neural Engineering, 2007. CNE '07. 3rd International IEEE/EMBS Conference on*, pages 337 –341, may 2007.

[60] C. Vidaurre, A. Schlogl, R. Cabeza, R. Scherer, and G. Pfurtscheller. A fully on-line adaptive bci. *Biomedical Engineering, IEEE Transactions on*, 53(6):1214 –1219, june 2006.

[61] Pradeep Shenoy, Matthias Krauledat, Benjamin Blankertz, Rajesh P N Rao, and Klaus-Robert Müller. Towards adaptive classification for bci. *Journal of Neural Engineering*, 3(1):R13, 2006.

[62] Ernest J. Pusateri and Timothy J. Hazen. Rapid speaker adaptation using speaker clustering. In *Proc. of ICASSP*. unknown, 2002.

[63] X. He and Y. Zhao. Fast model selection based speaker adaptation for nonnative speech. *IEEE Trans. on Speech and Audio Processing*, 11(4):298–307, 2003.

[64] J. M. Baker, L. Deng, J. Glass, S. Khudanpur, Lee C.-H., N. Morgan, and D. O'Shaughnessy. Research developments and directions in speech recognition and understanding, part 1. *IEEE Signal Processing Magazine*, 26(3):75–80, 2009.

[65] R. Ohmura, N. Hashida, and M. Imai. Preliminary evaluation of personal adaptation techniques in accelerometer-based activity recognition. In *Proc. 13th IEEE Int. Symposium on Wearable Computers: Late Breaking Results*, 2009.

[66] T. Maekawa and S. Watanabe. Unsupervised activity recognition with user's physical characteristics data. In *Proc. International Symposium on Wearable Computers (ISWC 2011)*, 2011.

[67] Kai Kunze and P. Lukowicz. Using acceleration signatures from everyday activities for on-body device location. In *Wearable Computers11th IEEE International Symposium on Wearable Computers*, pages 115–116, 2007.

[68] Kai Kunze and Paul Lukowicz. Dealing with sensor displacement in motion based onbody activity recognition systems. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 20–29, 2008.

[69] K. Höök. Evaluating the utility and usability of an adaptive hypermedia system. *Knowledge-Based Systems*, 10(5):311–319, 1998.

[70] Stephan Weibelzahl. *Evaluation of Adaptive Systems*. PhD thesis, University of Education Freiburg, 2002.

[71] C. Gena. Methods and techniques for the evaluation of user-adaptive systems. *The knowledge engineering review*, 20(01):1–37, 2005.

[72] W. E. Mackay. From gaia to hci: On multi-disciplinary design and co-adaptation. 2008.

[73] W. E. Mackay. *Users and customizable software: A co-adaptive phenomenon*. PhD thesis, Massachusetts Institute of Technology, 1990.

[74] B. Mahmoudi, J. DiGiovanna, J. C. Principe, and J. C. Sanchez. Co-adaptive learning in brain-machine interfaces. *Brain inspired cognitive systems. Sao Luis, Brazil*, 2008.

[75] Ryu Kato, F. Matsumoto, H. Yokoi, T. Arai, and T. Onishi. Co-adaptation system between human and machine in welfare robotics-development of adaptive emg prosthetic hand. *Nippon Kikai Gakkai Robotikusu, Mekatoronikusu Koenkai Koen Ronbunshu (CD-ROM)*, 2006:2P2–A18, 2006.

[76] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[77] C. Alippi and M. Roveri. A computational intelligence-based criterion to detect non-stationarity trends. In *International Joint Conference on Neural Networks*, pages 5040–5044, 2006.

[78] C. Alippi and M. Roveri. An adaptive cusum-based test for signal change detection. In *Proceedings of IEEE International Symposium on Circuits and Systems*, 2006.

[79] Cornelia Setz, Bert Arnrich, Johannes Schumm, Roberto La Marca, Gerhard Tröster, and Ulrike Ehlert. Discriminating stress from cognitive load using a wearable eda device. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):410–417, 2010.

[80] Ashish Kapoor, Winslow Burleson, and Rosalind W. Picard. Automatic prediction of frustration. In *International Journal of Human Computer Studies*, 2007.

[81] R. S. Lazarus. *Psychological stress and the coping process*. McGraw Hill, New York, 1966.

[82] R. Chavarriaga, P. W. Ferrez, and J. del R. Millán. To err is human: Learning from error potentials in brain-computer interfaces. In *International Conference on Cognitive Neurodynamics*, 2007.

[83] Avrim Blum. *Online Algorithms: The State of the Art*, chapter 4. Fiat and Woeginger eds., 1998. 5.

[84] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. Reinforcement Learning and Dynamic Programming using Function Approximators. *Automation And Control Engineering*, page 280, 2010.

[85] Richard Bellman. *Dynamic Programming*. Princeton University Press, 2010.

[86] G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[87] P. R. Roelfsema and A. Ooyen. Attention-gated reinforcement learning of internal representations for classification. *Neural Computation*, 17(10):2176–2214, 2005.

[88] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. Citeseer, 1994.

[89] V. Rieser and M. Pinkal. *Bootstrapping Reinforcement Learning-based Dialogue Strategies from Wizard-of-Oz data*. German Research Center for Artificial Intelligence, 2008.

[90] S. Le Groux and P. F. M. J. Verschure. Towards adaptive Music Generation By Reinforcement Learning of Musical Tension. In *SMC Conference 2010*, 2010.

[91] N. Lilith and K. Dogançay. Reduced-state SARSA featuring extended channel reassignment for dynamic channel allocation in mobile cellular networks. *Networking-ICN 2005*, pages 531–542, 2005.

[92] J. C. Santamaria, R. S. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive behavior*, 6(2):163, 1997.

[93] C. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.

[94] J. Peng and R. J. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22(1):283–290, 1996.

[95] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[96] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, pages 363–372, 2006.

[97] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1994.

[98] K. Swingler. *Applying neural networks: a practical guide*. Morgan Kaufmann, 1996.

[99] ez430-chronos development tool user's guide (rev. c). Technical report, Texas Instruments, 2010.

[100] S. Nieuwenhuis, K. R. Ridderinkhof, J. Blom, G. P. Band, and A. Kok. Error-related brain potentials are differentially related to awareness of response errors: Evidence from an antisaccade task. *Psychophysiology*, 38(5):752–760, Sep 2001.

[101] A. Yasuda, A. Sato, K. Miyawaki, H. Kumano, and T. Kuboki. Error-related negativity reflects detection of negative reward prediction error. *Neuroreport*, 15(16):2561–2565, Nov 15 2004.

[102] Michael J Frank, Brion S Woroch, and Tim Curran. Error-related negativity predicts reinforcement learning and conflict biases. *Neuron*, 47(4):495–501, Aug 2005.

[103] Stephan F Taylor, Emily R Stern, and William J Gehring. Neural systems for error monitoring: Recent findings and theoretical perspectives. *Neuroscientist*, 13(2):160–172, 2007.

[104] M. Falkenstein, J. Hoormann, S. Christ, and J. Hohnsbein. ERP components on reaction errors and their functional significance: A tutorial. *Biol Psychol*, 51(2-3):87–107, Jan 2000.

[105] Pierre W. Ferrez and José del R. Millán. Error-related EEG potentials generated during simulated brain-computer interaction. *IEEE Trans Biomed Eng*, 55:923–929, 2008.

[106] G. Schalk, J. R. Wolpaw, D. J. McFarland, and G. Pfurtscheller. EEG-based communication: Presence of an error potential. *Clin Neurophysiol*, 111(12):2138–2144, Dec 2000.

[107] L. C. Parra, C. D. Spence, A. D. Gerson, and P. Sajda. Response error correction–A demonstration of improved human-machine performance using real-time EEG monitoring. *IEEE Trans Neural Syst Rehabil Eng*, 11(2):173–177, June 2003.

[108] Mehrdad Fatourechi, Ali Bashashati, Rabab K Ward, and Gary E Birch. EMG and EOG artifacts in brain computer interface systems: A survey. *Clin Neurophysiol*, 118(3):480–494, Mar 2007.

[109] Scott Makeig, Klaus Gramann, Tzyy-Ping Jung, Terrence J. Sejnowski, and Howard Poizner. Linking brain, mind and behavior. *International Journal of Psychophysiology*, 73(2):95 – 100, 2009. Neural Processes in Clinical Psychophysiology.

[110] Jean-Marc Bollon, Ricardo Chavarriaga, José del R. Millán, and Pierre Bessière. EEG error-related potentials detection with a Bayesian filter. In *4th International IEEE EMBS Conference on Neural Engineering*, Antalya Turkey, 2009.

[111] A. H. Jazwinski. *Stochastic processes and filtering theory*, volume 63. Academic Pr, 1970.

[112] A. Schlögl, C. Keinrath, D. Zimmermann, R. Scherer, R. Leeb, and G. Pfurtscheller. A fully automated correction method of EOG artifacts in EEG recordings. *Clin Neurophysiol*, 118(1):98–104, Jan 2007.

[113] Huan Liu and Rudy Setiono. A probabilistic approach to feature selection - a filter solution. In *ICML'96*, pages 319–327, 1996.

[114] Felix Garci'a Lopez, Miguel Garci'a Torres, Belen Melian Batista, Jose A. Moreno Perez, and J. Marcos Moreno-Vega. Solving feature subset selection problem by a parallel scatter search. *European Journal of Operational Research*, 169(2):477–489, March 2006.

[115] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition, October 1999.

[116] José del R Millán, Frédéric Renkens, Josep Mouriño, and Wulfram Gerstner. Noninvasive brain-actuated control of a mobile robot by human EEG. *IEEE Trans Biomed Eng*, 51(6):1026–1033, 2004.

[117] A. J. Casson, S. Smith, J. S. Duncan, and E. Rodriguez-Villegas. Wearable EEG: what is it, why is it needed and what does it entail? In *Proc IEEE Eng Med Biol Soc.*, pages 5867–5870, 2008.

[118] Kurt Andrew Thoroughman. *Human motor learning in stationary and nonstationary novel dynamic environments: psychophysical, electromyographical, and computational verification and extension of the inverse model hypothesis*. PhD thesis, Johns Hopkins University, 1999.

[119] Thomas M. Brashers-Krug. *Consolidation in Human Motor Learning*. PhD thesis, Massachusetts Institute of Technology, 1995.

[120] K. C. Santosh and Cholwich Nattee. A comprehensive survey on on-line handwriting recognition technology and its real application to the nepalese natural handwriting. *Kathmandu University Journal of Science, Engineering and Technology*, 2009.

[121] Stefano Nolfi and Dario Floreano. Coevolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artif. Life*, 4:311–335, October 1998.

[122] Oliver Amft, Roman Amstutz, Asim Smailagic, Daniel Siewiorek, and Gerhard Tröster. Gesture controlled user input to complete questionnaires on wristworn watches. In *HCII 2009: Proceedings of the 13th International Conference on Human-Computer Interaction*, volume 5611 of *Lecture Notes in Computer Science*, pages 131–140. Springer, 2009.

[123] F. Fang, Y. Xu, and CS Chen. Gesture interface: Modeling and learning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1747–1752. IEEE, 1994.

[124] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, et al. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[125] B. Settles. Active learning literature survey. *Machine Learning*, 15(2), 1994.