Bayesiaanse zelflerende algoritmes
voor kalibratieloze brein-computer-interfaces

A Bayesian Machine Learning Framework
for True Zero-Training Brain-Computer Interfaces

Pieter-Jan Kindermans

Promotor: prof. dr. ir. B. Schrauwen
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2013 - 2014

ISBN 978-90-8578-687-0
NUR 984
Wettelijk depot: D/2014/10.500/33

Acknowledgements

I would like to thank my supervisor Benjamin Schrauwen for giving me the opportunity, financial support and freedom to perform this research. He also taught us that, if you want to achieve something, you just have to go for it (and work hard to get to that goal). I am grateful to the other members of the PhD committee Jan Van Campenhout, Joni Dambre, Willem Waegeman, Klaus-Robert Müller and Michael Tangermann for their careful proofreading, discussions and suggestions to improve this dissertation.

Furthermore, without Klaus and Michael, this dissertation would be rather different (and less substantial). At the Machine Learning Summer School in Kyoto, Klaus invited me to visit his lab at the TU-Berlin. During this stay, we had not only several interesting and productive discussions, we also performed a thorough online evaluation of the proposed methods. For this, I collaborated closely with Michael, whose experience with online experiments was invaluable. Finally, I would like to thank Martijn for his help in getting the online experiment running, Arash and Jan-Eike for performing most of the online experiments.

Obviously, there was also a lot of support for the BCI research from our own Lab. Pieter, the go-to guinea pig, and David, thanks for listening patiently to my ideas and for proofreading a pile of manuscripts. Hannes, who was a master student in our lab, went far beyond what we could expect of him. His aid in getting the first online demonstration of the unsupervised speller working was greatly appreciated. I am grateful to Thibault, who joined us last summer,

for all his help related to the BCI work and the collaboration. I would also like to thank two people outside of the lab: Dieter Devlaminck, who guided us during our first real BCI experiments, and Patrick Santens, who lends us an EEG amplifier.

I would like to thank Sander, with whom I have shared an office for almost 4 years. There is simply no better person to have in the office with you. His calm nature, patience and cluster-maintenance work were greatly appreciated. Ken, with whom I started my university studies almost 9 years ago, is the polar opposite. Nevertheless, his remarks (in his own and rather unique style) and the interesting discussions were always welcome. Francis is the cornerstone of the lab. Not only do we appreciate his brownie and ice-cream making skills, he also ensures everything (from student affairs to the robot competition) runs smoothly. A different but still vital role in the lab is taken up by Aaron, my ET nemesis, who schedules all the mini team-building events. Obviously, the lab would not be the same without the other (some former) members. Therefore, I would also like to thank Tim, Michiel H., Philémon, Juan Pablo, Michiel D., Jonas and Michael.

I appreciate the hard work that goes on behind the scenes. Without our local SAP guru Marnix and the administrative/technical staff Ronny, Jeroen, etc., we would not be able to perform our research.

I am grateful for my friends Valerie, Thomas, Nele, Nausikaä, Mira, Alex, for making attempts to keep me sane, for providing much appreciated funny moments, for their support and for their help.

Finally, I would like to thank my parents. Without them, I would never have gotten to this point. Their support was simply limitless.

Pieter-Jan Kindermans

Examencommissie

prof. Jan Van Campenhout, voorzitter
Vakgroep Elektronica en Informatiesystemen
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

prof. Joni Dambre, secretaris
Vakgroep Elektronica en Informatiesystemen,
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

prof. Benjamin Schrauwen, promotor
Vakgroep Elektronica en Informatiesystemen
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

prof. Willem Waegeman
Vakgroep Wiskundige modellering, Statistiek en Bio-informatica
Faculteit Bio-ingenieurswetenschappen
Universiteit Gent

prof. Klaus-Robert Müller
Department of Software Engineering and Theoretical Computer Science
Fakultät Elektrotechnik und Informatik
Technische Universität Berlin

dr. Michael Tangermann
Institut für Informatik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Samenvatting

Dit werk is een uiteenzetting over de ontwikkeling van niet-gesuperviseerd lerende algoritmes voor een brein-computer-interfaces (BCI) gebaseerd op gebeurtenis-gerelateerde potentialen (ERP) (in het Engels Event-Related Potential).

Brein-computer-interfaces

Een BCI heeft als doel een gebruiker de mogelijkheid te geven om een computer (bv. een tekstverwerker) of randapparatuur (bv. een robotarm) rechtstreeks aan te sturen met hersensignalen. Het BCI concept bestaat sinds 1973. De eerste generatie van BCI's vertrouwde op de gebruiker om zijn hersensignalen zo te controleren zodat de computer de juiste actie uitvoerdde. Het probleem met deze aanpak is dat de gebruiker een lange training moet ondergaan om te leren hoe de hersensignalen te controleren. De huidige generatie BCI's daarentegen is gebouwd rond het principe dat de machine moet leren de hersensignalen van de gebruiker te begrijpen. Om dit te bewerkstelligen worden lerende algoritmes gebruikt, die op basis van voorbeelden getraind worden om de specifieke hersenactiviteit te herkennen. Het verzamelen van deze voorbeelden vereist echter een omslachtige procedure. Deze bestaat uit een kalibratieopname, waarbij de gebruiker op specifieke tijdstippen een mentale opdracht moet uitvoeren. Voorbeelden van zulke opdrachten zijn zich concentreren op een stimulus, of het zich inbeelden van een beweging. Omdat men tijdens deze opname weet wat de gedachten van de gebruiker zijn, kan de

opgenomen data gemarkeerd worden. Deze gemarkeerde data wordt dan gebruikt om het computerprogramma te trainen in het herkennen van de hersensignalen. Natuurlijk is zo een kalibratieopname niet productief. Daarom wordt er onderzoek gevoerd naar methodes die deze kalibratie tot een minimum beperken of zelfs het volledig overslaan ervan mogelijk maken. Dit is natuurlijk bijzonder belangrijk voor patiënten met een beperkte aandachtsspanne.

Een BCI op basis van ERP

Een van de meest gebruikte types BCI is de ERP gebaseerde BCI. Deze BCI werd reeds uitgevonden in 1988 door Farwell en Donchin. De BCI werkt als volgt. De gebruiker krijgt een stroom van stimuli te verwerken. Iedere stimulus is gekoppeld aan een specifieke actie. Wanneer de gebruiker zich concentreert op een bepaalde stimulus, dan zal er, na de presentatie van deze stimulus, een ERP doelrespons worden waargenomen in het EEG. Als de computer er in slaagt deze respons te herkennen, dan kan de correcte actie uitgevoerd worden. Bijgevolg slaagt de persoon er dus effectief in om de computer te besturen door middel van zijn hersensignalen. Oorspronkelijk waren de stimuli het oplichten van rijen en kolommen in een raster van letters op het scherm. Door zich simpelweg op de gewenste letter te concentreren kan de gebruiker deze letter schrijven. Hedendaagse versies van dit paradigma werken echter ook met auditieve of tactiele stimuli. Daarbovenop zijn de toepassingen voor dit type BCI ook uitgebreid. Bovenop het oorspronkelijke spellingsprogramma is het nu ook mogelijk om dit soort BCI te gebruiken om te tekenen, een robot te besturen, op het werelwijde web te surfen, enz. .

Een coherent model op basis van waarschijnlijkheidsleer

Over de jaren heen hebben onderzoekers nieuwe technieken ontwikkeld om de ERP-BCI preciezer en sneller te maken. Het resultaat van dit onderzoek is een immer complexer wordend aanbod aan lerende algoritmes. Spijtig genoeg dienen deze algoritmes voor ieder individu geoptimaliseerd te worden. De huidige herkenners gebruiken niet enkel de hersensignalen, maar ook randinformatie zoals de voorheen gespelde tekst. Verder is een van de meest bruikbare uitbreidingen

het gebruik van een zogenaamde "dynamic stopping" strategie. Hierbij wordt de presentatie van de stimuli onderbroken op het moment dat de herkenner voldoende zeker is over de gewenste stimulus.

Deze uitbreidingen staan echter vaak op zichzelf. Bijgevolg wordt extra complexiteit geïntroduceerd wanneer de verschillende onderdelen moeten worden samengebracht. Dit brengt extra parameters met zich mee. Het resultaat is dat deze technieken moeilijk te optimaliseren zijn en daarvoor extra gemarkeerde data nodig hebben.

Wij hebben dit probleem verholpen door een probabilistisch model voor te stellen dat al deze technieken op een samenhangende wijze kan combineren. Het resultaat is een herkenner die zeer efficiënt werkt, heel accuraat is, maar nog steeds eenvoudig te optimaliseren is. Het nadeel van deze methode is echter dat een verzameling van gemarkeerde voorbeelden per gebruiker nog steeds vereist is.

Een zelf-lerend algoritme voor ERP-BCI

Een tweede bijdrage van dit werk is de ontwikkeling van een zelf-lerend algoritme voor de ERP-BCI. In tegenstelling tot de typische herkenners moet dit model niet getraind worden met gelabelde voorbeelden. Het zelf-lerend algoritme is in staat om, door de data te analyseren, uit te vissen hoe de hersensignalen gestructureerd zijn. Dat laat toe dat een nieuwe gebruiker de BCI kan benutten zonder voorafgaande kalibratie. Initieel maakt het zelf-lerend model nog een aantal fouten, maar tijdens het gebruik ervan wordt meer en meer data geanalyseerd. Uiteindelijk zal dit leiden tot een zeer betrouwbare herkenner die in staat is zijn eigen fouten te corrigeren.

Deze nieuwe methode is zeer uitgebreid getest, zowel in simulatie als tijdens zogenaamde online BCI-experimenten. Onze simulatieresultaten werden uitgevoerd op drie verschillende verzamelingen van data. Samen bevatten deze de data van de hersenactiviteit van 25 verschillende personen. Deze personen gebruikten bij de opname het originele ERP-BCI-systeem. Bovenop deze eerste simulatie hebben we ook een tweede test uitgevoerd op data van 21 personen die een auditieve BCI gebruiken. Uiteindelijk werd dit gevolgd door een echt BCI-experiment uitgevoerd in samenwerking met Michael Tangermann, Martijn Schreuder en Klaus-Robert Müller aan de Technische Univer-

siteit in Berlijn.

Onze experimenten hebben aangetoond dat het effectief mogelijk is om een zelf-lerende herkenner te bouwen die even accuraat is als een herkenner die op basis van voorbeelden mocht leren. Een zelf-lerende herkenner bouwen is uiteraard een veel grotere uitdaging. Verder kan een zelf-lerend systeem zich aanpassen aan wijzigingen in de structuur van de data. Dit kan gebeuren wanneer de gebruiker vermoeid raakt. Een standaard op voorbeelden getraind systeem zal dan slechter beginnen werken. Een zelf-lerend systeem kan echter deze wijzigingen detecteren en een nieuw verbeterd model bouwen.

Het delen van kennis over personen heen

Om een systeem te bouwen dat vanaf het begin betrouwbaar is hebben wij ons coherent probabilistisch model uitgebreid met een module die toelaat om kennis over hoe de hersensignalen te decoderen te delen over verschillende personen. Dit stelt ons in staat om een algemene herkenner te bouwen die voor verschillende personen bruikbaar is. Het nadeel van zo een algemene herkenner is dat deze niet even precies werkt als een persoonsgebonden herkenner. Echter, door dit algemeen model te combineren met het zelf-lerende algoritme zijn we er in geslaagd een BCI te bouwen die meteen werkt, zonder enige vorm van training. We hebben dit eveneens zeer uitgebreid geëvalueerd. Zowel in een werkende experimentele opstelling als in een simulatie.

Het begrijpen van het zelf-lerend en kennis-delend model

De laatste wetenschappelijke bijdrage bestaat uit een theoretische analyse van het delen van decodeer-informatie over de gebruikers heen. De traditie bij BCI-onderzoekers is om persoon-specifieke modellen te gebruiken. Deels komt deze traditie door het feit dat de hersensignalen drastisch kunnen verschillen tussen de personen. Wij tonen echter aan dat dit verschil eerder oppervlakkig is. De informatie die in deze signalen is bevat bevindt zich immers in een kleine deelruimte van de gemeten signalen. Bovendien zijn de signalen in deze deelruimte zeer gelijkaardig over de verschillende personen heen. In onze kennis-delende aanpak wordt dit principe uitgebuit.

Summary

Brain-Computer Interfaces

Brain-Computer Interfaces (BCI) are developed to allow the user to take control of a computer (e.g. a spelling application) or a device (e.g. a robotic arm) by using just his brain signals. The concept of BCI was introduced in 1973 by Jacques Vidal. The early types of BCI relied on tedious user training to enable them to modulate their brain signals such that they can take control over the computer. Since then, training has shifted from the user to the computer. Hence, modern BCI systems rely on a calibration session, during which the user is instructed to perform specific tasks. The result of this calibration recording is a labelled data-set that can be used to train the (supervised) machine learning algorithm. Such a calibration recording is, however, of no direct use for the end user. Hence, it is especially important for patients to limit this tedious process. For this reason, the BCI community has invested a lot of effort in reducing the dependency on calibration data. Nevertheless, despite these efforts, true zero-training BCIs are rather rare.

Event-Related Potential based spellers

One of the most common types of BCI is the Event-Related Potentials (ERP) based BCI, which was invented by Farwell and Donchin in 1988. In the ERP-BCI, actions, such as spelling a letter, are coupled to specific stimuli. The computer continuously presents these stimuli to the user. By attending a specific stimulus, the user is able

to select an action. More concretely, in the original ERP-BCI, these stimuli were the intensifications of rows and column in a matrix of symbols on a computer screen. By detecting which row and which column elicit an ERP response, the computer can infer which symbol the user wants to spell. Initially, the ERP-BCI was aimed at restoring communication, but novel applications have been proposed too. Examples are web browsing, gaming, navigation and painting. Additionally, current BCIs are not limited to using visual stimuli, but variations using auditory or tactile stimuli have been developed as well.

A unified probabilistic model

In their quest to improve decoding performance in the ERP-BCI, the BCI community has developed increasingly more complex machine learning algorithms. However, nearly all of them rely on intensive subject-specific fine-tuning. The current generation of decoders has gone beyond a standard ERP classifier and they incorporate language models, which are similar to a spelling corrector on a computer, and extensions to speed up the communication, commonly referred to as dynamic stopping. Typically, all these different components are separate entities that have to be tied together by heuristics. This introduces an additional layer of complexity and the result is that these state of the art methods are difficult to optimise due to the large number of free parameters. We have proposed a single unified probabilistic model that integrates language models and a natural dynamic stopping strategy. This coherent model is able to achieve state of the art performance, while at the same time, minimising the complexity of subject-specific tuning on labelled data.

Unsupervised training for ERP-BCI

A second and major contribution of this thesis is the development of the first unsupervised decoder for ERP spellers. Recall that typical decoders have to be tuned on labelled data for each user individually. Moreover, recording this labelled data is a tedious process, which has no direct use for the end user. The unsupervised approach, which is an extension of our unified probabilistic model, is able to learn how to

decode a novel user's brain signals without requiring such a labelled dataset. Instead, the user starts using the system and in the meantime the decoder is learning how to decode the brain signals.

This method has been evaluated extensively, both in an online and offline setting. Our offline validation was executed on three different datasets of visual ERP data in the standard matrix speller. Combined, these datasets contain 25 different subjects. Additionally, we present the results of an offline evaluation on auditory ERP data from 21 subjects. Due to a less clear signal, this auditory ERP data present an even greater challenge than visual ERP data. On top of that we present the results from an online study on auditory ERP, which was conducted in cooperation with Michael Tangermann, Martijn Schreuder and Klaus-Robert Müller at the TU-Berlin.

Our simulations indicate that when enough unlabelled data is available, the unsupervised method can compete with state of the art supervised approaches. Furthermore, when non-stationarity is present in the EEG recordings, e.g. due to fatigue during longer experiments, then the unsupervised approach can outperform supervised methods by adapting to these changes in the data. However, the limitation of the unsupervised method lies in the fact that while labelled data is not required, a substantial amount of unlabelled data must be processed before a reliable model can be found. Hence, during online experiments the model suffers from a warm-up period. During this warm-up period, the output is unreliable, but the mistakes made during this warm-up period can be corrected automatically when enough data is processed.

Transfer Learning for ERP-BCI

To maximise the usability of ERP-BCI, the warm-up of the unsupervised method has to be minimised. For this reason, we propose one of the first transfer learning methods for ERP-BCI. The idea behind transfer learning is to share information on how to decode the brain signals between users. The concept of transfer learning stands in stark contrast with the strong tradition of subject-specific decoders commonly used by the BCI community. Nevertheless, by extending our unified model with inter-subject transfer learning, we are able to build

a decoder that can decode the brain signals of novel users without any subject-specific training. Unfortunately, basic transfer learning models do perform as well as subject-specific (supervised models). For this reason, we have combined our transfer learning approach with our unsupervised learning approach to adapt it during usage to a highly accurate subject-specific model.

Analogous to our unsupervised model, we have performed an extensive evaluation of transfer learning with unsupervised adaptation. We tested the model offline on visual ERP data from 22 subjects and on auditory ERP data from 21 subjects. Additionally, we present the results from an online study, which was also performed at the TU-Berlin, where we evaluate transfer learning online on the auditory AMUSE paradigm.

From these experiments, we can conclude that transfer learning in combination with unsupervised adaptation results in a true zero training BCI, that can compete with state of the art supervised models, without needing a single data point from a calibration recording. This method allows us to build a BCI that works out of the box.

Understanding transfer learning

Our final contribution considers a more theoretical evaluation of transfer learning to enhance our understanding of transfer learning in ERP-BCI. This is especially important given the high variability observed in the data of different users, which has partially driven the push for subject-specific models within the BCI community. Our analysis has shown that while the recorded brain signals appear to be distinct, the information present in the signal is limited to only a specific subspace of the signal. Moreover, this informative subspace of the signal can be shared across users.

List of Abbreviations

ALS	Amyotrophic Lateral Sclerosis
AUC	Area Under Curve in a Receiver Operator Characteristic
BCI	Brain-Computer Interface
BLDA	Bayesian Linear Discriminant Analysis
BLR	Bayesian Linear Regression
CNN	Convolutional Neural Network
CSP	Common Spatial Patterns
EEG	Electroencephalogram
EM	Expectation Maximisation
ECoG	Electrocorticogram
ERP	Event-Related Potential
FMRI	Functional Magnetic Resonance Imaging
GMM	Gaussian Mixture Model
ITR	Information Transfer Rate
ISI	Inter-Stimulus Interval
kPCA	Kernel Principal Component Analysis
LDA	Linear Discriminant Analysis
LM	Language Model
MAP	Maximum A Posteriori
NIRS	Near Infrared Spectroscopy

PCA	Principal Component Analysis
RDE	Relevant Dimensionality Estimation
ROC	Receiver Operator Characteristic
SF	Supervised, fixed decoder
SOA	Stimulus Onset Asynchrony
SPM	Symbols Per Minute
swLDA	Stepwise Linear Discriminant Analysis
SVM	Support Vector Machine
UA	Unsupervised, randomly initialised decoder
UA-P	Unsupervised, pre-trained decoder
UA^U	Re-evaluation of UA
UB	Unsupervised, batch trained decoder
UB-P	Unsupervised, batch trained decoder, used an extended dataset
UF-P	Unsupervised, fixed, pre-trained decoder
SSVEP	Steady State Visually Evoked Potentials
TA	Adaptive transfer learning decoder
TA^U	Re-evaluation of TA
TF	Fixed transfer learning decoder

Contents

1 Introduction

Brain-Computer Interfaces (BCI) provide a direct communication channel between the brain and a computer (Vidal, 1973). This extra communication channel makes it possible to control the computer by thought alone. To facilitate this, a typical BCI entails three components, as illustrated in Figure 1.1. First, we have a recording device, e.g. an EEG cap, to measure brain signals, second, a computer that is able to understand these brain signals, and third the intended end-user application, e.g. a word processor, or device, such as a robotic arm, to be controlled. One of the most common types of BCI is the Event-Related Potential (ERP) based BCI (Farwell and Donchin, 1988) which aims at providing a new means of communication and control to patients who cannot communicate directly with their environment. A key problem of BCIs is that they have to rely on a calibration recording to train the computer on how to decode the user's brain signals. Moreover, this calibration session is rather tedious and time-consuming. This can be problematic for patients with a limited attention span (Li et al., 2011). For this reason, the BCI community has spent considerable effort on reducing the need for a calibration session and has worked towards developing so-called zero-training BCIs. The goal for these zero-training BCIs is that they can be used reliably without any user-specific calibration. Despite the significant effort, true zero-training BCIs are rather rare. This doctoral thesis details the development of novel machine learning algorithms to facilitate a true zero- training approach for ERP-based BCI.

This introductory chapter is devoted to a slightly more elaborate

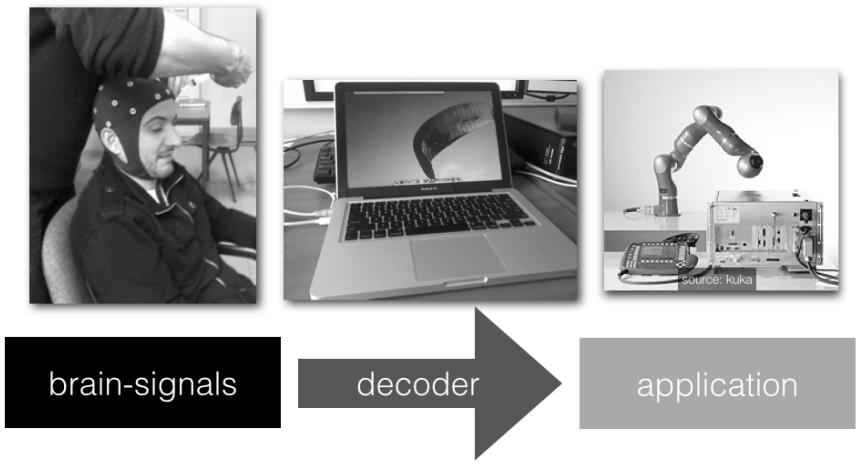


Figure 1.1: Schematic overview of a BCI system. A typical BCI comprises three components. The first component is responsible for recording the brain signals, the second component, the decoder has to understand these brain signals and use them to perform the correct action in the application, which is the third component.

introduction to BCI, where I will illustrate the necessity of machine learning in BCI research. This will be followed by a section that focuses on ERP-based BCI. Afterwards, I will present an overview of the state of the art in machine learning based decoding in BCI, with a focus on ERP-based BCI. To conclude, I will give an overview of my research contributions and highlight the structure of the subsequent chapters.

1.1 Brain-Computer Interfaces

The concept of BCI was introduced in the early seventies by Vidal (1973), and considers the control of a computer by thought alone. In general, BCI research is motivated by the desire to aid patients by restoring motor function (e.g. after spinal cord injury) or communication (e.g. locked-in patients). On top of that, a new target population is emerging: healthy people with the desire to use BCI

for gaming (Müller et al., 2008; Lécuyer et al., 2008; Bonnet et al., 2013). The aforementioned BCIs require the user to be actively involved, a different type of BCI, so-called passive BCIs, can be used to assess workload (Brouwer et al., 2013) and for the evaluation of the perceived image or sound quality (Porbadnigk et al., 2013). At this point, I would like to stress that BCIs are not (yet) capable of reading a mind, although the fMRI based video reconstruction developed by Nishimoto et al. (2011) can create a fuzzy reconstruction of the video a user is watching by decoding the brain activity. Instead, BCIs rely on the detection of very specific brain signals. These specific signals can be the response to stimuli (Farwell and Donchin, 1988) or the voluntary modulation of brain rhythms, e.g. by imagined movement (Pfurtscheller et al., 1997).

1.1.1 Recording the brain signals

The first step in building a BCI is sensing (and almost always recording) the brain activity, for which a variety of devices can be used. A first example is functional magnetic resonance imaging (fMRI), which measures the level of blood oxygenation. This method allows the researcher to pinpoint precisely which specific brain-regions are activated. However, while the spatial resolution is high (on the order of millimetres), the temporal resolution is quite low (on the order of seconds, which is slow considering that our brain is can recognise faces in less than two tenths of a second). Furthermore, fMRI devices are bulky (i.e. they can fill an entire room) and expensive. Near Infrared Spectroscopy (NIRS) is similar to fMRI in the sense that it can locate the active brain regions with high precision, but it is limited to the outer regions of the brain, up to a depth of a couple of centimetres. On the other hand, NIRS is portable and can be taken to the user, instead of the user coming to the lab. Both aforementioned methods are non-invasive, but it is also possible to measure brain activity directly within the brain itself. Obviously, this requires an operative procedure to place an electrode array directly in the cortex, which is not without risk. Nevertheless, it has been shown that the signals recorded in the cortex can be used, in animals and more recently in humans too, to control a prosthetic device such as a robotic arm. Fur-

thermore, it is also possible to place electrodes on top of the cortex but under the skull, which is called ECoG or electrocorticography. Quite often, users of ECoG based BCI are epilepsy patients who had the electrodes implanted to locate the epileptic seizure's focal points precisely before removing them operatively. The most commonly used and one could say most user friendly[1] approach to measuring brain activity is the electroencephalogram (EEG), which will be used throughout this work. Using EEG, electrical activity is measured on the scalp (Berger, 1929). The advantage of EEG is that it is a portable setup which requires only an amplifier the size of a tissue box and an EEG cap. Furthermore, it has a high temporal resolution, of up to a thousand samples per second. The signal, however, is very noisy, artefact prone and offers a spatial resolution of at best 3 cm (Nunez et al., 1997). Typical EEG caps are so-called wet caps, in which conductive gel is injected between the scalp and the electrode. This increases setup time but the measured signal is of better quality than with dry-caps where the conductive gel is not needed. Furthermore, there exists a range of consumer EEG devices, e.g. the Emotiv Epoc, which offer low quality EEG at a low price. Nevertheless, these devices are suitable for demonstrations and portable inexpensive EEG could eventually allow us to bring BCI to the masses. However, to build a truly reliable system and for research purposes or in medical applications, high quality EEG is desired.

1.1.2 Decoding the brain signals

A BCI does not only requires raw brain data, but there also needs to be an informative signal within the recordings. For this informative component, BCIs rely on the voluntary generation of specific brain signals, such as imaginary movement, or on involuntary cortical responses elicited by external stimuli. Examples of the latter are Steady state visually evoked potentials (i.e. an image flickering at a specific frequency) and Event-Related Potentials, which will be discussed later. However, recognising these different responses is quite hard due to the large amount of noise, variability and the huge dimensionality

[1] One could also argue that user friendly is not really applicable yet to devices designed to sense brain signals.

of the signals.

To enable man-machine communication, early BCI prototypes relied on the users to modulate their brain signals such that they could easily be recognised by the computer (Wolpaw et al., 1991; Birbaumer et al., 1999). Here, the users were responsible for generating the correct signals. As a result, the users had to follow an extensive training program just to learn how to modulate their brain signals. This was a very time consuming and exhausting process, which is not desirable. Hence, over the years, based on the concept: *let the machines learn*, which was promoted by the Berlin BCI group, the training process shifted from the user towards the computer (Müller et al., 2004; Dornhege et al., 2007; Blankertz et al., 2007).

Machine learning (Bishop, 2007), which considers the development of algorithms that enable a computer to learn from data, is needed because difficult problems, such as decoding brain signals, cannot be solved by standard algorithms, not even when a team of engineers spends years on developing algorithms because they cannot predict all the variability in the data. To make this more intuitively clear, I would like to present the following analogy.

Imagine that you are lost in an unknown city and you have to find the train station. When you ask for directions, probably on Google Maps, you will be given a very specific set of instructions. Go to the end of this street, turn left, follow this street, ... If you execute the instructions rigorously then you will arrive at your destination. In a sense, you are executing a standard algorithm, there is no ambiguity. This was, however, an easy problem.

A much more difficult problem, that has been conquered by anybody who has gotten this far in the thesis, is reading. Reading involves the recognition of symbols, which can be quite hard, especially if the text is written by hand. Imagine that you have to explain (on the phone) to an illiterate person how to recognise a handwritten *a*. Furthermore, you have to make sure that he does not confuse it with any other symbol such as an *0* or a *d*. Coming up with a set of unambiguous steps that allow you to discriminate between all symbols regardless of how they are written is nigh on impossible. There is just too much variability to describe. Moreover, if you are not able to explain this recognition task to a human, you will certainly fail when

you would have to formalise these steps in a computer algorithm.

Nevertheless, for humans the solution is (reasonably) simple. All you have to do is teach them, by presenting them with examples of the symbols and telling them which is which. Eventually, they will learn how to recognise the symbols. What's more, learning is not limited to humans: computers can learn as well. This is exactly the subject of study in machine learning. Machine learning methods allow us to train the computer on difficult tasks by presenting examples of the data and the corresponding solution. This allows us to recognise specific brain responses but also enables us to solve other (for a computer) challenging tasks such as image recognition (Krizhevsky et al., 2012), or music recommendation (van den Oord et al., 2013).

In the field of BCI, the machine learning algorithms are trained on data from a so-called calibration session, in which the user is requested to perform a specific task at a specific moment. This enables the experimenter to label the EEG data with the user's actions. This labelled dataset is then fed to the machine learning algorithm, which analyses the data and learns how to decode this specific user his brain signals. Only after this training procedure, can the decoder be put to work by decoding the brain signals while the user actually uses the intended application. Obviously, such a calibration session is not a truly productive way for the user to spend his time. Furthermore, calibration sessions can significantly reduce the time available for using the BCI, especially for patients with a limited attention span (Li et al., 2011). Therefore, a major challenge in the BCI community is to reduce or even remove the dependency on the calibration session. This is where this thesis contributes by developing a true zero-training approach for one such type of BCI, the ERP-based BCI.

1.1.3 BCI Applications

I would like to point to the wide variety of BCI applications already in existence. By far the most common application, which is also considered in this thesis, is communication (Wolpaw et al., 2002). BCI spellers can be based on ERP paradigms (e.g. (Farwell and Donchin, 1988; Höhne et al., 2010; Schreuder et al., 2011b; Acqualagna and Blankertz, 2013)), SSVEP (Cheng et al., 2002; Müller-Putz et al.,

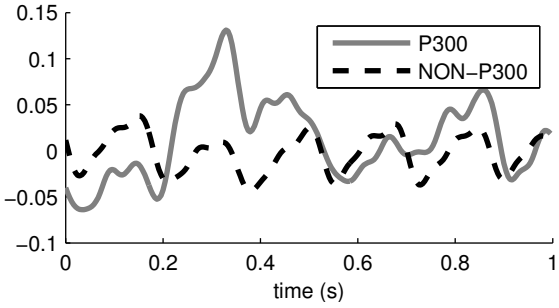


Figure 1.2: A depiction of an averaged ERP target response versus a non-target response from a dataset of visual ERP data.

2005) or motor imagery (Blankertz et al., 2006a). Cursor control is typically based on imagined movement (Wolpaw et al., 1991). More advanced applications of imagined movement include the control of a helicopter (LaFleur et al., 2013) and pinball (Tangermann et al., 2008). Additionally, BCIs have been developed to control robotic arms (Hochberg et al., 2006, 2012; Santhanam et al., 2006; Bishop et al., 2014), to decode finger movements (Kubanek et al., 2009; Wang et al., 2011), to perform speech reconstruction (Pasley et al., 2012) and to aid in rehabilitation (Pfurtscheller et al., 2008).

1.2 Event-Related Potential based BCI

One of the most common types of BCI based on Event-Related Potentials (ERP), and was introduced by Farwell and Donchin (1988). We will begin with a general introduction to the concept of ERP-based BCI, followed by a discussion of two specific ERP paradigms.

An ERP is a specific brain response to an event such as the presentation of a stimulus (e.g. a specific sound), a mistake, a motor event, ... (Luck, 2005). In theory, these brain responses are, depending on the event, a positive or negative deflection of the scalp potential that is time-locked, this means that there is a fixed delay, to the event that evoked them. However, there can be quite a bit of variability between different subjects and even between different measurements of

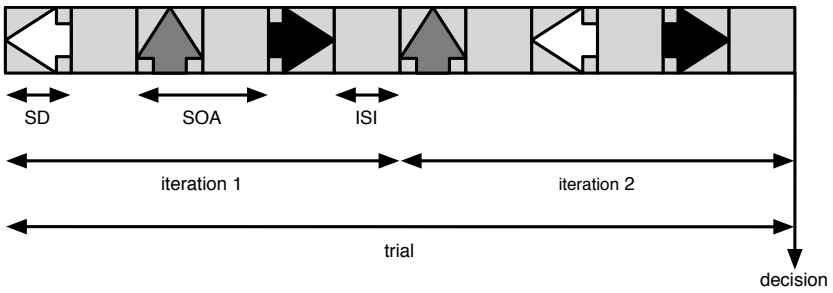


Figure 1.3: Schematic overview of an ERP-based paradigm that uses visual stimuli (arrows).

the same type of ERP response from the same user. Several of these ERP responses are described in literature and we will highlight just two of them. The first example is the N170 response, that is associated with the processing of faces. This response consists of a negative deflection in the EEG, 170 ms post-stimulus. The second example is the P300 ERP, which is a positive deflection, 300 ms post stimulus, that can be generated by so-called oddball paradigms. In these oddball paradigms, the user has to focus his attention on a rare target stimulus that is embedded in a stream of more frequent non-target stimuli (Squires et al., 1977). When the target stimulus is presented, the P300 response is elicited. Furthermore, the P300 response can be modulated by a multitude of factors including its frequency of appearing (Squires et al., 1977), the discriminability of the stimulus (Johnson Jr and Donchin, 1978), but also by motivation when used in a BCI (Kleih et al., 2010).

ERPs can be utilised in a BCI by coupling stimuli to actions. We will use a toy example, shown in Figure 1.3, to explain this concept. In our toy ERP paradigm, we use three different stimuli: an arrow pointing to the left, which is assigned to the action of turning left, an upwards pointing arrow to move forward and to turn right an arrow pointing to the right. These stimuli are presented to the user, who has to focus on one of them. When the attended stimulus is presented, an ERP target response will appear. Hence, when the computer is able to correctly detect the ERP target responses, it can execute the desired action. As a result, the computer can be controlled simply

by focussing on the stimuli. Unfortunately, it is not possible to detect the target ERP response reliably based on a single presentation of each stimulus. For this reason, the stimulus presentations are repeated in an ERP-BCI. Only after each stimulus has been presented multiple times, a decision has to be made. Now, we will formalise this a bit and we will introduce some ERP terminology (also illustrated in Figure 1.3).

The control in an ERP-BCI is discrete and there has to be one decision at the end of a *trial*. Each trial consists of several *iterations*, during which each individual stimulus is presented exactly once. The time between the onset of two subsequent stimuli is the *Stimulus Onset Asynchrony* (SOA). This is equal to the sum of the *Stimulus Duration* (SD) and the *Inter-Stimulus Interval* (ISI), which is defined as the time between the end of a stimulus and the beginning of the next stimulus. In a typical ERP-based BCI the SOA is in the range of 100-200 ms, which is considerably shorter than the time delay between the ERP response and the stimulus. As a result, there will be some overlap. This background knowledge allows us to move on to the P300-BCI, which is the first ERP-based BCI developed.

1.2.1 The P300-BCI

The first ERP-based BCI that was introduced is the P300-BCI, also known as the matrix speller, was introduced in the eighties by Farwell and Donchin (1988). The original matrix speller uses a 6×6 grid of symbols on a computer screen, as shown in Figure 1.4. By focussing his attention on a specific symbol, the user is able to select that symbol. The stimulus sequences are pseudo-random intensifications of rows and columns. Hence, each iteration consists of 12 stimuli, 6 for the rows and 6 for the columns. When the row or column containing the desired symbol is highlighted, a target ERP response is evoked. The intensification of non-target stimuli evokes a non-target response. An example of an average target and non-target response is given in Figure 1.2.

One of the appealing aspects of the P300 based BCI is the fact that it has been tested on patients, which are the true end users (Nijboer et al., 2008). In these experiments, research has shown that

Figure 1.4: An experimental setup for visual ERP spelling. (photo: Francis wyffels)

patients are not only able to control the BCI, they can do so with high accuracy. In fact, the P300-BCI has already been taken out of the lab and found its way into the home of the patient (Vaughan et al., 2006).

Since its inception, researchers have suggested improvements for the visual ERP-based BCI. Here we will focus on paradigm based improvements, whereas machine learning based improvement will be detailed in Section 1.3. First, it is possible to optimise the stimuli themselves (Kaufmann et al., 2011), for example by replacing the intensifications with face based stimuli, such that in addition to the normal ERP response, an N170, related to the recognition of faces, can be observed and effectively enhances the signal to noise ratio. Another option is to optimise the stimulus sequence, where it is even possible to use fixed sequences in place of the common pseudo-random highlighting structure (Tangermann et al., 2012). A different approach to improving the stimuli was presented by Townsend et al. (2010), where the row-column structure of the stimuli was abandoned in favour of groupings of pseudo-random stimulus groups. The goal here is to minimize the most common error in the original matrix speller, where a

neighbouring letter is selected erroneously. Currently, such improvements are also being investigated in our lab (Verhoeven, 2013). Ryan et al. (2010) proposed to incorporate words in the spelling matrix to enable the user to spell an entire word in a single selection.

Yet, neither of those improvements address the major limitation of the matrix speller: the dependency on eye-gaze, over which control can be lost in later stages of ALS. To alleviate this, a novel generation of gaze independent visual ERP spellers have been proposed (Treder et al., 2011; Acqualagna and Blankertz, 2013). There are tactile (Brouwer and Van Erp, 2010; Cincotti, 2010; Kaufmann et al., 2013) and auditory variations (Hill et al., 2004; Schreuder et al., 2010, 2011b; Höhne et al., 2011) and a bi-modal version (Thurlings et al., 2012) as well. One of the auditory variations, the AMUSE paradigm (Schreuder et al., 2010, 2011b), is used extensively in this work and is the topic of our next discussion. For a complete overview of gaze independent ERP-BCI we refer the reader to (Riccio et al., 2012).

1.2.2 Auditory ERP (AMUSE)

The AMUSE paradigm is an auditory ERP-based paradigm. In AMUSE, each stimulus is a specific tone that originates from a specific direction. Our implementation uses 6 unique auditory stimuli, each produced by one of six speakers that are positioned on a ring of 130 cm diameter around the user, as shown in Figure 1.5. This gives the user two different options to discriminate between the stimuli: the tone and the direction. As is common in ERP paradigms, each stimulus is assigned to an action and by attending a specific stimulus, the user can execute the corresponding action. Since AMUSE is limited to 6 different stimuli, it has to rely on a hierarchical multi step process to perform additional actions. In the version of AMUSE that we have used, spelling is done in a two-step process. Each action in the first step allows the user to select a group of 6 stimuli. In the second step, one of these symbols can be selected. Thus, the two-step process allows for the spelling of 36 distinct symbols. In our study, we have omitted the backspace functionality, but it was present in a previous AMUSE study (Schreuder et al., 2011b). Moreover, in that study, it has been shown that AMUSE can be used solely in the auditory

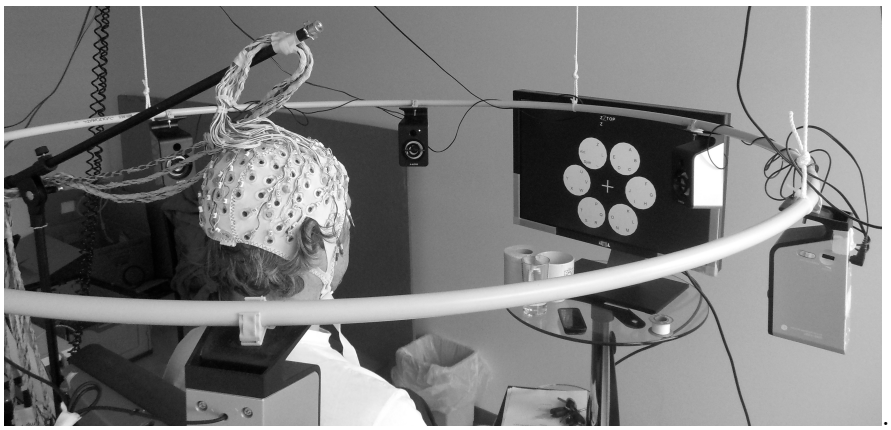


Figure 1.5: The AMUSE spatial auditory ERP setup. The user sits in the centre of a ring, on which 6 speakers are positioned. A specific tone is associated with each of the 6 speakers. By attending one of the six stimuli, the user can perform an action. This picture was taken at the TU-Berlin during the pilot study for the evaluation of the unsupervised algorithm that is proposed in this thesis. The eagle-eyed readers can spot a NIPS mug, which is a much needed accessory in a machine learning lab.

domain, which allowed a blind user to control the BCI.

AMUSE is not the only auditory ERP paradigm in existence and many variations exist (e.g. (Höhne et al., 2011; Hill et al., 2004)) The disadvantage of these Auditory ERP paradigms is that they have a lower signal to noise ratio than visual ERP paradigms, but ongoing research is investigating approaches to improve the raw signal to ratio of auditory ERP paradigms (Höhne et al., 2012; Tangermann and Höhne, 2011).

1.2.3 Measuring performance

In this work, we aim at improving performance of ERP-based BCI and we will develop zero-training methods for ERP-based BCI. Hence, the following question arises naturally: how is performance defined? Do we have to improve the discrimination between target and non-target responses? Do we have to improve the accuracy of the symbol/action selection? Which is more important, the speed of decision making or the accuracy of the BCI? When exactly is one approach better then

another one? The answer to these question is rather subtle.

For this reason, we will utilise different performance metrics, based on the context. The metrics considered in this work are: area under curve (AUC), (selection) accuracy and Symbols Per Minute (SPM). Additionally, we will discuss the Information Transfer Rate (ITR) too, since it is often (mis-)used in BCI research.

Area Under Curve

A first performance measure is the Area Under Curve (AUC). The AUC measures the area under a receiver operator curve. In a receiver operator characteristic, the true positive rate (y-axis) is plotted against the false positive rate (x-axis). The resulting area under the curve is equal to 0.5 when the classifier output is random. It is equal to 1 when the classifier works flawlessly and equal to 0 when the classifier assigns the opposite labels. To compute the AUC, one can use the following procedure.

1. Sort the classifier outputs for all data points
2. use each value of the classifier output as a threshold
   (a) Assign all values above or equal to the threshold to the positive class, assign all values below the threshold to the negative class.
   (b) Compute the true positive and false positive rate and plot it
3. compute the area under the resulting curve.

Accuracy

Evaluating the number of mistakes made by an ERP-BCI is straightforward by computing the accuracy (A):

$$A = \frac{\text{correct selections}}{\text{selections made}}.$$

There is a remark to be made. When a two-step selection procedure is used in a spelling application the selection accuracy will differ from

the spelling accuracy. Throughout this book we will use the selection accuracy unless specified otherwise.

Symbols per minute

The selection accuracy or the AUC do not convey how efficient the BCI actually is. To evaluate the efficiency, we will use the symbols per minute (SPM) approximation from Schreuder et al. (2011a), which is related to the utility metric from Dal Seno et al. (2010). The idea behind SPM is that a user wants to complete a (spelling) task without error. Hence, each error has to be corrected by an undo command. When we have succeeded in predicting the correct action, then we have conveyed one unit of useful information. However, when we made a mistake, then we will have to correct this mistake by the undo command. Therefore, a mistake counts as a negative unit of information. Furthermore, we want to measure efficiency, so we have to take the time per decision, for which the average is denoted by Δ_t. When we bring this all together we get the SPM formula:

$$\text{SPM} = \frac{A - (1 - A)}{\Delta_t} = \frac{2A - 1}{\Delta_t}.$$

Clearly, when the accuracy increases, the SPM increases. Similarly, when the time per decision is shorter, we will be able to spell more symbols. Finally, we would like to link this back to the absolute reliability of the system. Imagine that we have a BCI which is correct 75% of the time. in this case, we will (on average) spend half the time correcting mistakes. More importantly, when the accuracy drops below 50 %, we will not make any progress at all.

Information Transfer Rate

A final performance measure is the Information Transfer Rate or ITR (Wolpaw et al., 2000). The Information Transfer Rate (ITR,bits /minute) is quite often (mis)used (Yuan et al., 2013) as the default metric for communication efficiency in a BCI. It can be computed as follows:

$$\text{ITR} = \frac{\log_2 N + A \log_2 A + (1 - A) \log_2 \frac{1-A}{N-1}}{\Delta_t},$$

where N is the total number of options that can be selected. We will illustrate this measure in our first online evaluation (Section 3.2.2). Its most important disadvantage is that it does not consider mistakes as a type of adverse (negative) information. It does not consider the fact that mistakes might have to be corrected. As a result, the information transfer rate favours very short times between decisions, even when the reliability of the system suffers drastically and makes the system unusable.

1.3 State of the art

In what follows, we give an overview of the current state of the art in machine learning methods for BCI, with a specific focus on ERP-based BCI. For the reader who is not familiar with the basic concepts of machine learning such as cross-validation, regularisation and over-fitting, it might be useful to read Section 2.1 first.

The major challenge in machine learning models is, paraphrasing Mak et al. (2011), to develop methods which perform accurately, but that are not overly complex to be used in a practical setting. To make this clear, the complexity of the underlying method does not actually matter, what matters is whether it can be packaged such that it can be used by the care-takers, which are in most cases rather unfamiliar with machine learning. For this reason, Kaufmann et al. (2012) have investigated the simplification of classifier calibration by developing an easy to use software package. Nevertheless, it should not prevent the machine learning enthusiast from developing novel advanced machine learning methods. But we do have to keep in mind that, in the end, they should be optimised automatically, without the intervention of an engineer.

1.3.1 Feature engineering

One of the major limitations of EEG based BCI is the low signal to noise ratio. For this reason, researchers have spent considerable effort on increasing the signal to noise ratio of the data, for example by performing sensor selection using machine learning based optimisation

(Lal et al., 2004) For motor imagery based BCI, the well known CSP algorithm (Koles et al., 1990), or one of its more advanced successors that focus more on regularisation and generalisation (Lemm et al., 2005; Blankertz et al., 2008; Ang et al., 2012; Samek et al., 2012, 2014), is used in most current motor imagery decoders. However, these methods cannot be applied to ERP-based BCI. There does exist one spatial filtering approach that is designed for ERP paradigms: the XDawn algorithm (Rivet et al., 2009). However, it should be noted that supervised spatial filtering methods cannot be combined with true zero training BCI.

1.3.2 Classifiers

The work by Blankertz et al. (2011) shows that ERP features can be approximated well by a mixture of two multivariate Gaussian distributions. In this mixture, the covariance matrix is shared between the classes but the mean is class-dependent. The optimal classifier in this case corresponds to Linear Discriminant Analysis (LDA). When applied in a machine learning context, proper regularisation must be used to control the model complexity. Moreover, recent work by Farquhar and Hill (2013) has shown that the key to obtaining good performance is to use a properly regularised linear classifier. They have shown that these properly regularised models outperform the most common classifier used in BCI research: stepwise Linear Discriminant Analsysis (swLDA), which is the default option in the BCI2000 software (Schalk et al., 2004).

On the BCI competition III dataset, the best performing method on the ERP dataset is the ensemble of SVMs approach from Rakotomamonjy and Guigue (2008). They divided the training data into 17 parts. Subsequently, they optimised a classifier on each of these 17 subsets. To obtain the final prediction, they combined the outputs of all 17 classifiers. Currently, this method is still one of the best performing methods on the BCI Competition III dataset. While this is an excellent application of machine learning principles (especially related to the cross-validation), the extremely intensive optimisation procedure is rather data hungry and cumbersome. As a result it is not really suited for practical use.

Cecotti and Gräser (2010) have investigated the use of convolutional neural networks (CNN) for ERP decoding. While CNNs are able to achieve state of the art performance on a multitude of tasks (Dieleman et al., 2011; Krizhevsky et al., 2012; van den Oord et al., 2013), they do not perform better than a linear classifier on ERP analysis (Kindermans et al., 2012b). Furthermore, they require large amounts of data to be trained properly, which is rarely available.

1.3.3 Adaptive methods

A major problem in many real-world application is non-stationarity of the data distributions (Sugiyama et al., 2007; Quionero-Candela et al., 2009). In BCI this non-stationarity can be caused by learning effects through co-adaptation (Shenoy et al., 2006; Vidaurre et al., 2011b), reduced attention or fatigue, as well as task-unrelated changes in the mental state. For this reason, the BCI community has investigated methods that can discover the stationary subspace which is stable over time (von Bünau et al., 2009), or over users, such that shared noise subspaces can be removed beforehand (Samek et al., 2013). A different strategy is to adapt the machine learning model on the fly. Adaptive methods have been investigated for many paradigms, including motor imagery (Vidaurre et al., 2011a; Xu et al., 2011) and prosthetic control (Bishop et al., 2014). We will highlight three selected methods that introduce adaptation in the ERP-BCI.

Dähne et al. (2011), inspired by Vidaurre et al. (2011a), have proposed an unsupervised LDA adaptation scheme. In LDA, the classifier is computed as follows

$$\boldsymbol{w} = \Sigma^{-1} \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \right),$$

where Σ is the estimated covariance matrix and $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ are the estimated means of the target and non-target class. To counter a shift in the covariance structure of the data, they have adapted the covariance matrix by approximating it without using label information.

Panicker et al. (2010) have proposed an adaptive mechanism that is composed of two classifiers, where the output of each classifier is used to label the data to train the second classifier. The resulting

method is rather complex and has to be pre-trained using labelled data.

Finally, the work by Lu et al. (2009) uses adaptation in combination with transfer learning, which will be discussed next. They initialised a model on data from other subjects and subsequently adapted it to a novel subject. The major limitation of this method is the high number of parameters and the need for labelled data in the pre-training of the general classifier.

The major difference with our work is that the method presented in this book can learn from scratch, without seeing a single labelled data point (Kindermans et al., 2012b)

1.3.4 Transfer learning in BCI

In transfer learning (Thrun and Pratt, 1998; Pan and Yang, 2010), the goal is to share information across related tasks. In a BCI setting, each user is considered to be a task and we aim to perform inter-subject transfer learning by sharing decoding information between the different users. Most transfer learning methods are based on learning shared representations (Argyriou et al., 2008), building a hierarchical model of the task (Kemp et al., 2007; Salakhutdinov et al., 2011; Gopal et al., 2012) or integrating information from different sources (Palatucci et al., 2009; Frome et al., 2013; Socher et al., 2013).

Transfer learning in BCI is investigated mainly to improve decoding of motor imagination. Fazli et al. (2009, 2011) relied on large-scale optimisation and mixed effects models to leverage decoding knowledge for novel users. Devlaminck et al. (2011) have enhanced the CSP algorithm to directly incorporate multi-subject transfer learning. Krauledat et al. (2008) have investigated the use of prototypical spatial filters that can be shared between users. Finally, Samek et al. (2013) have shown that, while it is possible to find shared subspaces of the signal that only contain noise, finding and sharing informative subspaces between users in motor imagery BCI is quite difficult.

The aforementioned work by Lu et al. (2009) was the first introduction to transfer learning based classification in ERP-BCI. Additonally, Jin et al. (2012) have evaluated the use of an swLDA model trained

on a set of subjects. However, to obtain a highly accurate model, they had to fine-tune it with additional labelled data. Furthermore, the dependency on labelled data remains a major limitation of most recent transfer learning models (Colwell et al., 2013; Herweg et al., 2013)

We contribute with a transfer learning model, that can be trained without using a labelled data point and that can be adapted to the novel user (Kindermans et al., 2012a, 2014)

1.3.5 Incorporating side information

Speier et al. (2012) propose to use a language model in combination with swLDA. To combine classifier, tri-gram language model and dynamic stopping, the output of the linear classifier during training is used to fit a two component 1D Gaussian mixture model. This mixture model is used to transform the projected data into probabilities. These are combined with the language models and finally used in the dynamic stopping strategy. Just to tie the separate components together, 5 additional parameters have to be tuned. This contrasts with our model, which will be introduced in Chapter 3, where we have the same capabilities but without introducing additional parameters. A different change to the input mechanism was proposed by Höhne et al. (2010), who used a T9-based predictive text system in an auditory BCI.

A different take on using side information was used by Wang et al. (2011) to improve the decoding performance of finger flexion based on ECoG. In their work, the authors presented a probabilistic model that was able to capture the dynamics of finger flexion in the evaluation dataset. This allowed them to filter the output to obtain a cleaner version of the predictions. In a sense, this approach is similar to language model based decoding. Both a language model and their probabilistic model define a prior expectation over the decoding output. Hence, we can optimise our prediction such that it fits our expectation.

1.3.6 Improving the efficiency through dynamic stopping

Dynamic stopping in ERP-based BCI considers techniques which stop the stimulus presentation sequence as soon as the classifier is confident enough to predict the attended stimulus. As a result, dynamic stopping methods can increase the communication rate drastically. Many different dynamic stopping methods have been proposed, and for a detailed study, we refer to the excellent work of Schreuder et al. (2013). Here, we will briefly summarise a subset of the existing dynamic stopping approaches.

Schreuder et al. (2011b) used the median between the projection of the data from the most likely and the second most likely symbol to measure the confidence level of the classifier. The stimulus presentation is stopped when the confidence level exceeds a pre-determined subject-specific value, which has to be optimised on labelled data.

A more involved approach was proposed by Lenhardt et al. (2008), where the average LDA output per element in the spelling matrix was computed across iterations. These values were subsequently normalised such that the maximum is mapped to 1 and the minimum is mapped to 0. The sum of these normalised values is used to measure the confidence level. When the sum of these values is low, the classifier is predicting a single symbol with high confidence. Hence, when the sum of all values drops below a subject-specific threshold, the stimulus presentation is halted and the computer makes a prediction.

An SVM-specific method has been proposed by Liu et al. (2010). After supervised training, they compute the average distance between the ERP response and the separating hyperplane. During spelling, the EEG is averaged over the last three iterations. If the distance between the averaged EEG and the separating hyperplane is once or twice the average distance computed on the training data, the symbol is selected and stimulus presentation is stopped.

Jin et al. (2012) propose a classifier-independent dynamic stopping strategy. If the classifier predicts the same symbol twice in a row, they spell that symbol and move on to the next one.

From this overview, it is clear that most of the dynamic stopping strategies are additional post-processing procedures on the output.

This contrasts with what we will propose in Chapter 3, where the dynamic stopping strategy is a natural extension of the classifier itself.

1.4 Research contributions and structure

In this section, I briefly discuss my research contributions and present an overview of the contents of the subsequent chapters.

A unified probabilistic model for Event-Related Potential based BCI

Machine learning approaches for ERP-BCI have become more and more complex. Today, most state of the art decoders incorporate language statistics to improve reliability and a dynamic stopping criterion to speed up the communication process. As a result, many of these approaches are made up of several components with many parameters each. On top of that, additional parameters are introduced to tie all components together. Consequently, properly optimising these models requires complicated cross-validation procedures. This is only possible when a large quantity of labelled data is available and this involves lengthy calibration sessions.

Our unified probabilistic model was developed to simplify the optimisation of the decoder. It does not need extensive cross-validation procedures, but it does retain the reliability of a language model based method and the efficiency of dynamic stopping. We have achieved this by combining the basic decoder and the language model into a coherent probabilistic model. Hence, no additional parameters are needed to combine the decoder and the language model. Moreover, the parameters of the decoder can be optimised without cross-validation. Last but not least, the probabilistic output generated by this unified model gives way to a natural dynamic stopping strategy.

We view this unified probabilistic model as a building block, onto which additional components can be seamlessly integrated. This unified probabilistic model is the backbone of this thesis and will be extended with unsupervised and transfer learning.

A novel unsupervised, constraint-based learning algorithm

The aforementioned unified model relies on supervised training. It requires a calibration session to obtain the labelled data to train the subject-specific model. To remove the dependency on labelled data, we present what is to our knowledge the first unsupervised training algorithm for ERP-based BCI. Please note that this is different from adaptive methods, which are trained with label information and subsequently adapted without label information. The proposed model is able to learn from scratch, it does not require a single labelled data point. Instead it learns how to decode the user's brain signals automatically by analysing the unlabelled data. This novel algorithm exploits the constraint that in each trial only a single stimulus type can result in an ERP target response. The result is an approach that allows us to forego the calibration session completely. Moreover, when enough unlabelled data is available, the proposed algorithm is able to build a decoder that is as reliable as subject-specific supervised methods. Hence, this novel algorithm contributes significantly by bringing us one step closer to true zero-training BCI. Moreover, the unsupervised learning method results in an inherently adaptive model, which can cope with effects of non-stationarity, e.g. due to fatigue, that can cause a supervised model to fail.

However, the drawback to this method, and the reason that it does not directly result in a true zero-training BCI, is that while it does not require labelled data, it still requires a substantial amount of data. During online experiments, the need for a substantial amount of data is manifested as a warm-up period at the beginning of an experiment. During this warm-up period the model is not reliable and makes decoding errors. However, as more and more data is processed, the quality of the model improves and the model is able to revise its mistakes.

The proposed learning method is evaluated extensively on data from auditory and visual ERP paradigms. On top of that, we also present the results from an online study, conducted at the TU-Berlin, using the AMUSE paradigm which demonstrates that our method can effectively be used in real BCI experiments.

Transfer learning in ERP spellers

Clearly, the limitation of the unsupervised learning method is the warm-up period. Hence, in order to build a true zero-training BCI, the warm up period must be eliminated. To achieve this, we present an extension of our unified model that enables us to incorporate decoding knowledge from other users. The extension uses inter-subject transfer learning, which shares information on how to decode the EEG signals between the different users.

This approach can be used to build a subject-unspecific decoder, which is somewhat accurate, but cannot compete with subject-specific model. By combining this basic transfer learning model with the unsupervised learning algorithm, we are able to build a true zero-training BCI, which works reliably from the very first trial. Additionally, this model can be extended with language models and dynamic stopping. Our simulation results on this elaborate model indicate that in a visual ERP speller, a user can, on average, spell 29 symbols in the time required to do the calibration recording. This increase in efficiency is potentially very rewarding for patient applications.

Similar to the unsupervised model, we have evaluated this method extensively. Data from 22 subjects using a visual ERP paradigm was used to assess the performance. On top of that, we have analysed the results from the data generated by the original AMUSE study, which comprises 21 subjects and we have performed an additional online evaluation using AMUSE with 20 additional users.

Understanding unsupervised and transfer learning

Transfer learning has gained much attention from the BCI community recently. However, it contrasts with the tradition of building subject-specific supervised decoders. Furthermore, one would expect that transfer learning should not be possible given the high inter-subject variability observed in typical BCI data. Therefore, we provide an additional understanding of the mechanism underlying transfer learning, where we empirically show that while the subjects appear to be fundamentally different, the information content in their signals is actually structured in a similar manner. On top of that, we demonstrate that the unsupervised learning rule is not only able to achieve state

of the art decoding performance, it is able to identify the relevant underlying neurological activity as well.

Structure of this thesis

In the next chapter, we will give a brief and more mathematically inclined introduction to the most relevant machine learning methods and concepts. Afterwards in Chapter 3 we present our unified probabilistic model. In Chapter 4 we detail the unsupervised learning method, including an extensive offline and online evaluation. This model is enhanced with transfer learning in Chapter 5, where we present the results from an offline and an online study. The mechanisms behind transfer learning and how to recover the informative neurophysiological activity is discussed in Chapter 6. We end in Chapter 7, where we will present our conclusions and discuss how this thesis can influence future machine learning and BCI research.

1.5 List of publications

Journal publications

1. Thompson D., Quitadamo L., Mainardi L., Laghari K., Gao S., **Kindermans P.-J.**, Simaral J., Fazel-Rezai R., Matteucci M., Falk T., Bianchi L., Chestek C., Huggins J. (2014), Performance Measurement for Brain-Computer or Brain-Machine Interfaces: A Tutorial. *Journal of Neural Engineering*, submitted

2. **Kindermans P.-J.**, Schreuder M. Schrauwen B., Müller K.-R., Tangermann M. (2014) True Zero-Training Brain-Computer Interfacing - an Online Study. *PLoS ONE*, submitted

3. **Kindermans P.-J.**, Tangermann M., Müller K.-R., Schrauwen B. (2014) Integrating dynamic stopping, transfer learning and language models in an adaptive zero-training ERP speller. *Journal of Neural Engineering*, in press

4. **Kindermans P.-J.**, Verschore H., Schrauwen B. (2013) A unified probabilistic approach to improve spelling in an event-related

potential based brain-computer interface. *IEEE Transactions on biomedical engineering*, 60(10).

5. **Kindermans P.-J.**, Verstraeten D., Schrauwen B. (2012). A Bayesian model for exploiting application constraints to enable unsupervised training of a P300 based BCI. *PLoS ONE*, 7(4).

Conference publications

1. **Kindermans P.-J.**, Verschore H., Verstraeten D., Schrauwen B. (2012). A P300 BCI for the masses: prior information enables instant unsupervised spelling. *Advances in Neural Information Processing Systems 25.*

2. Wybo W., Colle C., **Kindermans P.-J.**, Schrauwen B. (2012). Distance dependent extensions of the Chinese restaurant process *In proceedings of the 21st Belgian-Dutch Conference on Machine Learning.*

3. Verschore H., **Kindermans P.-J.**, Verstraeten D., Schrauwen, B. (2012). Dynamic stopping improves the speed and accuracy of a P300 speller. *In proceedings of the International Conference on Artificial Neural Networks.*

4. wyffels F., Bruneel K., **Kindermans P.-J.**, DHaene M., Woestyn P., Bertels P., Schrauwen B. (2011). Robot competitions trick students into learning. *In proceedings of the 2nd international conference on robotics in education.*

5. **Kindermans P.-J.**, Verstraeten D., Buteneers P., Schrauwen B. (2011). How do you like your P300 speller: adaptive, accurate and simple? *In proceedings 5th international brain-computer interface conference.*

6. **Kindermans P.-J.**, Buteneers P., Verstraeten D., Schrauwen B. (2010). An uncued brain-computer interface using reservoir computing. *NIPS 2010 Workshop: machine learning for assistive technologies.*

Abstracts, posters, demonstrations, presentations

1. Verhoeven T., Buteneers P., Schrauwen B., **Kindermans P.-J.** (2014) An unsupervised plug and play BCI with consumer grade hardware. *Presented at the BBCI Winterschool on Neurotechnology 2014*, abstract.

2. Buteneers P., Verhoeven T., **Kindermans P.-J.**, Schrauwen B. (2013) A case study demonstrating the pitfalls during evaluation of a predictive seizure detector, *Presented at International Workshop on Seizure Predictions 6*, abstract.

3. **Kindermans P.-J.**, Schrauwen B. (2013) Dynamic stopping in a calibration-less P300 speller. *5th International Brain-Computer Interface Meeting*, abstract.

4. **Kindermans P.-J.** (2013) Unsupervised P300 speller and transfer learning in BCI, *g.tec BCI Workshop Gent*, invited talk.

5. Tangermann M., **Kindermans P.-J.**, Schreuder M., Schrauwen B., Müller K.-R. (2013). Zero training for BCI – reality for BCI systems based on event-related potentials. *Dreiländertagung der Deutschen, Schweizerischen und Österreichischen Gesellschaft für Biomedizinische Technik.*

6. **Kindermans P.-J.**, Verschore H., Verstraeten D., Schrauwen B. (2012) A Fast Accurate Training-less P300 Speller: Unsupervised Learning Uncovers new Possibilities. *Demonstration at Neural Information Processing Systems 2012*, demonstration.

7. **Kindermans P.-J.** (2012) Brain-Computer Interfaces lezen je gedachten. *Gespreksavond Breinwijzer vzw.*, invited talk.

8. **Kindermans P.-J.**, Verstraeten D., Schrauwen B. (2012). A bayesian model for exploiting application constraints to enable unsupervised training of a P300 based BCI. *Poster session at Machine Learning Summer School 2012 in Kyoto*, poster

9. Degrave J., wyffels F., Waegeman T., **Kindermans P.-J.**, Schrauwen B. (2012). Applying morphological changes during the evolution

of quadruped robots results in robust gaits. *In Proceedings of the 21st Belgian-Dutch Conference on Machine Learning*, abstract

10. **Kindermans P.-J.**, wyffels F., Caluwaerts K., Guns B., Schrauwen B. (2012). Towards incorporation of hierarchical Bayesian models into evolution strategies for quadruped gait generation. *In Proceedings of the 21st Belgian-Dutch Conference on Machine Learning*, abstract

2 Machine learning

Machine learning considers data analysis techniques that enable us to tackle difficult problems for which programming a (good) solution explicitly is not feasible. Machine learning methods are typically divided into the following groups: supervised methods, unsupervised methods and reinforcement learning approaches. Supervised methods rely on a labelled dataset to train the model. Typically, these methods involve the classification of data points or the prediction of values based on measurements (regression). Unsupervised learning methods are used to discover structure in the data (e.g. (Blei et al., 2003)), the grouping of data into clusters (e.g. (Hartigan and Wong, 1979)), the reduction of the dimensionality (e.g. (Jolliffe, 2005; Schölkopf et al., 1998; Montavon et al., 2013)) or for learning features that will be used by a subsequent classifier (e.g. (Coates and Ng, 2012)). Finally, reinforcement learning approaches try to solve a task by trial and error. Furthermore, they depend on a reward signal that indicates whether the attempt was successful or not. These reinforcement learning algorithms are typically used for the control of robots (e.g. the control of a helicopter (Abbeel et al., 2007)).

The methods developed in this thesis are supervised and unsupervised algorithms. To give the reader a background in machine learning that is sufficient to understand the contributions detailed in this book, I will give a brief introduction to the most closely related machine learning methods. In the next section I will discuss the two most important concepts in machine learning: over-fitting and regularisation. Afterwards, I will introduce one of the most simple, but

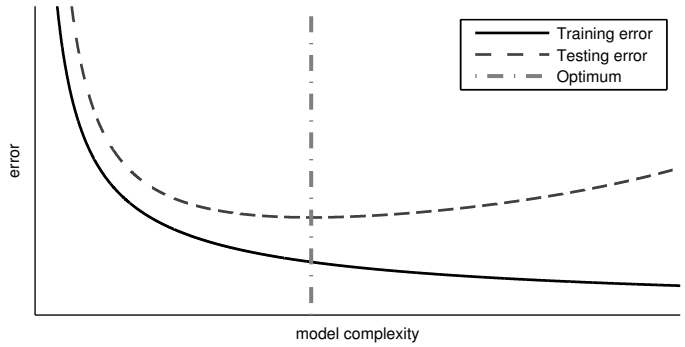


Figure 2.1: The model complexity increases from left to right. When the model is too simple to represent the data, both the train and testing error are high and the model is under-fitting. When the model complexity increases, we can find a better representation of the problem. As a result, both the train and testing error are improved. However, when the model becomes too complex, we start to learn the train data by heart and the model stops generalising. Consequently, the testing error starts to increase while the training error continues to diminish and the model is over-fitting.

effective, machine learning methods: least squares regression. This will be followed by a probabilistic perspective on this powerful regression method. Subsequently, I will detail the Linear Discriminant Analysis (LDA), which is very often used as a classifier in BCI research. Next, we will turn our attention to a widespread unsupervised learning model: the Gaussian Mixture Models (GMM) and more importantly to the Expectation Maximisation (EM) algorithm, that is used to train the GMM model. Finally, I will also show how EM can be used to set the hyper-parameters in Bayesian Linear Regression. I will follow the notation conventions used in the machine learning community (Bishop, 2007).

2.1 Over-fitting, regularisation and cross-validation

The concepts of over-fitting and regularisation are related to the generalisation capabilities of the machine learning model. We say that a machine learning model is able to generalise when it performs well on unseen data and not just on the data it was trained on. Obtaining a perfect result on the training data is easy, we just have to memorise all the individual data-points and the corresponding solution. However, when we learn the data by heart, our model won't perform well on unseen data. This phenomenon, where training error is very low, but the error on an unseen dataset is high is called over-fitting. Under-fitting, on the other hand, occurs when the model is too simple to even model the training data. This is illustrated in Figure 2.1. When the model complexity is low, both the training and the testing errors are high, the model is under-fitting. As the model complexity increases the model starts to perform well on both the train and test data. However, once the model becomes too complex, the testing error increases, while the training error continues to decrease and the model is over-fitting.

To understand how to correctly fine-tune the model complexity, we have to take a step back and look at how machine learning models are trained. For a machine learning model, the optimisation criterion, the so-called objective function, is based on the performance on the train set. Commonly used objective functions are the classification error rate[1] or the mean squared error between the predictions and the desired output in the case of regression. Hence, when we optimise this objective function, an over-fitting model will perform best. However, the performance on the train set is largely irrelevant, and we want models that generalise well to unseen data. Therefore, we need a method to make the trade-off between complex models, i.e. models with many parameters, which are potentially more powerful but also more susceptible to over-fitting and simpler less complex models.

[1] Quite often not the classification error rate, but a closely related metric that ensures tractability.

To enforce this trade-off between model complexity and performance on the train set during the optimisation process, we rely on regularisation. Regularisation methods add an extra term, with a free parameter, to the objective function that penalises model complexity or, when used in a probabilistic setting, by placing prior distributions on the parameters. Using a prior distribution is equivalent to pretending that you have already observed some data. In the following sections we will discuss the regularisation methods in conjunction with the machine learning techniques they are applied to. The regularisation parameter has to be tuned as well. Using only the performance on the train set, it is not possible to find a correctly optimised value. More importantly, using the evaluation data is not possible either, since we have no access to this data. The solution is rather simple when enough data is available. In this case, we can be split the data into a train set and a validation set. The train set is used to train the model, while the validation set is used to assess the performance on unseen data. This gives us a good estimate of the generalisation properties and enables us to find the optimal parameter settings. However, when datasets are small, using a dedicated validation set is not feasible. The solution to this problem is cross-validation. In K-fold cross-validation, the dataset is divided into K parts. During each of the K folds, one part is used for validation, while the remaining parts are used for training. By combining the validation results, the optimal parameter setting can be found. The concept of cross-validation is illustrated in Figure 2.2.

While this is a chapter devoted to machine learning concepts, we would like to briefly go back to optimisation in BCI. Recall that recording calibration data is rather cumbersome, and we would like to minimise this as much as possible. This clashes with the need for cross-validation which requires additional data to properly train the model. As a result, we would prefer to use methods that can be trained reliably without cross-validation procedures.

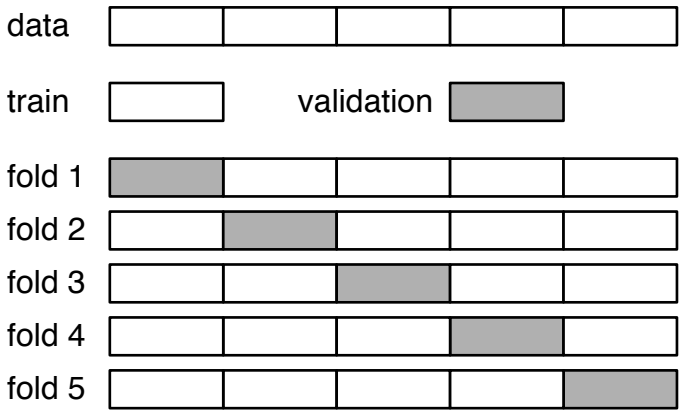


Figure 2.2: Illustration of 5-fold cross-validation. The dataset is divided into 5 chunks. In each fold, a single block of data is used for validation, the other blocks are used to train the model. This gives an approximation of the generalisation properties of the parameter setting.

2.2 Supervised Learning

We start with the most common machine learning models, those methods that are trained using labelled data, i.e. on examples of the data and their corresponding solution.

2.2.1 Linear regression

The task in regression is to use the measured data $\boldsymbol{x} = [x_1, \ldots, x_D]^T$ to predict a quantity y. In linear regression, it is assumed that the target y can be expressed as a linear combination of the input features $\boldsymbol{x}$

$$y = w_1 x_1 + \ldots + w_D x_D.$$

This can be written in vector notation as follows

$$y = \boldsymbol{w}^T \boldsymbol{x}.$$

A constant bias term, that is equal to 1, can be included in the feature vector to introduce an offset to the origin. Furthermore, this method

can be made non-linear by extending the feature vector using a fixed set of non-linear functions of the input features, e.g. taking the square.

$$y = \boldsymbol{w}^T \phi\left(\boldsymbol{x}\right).$$

Least squares optimisation

To train the model, we require an objective function, which, for regression is the mean squared error:

$$E_{\text{mse}} = \sum_{n=1}^{N} \frac{\left(y_n - \boldsymbol{w}^T \boldsymbol{x}_n\right)^2}{N}.$$

Here the subscript n is used to indicate the specific sample from the training data. We can write this error function in vector form as well, where $\boldsymbol{y} = [y_1, \ldots, y_N]^T$ and $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$

$$E_{\text{mse}} = \frac{\text{Tr}\left((\boldsymbol{w}^T X - \boldsymbol{y}^T)(\boldsymbol{w}^T X - \boldsymbol{y}^T)^T\right)}{N}.$$

To train the model, we have to minimise the objective function by taking the gradient of the objective function with respect to $\boldsymbol{w}$

$$\begin{aligned}\nabla_{\boldsymbol{w}} E_{\text{mse}} &= \nabla_{\boldsymbol{w}} \frac{\text{Tr}\left((\boldsymbol{w}^T X - \boldsymbol{y}^T)(\boldsymbol{w}^T X - \boldsymbol{y}^T)^T\right)}{N}, \\ &= \frac{2}{N}\left(\boldsymbol{w}^T X - \boldsymbol{y}^T\right) X^T.\end{aligned}$$

Then, the gradient has to be made equal to zero to find where $\boldsymbol{w}$ maximises E_{mse}

$$\boldsymbol{0} = \frac{2}{N}\left(\boldsymbol{w}^T X - \boldsymbol{y}^T\right) X^T,$$

where $\boldsymbol{0}$ denotes a vector of zeros. Solving the equation above with respect to $\boldsymbol{w}$ yields the following optimal solution

$$\boldsymbol{w}^T = \left(X X^T\right)^{-1} X \boldsymbol{y}.$$

One of the major advantages of regression is that we can compute this optimum in a single step. The difficulty, however, is the computation of the inverse of the correlation matrix, which can become problematic

when the variables are highly correlated and it might be necessary to use a pseudo inverse.

It has been shown that the model complexity in least squares regression is related to the size of the weights of the projection. Therefore, we can introduce a regularisation parameter λ to penalise complex models by using L_2 regularisation on the objective function

$$E_{\text{mse}+L_2} = \sum_{n=1}^{N} \frac{\left(y_n - \boldsymbol{w}^T \boldsymbol{x}_n\right)^2}{N} + \lambda ||\boldsymbol{w}||_2.$$

The optimisation is analogous to the previous optimisation procedure and results in the following solution for $\boldsymbol{w}$

$$\boldsymbol{w} = \left(XX^T + \lambda I\right)^{-1} X\boldsymbol{y}.$$

This regularised version of least squares regression is often called ridge regression (Hoerl and Kennard, 1970). To optimise λ we have to use cross-validation.

While regression (prediction) is fundamentally different from classification, linear regression is often used for classification by using the targets $-1, 1$ for the non-target and target class respectively. When $\hat{y} = \boldsymbol{w}^T \boldsymbol{x} > 0$, we predict the target class, otherwise, we predict the non-target class. However, when the ratio of target to non-target data points is not balanced, the projections may shift towards the most frequently occurring class. This can be solved by using class-re-weighted regression (Toh, 2008). This is, however, not a problem in ERP based BCI, where we will use it, because we are interested in the relative predictions of the data points compared to each other.

2.2.2 A probabilistic interpretation of linear regression

Linear or least-squares regression can also be approached from a probabilistic perspective. In this case, we assume that the output is a linear combination of the input variables plus some additive Gaussian noise

$$y = \boldsymbol{w}^T \boldsymbol{x} + \epsilon.$$

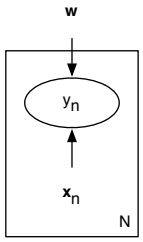


Figure 2.3: Graphical model for the probabilistic interpretation of linear regression

The noise is denoted by ϵ, which is normally distributed with zero mean and variance β^{-1}. This is equivalent to the assumption that the error in linear regression is normally distributed

$$\epsilon \sim \mathcal{N}\left(0, \beta^{-1}\right).$$

Therefore, our observations are normally distributed with variance β^{-1} and mean $\boldsymbol{w}^T\boldsymbol{x}$

$$y \sim \mathcal{N}\left(y|\boldsymbol{w}^T\boldsymbol{x}, \beta^{-1}\right).$$

The graphical model representation is shown in Figure 2.3, where we have used standard graphical model notation. The variables in the ovals[2] are random variables, for which we have defined a distribution. The variables without ovals are parameters/observations for which no distribution is defined. A rectangle indicates repetition.

To train the model, i.e. to find a suitable setting for $\boldsymbol{w}$, in a probabilistic manner, we maximise the likelihood of the observations given our model. This type of optimisation in called Maximum Likelihood Estimation. The expression for the likelihood on a dataset becomes

$$E_{\text{mle}} = \prod_{n=1}^{N} p\left(y_n|\boldsymbol{x}_n, \boldsymbol{w}\right) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\beta^{-1}}} \exp\left(-\frac{\left(y_n - \boldsymbol{w}^T\boldsymbol{x}_n\right)^2}{2\beta^{-1}}\right).$$

This is simply the joint probability of all predictions given our model

[2]Normally, circles are used, but we will need ovals later to keep the figures small.

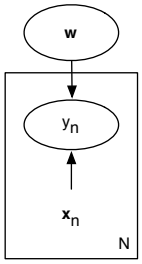


Figure 2.4: Graphical model for Bayesian linear regression

$\boldsymbol{w}$ under the assumption that all the $\boldsymbol{x}_n$ are independent observations.

Quite often, the product term in the computation of the likelihood makes it hard to maximise the likelihood directly. For this reason, to train the model, we take the log, which does not change the optimum but simplifies the process by replacing the product with a sum. When we compute the gradient of the log likelihood with respect to $\boldsymbol{w}$, set it equal to 0 and simplify we get the following

$$\begin{aligned}
\nabla_{\boldsymbol{w}} \log E_{\text{mle}} &= \nabla_{\boldsymbol{w}} \log \prod_{n=1}^{N} p\left(y_n | \boldsymbol{x}_n, \boldsymbol{w}\right), \\
\mathbf{0} &= \sum_{n=1}^{N} \nabla_{\boldsymbol{w}} \log p\left(y_n | \boldsymbol{x}_n, \boldsymbol{w}\right), \\
\mathbf{0} &= \sum_{n=1}^{N} \nabla_{\boldsymbol{w}} \log \frac{1}{\sqrt{2\pi\beta^{-1}}} \exp\left(-\frac{\left(y_n - \boldsymbol{w}^T \boldsymbol{x}_n\right)^2}{2\beta^{-1}}\right), \\
\mathbf{0} &= \sum_{n=1}^{N} -\nabla_{\boldsymbol{w}} \frac{\left(y_n - \boldsymbol{w}^T \boldsymbol{x}_n\right)^2}{2\beta^{-1}}.
\end{aligned}$$

From which it is clear that maximising the log likelihood is equal to minimising the mean squared error. Hence, the resulting optimum for $\boldsymbol{w}$ is identical

$$\boldsymbol{w} = \left(XX^T\right)^{-1} X\boldsymbol{y}.$$

Obviously, the solution is not regularised, and as a result there is a high risk of over-fitting. To introduce regularisation, we have to resort to Bayesian Linear Regression.

In Bayesian Linear Regression, we make $\boldsymbol{w}$ a random variable an

place a prior with zero mean and isotropic covariance matrix $\alpha^{-1}I$ on this weight vector. This allows us to write the complete model as follows

$$\begin{aligned} \boldsymbol{w} &\sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}I\right), \\ y_n|\boldsymbol{w} &\sim \mathcal{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n, \beta^{-1}\right). \end{aligned}$$

A graphical representation of this model is given in Figure 2.4. instead of optimising the likelihood of the model, we will compute the posterior distribution on the model parameters given the training data. This is a maximum a posteriori estimate of $\boldsymbol{w}$. Here, we maximise the likelihood of the model given our training data. For this, we have to compute the posterior distribution on the weight vector given the data. This posterior distribution is proportional to the prior multiplied by the conditional likelihood of our observations

$$E_{\text{map}} = p\left(\boldsymbol{w}|X, \boldsymbol{y}, \alpha^{-1}, \beta^{-1}\right) = \frac{p\left(\boldsymbol{y}|X, \boldsymbol{w}, \beta^{-1}\right) p\left(\boldsymbol{w}|\alpha^{-1}\right)}{p\left(\boldsymbol{y}|X, \beta^{-1}\right)}.$$

The basic concept behind Bayesian machine learning models is to start with a prior distribution, which is updated based on the observed data.

To optimise the Bayesian Linear Regression model, we maximise the posterior distribution on our model parameters. Hence, we compute the gradient of this expression and we set it equal to zero to obtain the following equation, where we have dropped the terms that do not depend on $\boldsymbol{w}$

$$\mathbf{0} = -\nabla_{\boldsymbol{w}} \sum_{n=1}^{N} \left(y_n - \boldsymbol{w}^T\boldsymbol{x}_n\right)^2 - \nabla_{\boldsymbol{w}} \frac{\alpha}{\beta} \boldsymbol{w}^T\boldsymbol{w}.$$

Here, we see that maximising this posterior distribution is equivalent to minimising the objective function of regularised linear regression. Consequently, we obtain the following solution

$$\boldsymbol{w} = \left(XX^T + \frac{\alpha}{\beta}I\right)^{-1} X\boldsymbol{y}.$$

To select values for the hyper-parameters α, β, (which can be combined in a single one $\lambda = \frac{\alpha}{\beta}$) we can resort to cross-validation, but we

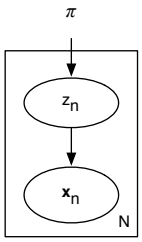


Figure 2.5: Graphical model for LDA.

will show later that those can also be fine-tuned by using the Expectation Maximisation (EM) algorithm.

2.2.3 Linear Discriminant Analysis

Linear Discriminant Analysis is a generative probabilistic model that is used for classification. Generative means that it models how the data was generated. This contrasts with the previous model, where we did not describe how the observed data X is generated. In LDA, we assume that we have $K = 2$ classes of data and each of the N data points $\boldsymbol{x}_n$ belongs to one of these classes. To indicate the class, we use the vector $\boldsymbol{z}_n$ and a one of K encoding, i.e. there is one element of $\boldsymbol{z}_n$ equal to one, the other $K - 1$ are zero. The data follows a multivariate Gaussian distribution, with class-conditional mean $\boldsymbol{\mu}_k$ and a covariance Σ which is shared between both classes. This model is depicted in Figure 2.5 and can be written formally as follows

$$\begin{aligned} k &\in \{1, 2\}, \\ n &= 1, \ldots, N, \\ \sum_{k=1}^{K} \pi_k &= 1, \\ p\left(z_{nk} = 1\right) &= \pi_k, \\ p\left(\boldsymbol{x}_n | z_{nk} = 1\right) &= \mathcal{N}\left(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \Sigma\right). \end{aligned}$$

Three examples of data generated by this model are shown in Figure 2.6.

When we know the parameters, we can predict the target label for

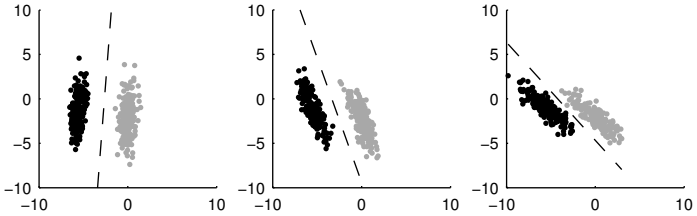


Figure 2.6: Three examples of data generated by the LDA model. Please note that in LDA both classes share the same covariance structure. To shown the influence of the covariance structure on the direction of the decision boundary, we have used the same means per class in all three examples. By changing the covariance structure over the three examples, we rotate the decision boundary.

an observation $\boldsymbol{x}_n$ by applying Bayes's rule

$$p\left(z_{n1}=1|\boldsymbol{x}_n\right)=\frac{p\left(\boldsymbol{x}|z_{n1}=1\right)p\left(z_{n1}=1\right)}{\sum_{k=1}^{2}p\left(\boldsymbol{x}|z_{nk}=1\right)p\left(z_{nk}=1\right)}.$$

We will assign the data point to the most likely class. Furthermore, it is interesting to re-write this using the log-odds ratio

$$p\left(z_{n1}=1|\boldsymbol{x}_n\right)=\frac{1}{1+\frac{p(\boldsymbol{x}|z_{n2}=1)p(z_{n2}=1)}{p(\boldsymbol{x}|z_{n1}=1)p(z_{n1}=1)}}.$$

Hence, predict that $\boldsymbol{x}_n$ belongs to class 1 when $p\left(z_{n1}=1|\boldsymbol{x}\right)>0.5$, which occurs if

$$\frac{p\left(\boldsymbol{x}|z_{n2}=1\right)p\left(z_{n2}=1\right)}{p\left(\boldsymbol{x}|z_{n1}=1\right)p\left(z_{n1}=1\right)}<1\Leftrightarrow\log\frac{p\left(\boldsymbol{x}|z_{n2}=1\right)p\left(z_{n2}=1\right)}{p\left(\boldsymbol{x}|z_{n1}=1\right)p\left(z_{n1}=1\right)}<0.$$

This equation can be simplified further and shows that LDA is actually a linear model

$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{x}+w_0&>0,\\ \boldsymbol{w}&=\Sigma^{-1}\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right),\\ w_0&=-\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1+\frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2+\log\frac{p\left(z_{n1}=1\right)}{p\left(z_{n2}=1\right)}.\end{aligned}$$

This means that LDA has the same discriminative power as the regression based classifier above. Furthermore, note that the covariance structure of the data directly influences the decision boundary, this effect is also illustrated in Figure 2.6.

In general, the true means $\boldsymbol{\mu}_k$ and the true covariance matrix Σ are unknown. Therefore, they have to be estimated during training. To train the model, we can use maximum likelihood estimation which uses the following objective function

$$E_{\text{mle}} = \prod_{n=1}^{N} \prod_{k=1}^{2} \left(\pi_k \mathcal{N}\left(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \Sigma\right)\right)_{nk}^{z} .$$

Which, in turn, results in the following maximum likelihood solution

$$\begin{aligned}
\pi_k &= \frac{N_k}{N}, \\
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} \boldsymbol{x}_n, \\
\Sigma &= \sum_{k=1}^{2} \frac{N_k}{N} \left(\frac{1}{N_k} \sum_{n, z_{nk}=1} \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right) \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right)^T \right).
\end{aligned}$$

The prior probability for each class is proportional to the number of data points in each class. The class-conditional means are equal to the mean of all data-points belonging to that class and the shared covariance matrix is a weighted version of the class-conditional covariance matrix.

In high dimensions, estimation of the covariance matrix can become difficult by lack of data. Therefore, we regularise this covariance matrix by shrinking it towards the identity matrix. In lieu of using the estimated covariance matrix Σ, we will use the shrunken version $\hat{\Sigma}$

$$\hat{\Sigma} = \lambda \Sigma + (1 - \lambda) I.$$

To set λ, we can resort to cross-validation, but it has been shown that it can be optimised analytically (Ledoit and Wolf, 2004). This direct analytical solution will be used in our online experiments. Naturally, LDA can be extended in several ways. We can introduce a prior on the covariance structure or the means of the data. Additionally, we

can extend it to consider multiple classes and to use class-conditional covariance matrices. While this is not often used in a supervised setting, such a model is rather popular in unsupervised learning and known as the Gaussian Mixture Model.

2.3 Unsupervised Learning

A second class of methods are the so-called unsupervised learning methods. In general, unsupervised methods are used to uncover the hidden structure in the data, e.g. the Gaussian Mixture Model (GMM) and Latent Dirichlet Allocation (Blei et al., 2003), for image denoising, or for feature learning (Coates and Ng, 2012). We will focus on the Gaussian Mixture Model and the Expectation Maximisation (Dempster et al., 1977) algorithm to train them, given the close relation to the methods developed in this thesis. We will start with the Gaussian Mixture Model, followed by a discussion on EM and on how to use EM to train a GMM and how to use it to set the hyper-parameters of Bayesian Linear Regression.

2.3.1 Gaussian Mixture Models

A two-component Gaussian Mixture model is almost identical to the LDA model we discussed before, the only difference is that a GMM assumes that each group of data-points has its own covariance matrix and that GMM models are trained without label information. In a GMM model, there are N data points $\boldsymbol{x}_n$, where the generative model specifies that each data point belongs to one of the K groups (or clusters). The data in group k is distributed as a multivariate Gaussian with mean $\boldsymbol{\mu}_k$ and covariance Σ_k. This gives us the following

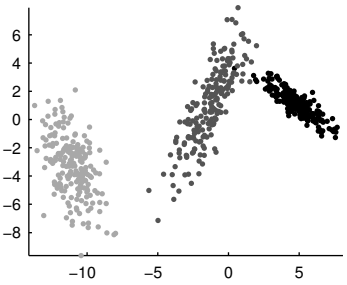


Figure 2.7: Example of data generated from a three component Gaussian Mixture Model. Note that unlike data generated by an LDA model, each cluster has its own covariance structure.

model

$$\begin{aligned} k &= 1, \ldots, K, \\ n &= 1, \ldots, N, \\ \sum_{k=1}^{K} \pi_k &= 1, \\ p\left(z_{nk} = 1\right) &= \pi_k, \\ p\left(\boldsymbol{x}_n | z_{nk} = 1\right) &= \mathcal{N}\left(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \Sigma_k\right). \end{aligned}$$

In this model $\boldsymbol{z}_n$ uses a one of K encoding to specify the cluster to which the nth data point belongs. An example of data generated by this model is shown in Figure 2.7, this figure illustrates the difference with an LDA model where the covariance matrix is shared between the clusters.

To simplify the notation, we have combined our current estimate of the parameters of the model in

$$\Theta = \{\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K\}.$$

To train this model, we use the following update equations until the

parameters converge

$$\begin{aligned}
\hat{\pi}_k &= \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)}{N}, \\
\hat{\boldsymbol{\mu}}_k &= \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right) \boldsymbol{x}_n}{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)}, \\
\hat{\Sigma}_k &= \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right) \left(\boldsymbol{x}_n - \mu_k\right) \left(\boldsymbol{x}_n - \mu_k\right)^T}{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)}.
\end{aligned}$$

The update equations are rather intuitive. We compute for each data-point how likely it is to belong to each individual cluster. Subsequently, we update the parameters of the clusters, where we weight the contributions of the data-points by their likelihood for belonging to this cluster. Apart from the per-cluster weighting, these update equations are pretty similar to the training of an LDA model. Whereas in LDA the parameters were selected using maximum likelihood, this is not possible for the GMM, therefore we rely on the Expectation Maximisation algorithm.

2.3.2 Expectation Maximisation

The Expectation Maximisation (EM) framework (Dempster et al., 1977) can be used to optimise latent variable models, such as the GMM, where it is difficult to maximise the likelihood

$$p\left(X|\hat{\Theta}\right)$$

of the data X with respect to the parameters $\hat{\Theta}$ directly, but where optimisation of the joint distribution of the data X and the latent variables Z

$$p\left(X, Z|\hat{\Theta}\right)$$

is easy. In theory, the EM algorithm maximises a lower bound on $p\left(X|\hat{\Theta}\right)$. Intuitively, the EM algorithm works as follows. We start with an initial hypothesis (i.e. a guess) to explain our observations (i.e. our dataset). Then we verify whether this is a good hypothesis by testing it on the data. This experiment will give us new information on how to improve our hypothesis. After we have updated our hypothesis,

we restart the cycle, by testing and updating until we cannot find a better hypothesis.

The EM algorithm

1. **Initialisation:** Start with an initial parameter setting Θ, which can be random.

2. **Expectation step:** for each possible assignment of the latent variables Z the conditional probability of Z given the current parameter estimate Θ is computed.

$$p\left(Z|X,\Theta\right).$$

3. **Maximisation step:** maximise the expected data log likelihood with respect to $\hat{\Theta}$

$$\hat{\Theta} = \arg\max_{\hat{\Theta}} \sum_{Z} p\left(Z|X,\Theta\right) \log p\left(X,Z|\hat{\Theta}\right).$$

4. **Check for convergence:** either in terms of the log likelihood of the model, or in terms of the parameters. If the model has not converged, set $\Theta = \hat{\Theta}$ and repeat from the expectation step.

What does EM optimise?

It is, however, non-trivial to see that this approach actually maximises a lower bound on the data log likelihood. To make this clear, we start with the data log likelihood and transform that expression. Note that $\hat{\Theta}$ is what we try to optimise and that Θ is our current parameter estimate. We can write the data log likelihood that we wish to optimise as follows.

$$\begin{aligned} &\log p\left(X|\hat{\Theta}\right), \\ =&\sum_{Z} p\left(Z|X,\Theta\right) \log p\left(X|\hat{\Theta}\right) \end{aligned}$$

Here, we have made use of the fact that $\log p\left(X|\hat{\Theta}\right)$ does not depend on Z and that $p\left(Z|X,\Theta\right)$ sums to one. We can expand this further

as follows.

$$
\begin{aligned}
&\sum_Z p\left(Z|X,\Theta\right)\log p\left(X|\hat{\Theta}\right)\\
&=\sum_Z p\left(Z|X,\Theta\right)\left[\log p\left(X,Z|\hat{\Theta}\right)-\log p\left(Z|X,\hat{\Theta}\right)\right],\\
&=\sum_Z p\left(Z|X,\Theta\right)\\
&\quad\left[\log p\left(X,Z|\hat{\Theta}\right)-\log p\left(Z|X,\Theta\right)+\log p\left(Z|X,\Theta\right)-\log p\left(Z|X,\hat{\Theta}\right)\right],\\
&=\sum_Z p\left(Z|X,\Theta\right)\left[\log\frac{p\left(X,Z|\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)}-\log\frac{p\left(Z|X,\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)},\right],\\
&=\sum_Z p\left(Z|X,\Theta\right)\log\frac{p\left(X,Z|\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)}-\sum_Z p\left(Z|X,\Theta\right)\log\frac{p\left(Z|X,\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)}.
\end{aligned}
$$

The latter part of the equation

$$\sum_Z p\left(Z|X,\Theta\right)\log\frac{p\left(Z|X,\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)}\geq 0$$

is a Kullback-Leibler divergence which is greater than or equal to 0. It is only equal to 0 when $\hat{\Theta}=\Theta$. As a result, the first part of the equation is a lower bound on the data log likelihood

$$\log p\left(X|\hat{\Theta}\right)\geq\sum_Z p\left(Z|X,\Theta\right)\log\frac{p\left(X,Z|\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)}.$$

What's more, this lower bound is exactly what is maximised with respect to $\hat{\Theta}$ in each maximisation step. The Kullback-Leibler divergence, on the other hand is set to 0 in each Expectation step by setting $\Theta=\hat{\Theta}$. Which means that when we find a new $\hat{\Theta}$ in the following maximisation step, we have improved our lower bound

$$\sum_Z p\left(Z|X,\Theta\right)\log\frac{p\left(X,Z|\hat{\Theta}\right)}{p\left(Z|X,\Theta\right)}\geq\sum_Z p\left(Z|X,\Theta\right)\log\frac{p\left(X,Z|\Theta\right)}{p\left(Z|X,\Theta\right)}.$$

Additionally, it is possible to use EM to obtain MAP estimates of the parameters by including a prior in the model. This does not

modify the expectation step and only results in a minor change of the maximisation step, similar to the inclusion of a regularisation term. To conclude this chapter, we will present two examples of EM applied to real machine learning models. We will begin with the optimisation of the Gaussian Mixture Model, and we will conclude with the optimisation of the hyper-parameters in Bayesian Linear Regression.

EM training for the Gaussian Mixture Model

Now, we return to the GMM model, which was defined as follows.

$$\begin{aligned} k &= 1, \ldots, K, \\ n &= 1, \ldots, N, \\ \sum_{k=1}^{K} \pi_k &= 1, \\ p\left(z_{nk} = 1\right) &= \pi_k, \\ p\left(\boldsymbol{x}_n | z_{nk} = 1\right) &= \mathcal{N}\left(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \Sigma_k\right). \end{aligned}$$

The parameters of the model, that have to be optimised, are the following

$$\Theta = \left\{\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K\right\}.$$

The expectation step is a straightforward application of Bayes's rule. However, we would like to stress that computing $p\left(Z|X, \Theta_p\right)$ for all Z, for which K^N possibilities exist, is not required. Instead, one should simply compute $p\left(z_{nk}|\boldsymbol{x}_n, \Theta_p\right)$ for all clusters and all data-points individually. This quantity will be used directly in the maximisation step. Recall that in this maximisation step, we want to find the optimum parameter setting for $\hat{\Theta}$ in

$$\sum_{Z} p\left(Z|X, \Theta\right) \log p\left(X, Z|\hat{\Theta}\right),$$

where Θ contains our current parameter estimates. This can be ex-

panded as follows

$$\sum_{z_1 k} \sum_{z_{2k}} \dots \sum_{z_{Nk}} \left(\prod_{n=1}^{N} p\left(z_{nk} | \boldsymbol{x}_n, \Theta\right) \sum_{n=1}^{N} \left[\log p\left(\boldsymbol{x}_n | \hat{\boldsymbol{\mu}}_{z_{nk}}, \hat{\Sigma}_{z_{nk}} \right) + \log p\left(z_{nk} | \hat{\pi}_k\right) \right] \right).$$

Which becomes much less involved when we use the fact that each term in the sum of the log likelihood

$$\log p\left(\boldsymbol{x}_n | \hat{\boldsymbol{\mu}}_{z_{nk}}, \hat{\Sigma}_{z_{nk}} \right) + \log p\left(z_{nk} | \hat{\pi}_k\right)$$

depends only on one z_{nk}, while the other values are summed out. This yields the following equation to maximise

$$\sum_{n=1}^{N} \sum_{z_{nk}} p\left(z_{nk} | \boldsymbol{x}_n, \Theta\right) \left[\log p\left(\boldsymbol{x}_n | \hat{\boldsymbol{\mu}}_{z_{nk}}, \hat{\Sigma}_{z_{nk}} \right) + \log p\left(z_{nk} | \hat{\pi}_k\right) \right].$$

To optimise this equation with respect to the parameters $\hat{\Theta}$, we only have to consider the part where the specific parameter occurs. As a result to tune $\hat{\pi}_k$, we only have to consider the following equation, where we have added a Lagrange multiplier to enforce the constraints that the different $\hat{\pi}_k$ must sum to one

$$\sum_{n=1}^{N} p\left(z_{nk} | \boldsymbol{x}_n, \Theta\right) \log p\left(z_{nk} | \hat{\pi}_k\right) + \lambda \left(\sum_{j=1}^{K} \hat{\pi}_j - 1 \right).$$

When we take the derivative of this equation and set it equal to 0, we obtain the following expression

$$0 = \sum_{n=1}^{N} p\left(z_{nk} | \boldsymbol{x}_n, \Theta\right) \frac{1}{\hat{\pi}_k} + \lambda.$$

Which can be re-arranged to

$$\hat{\pi}_k = \frac{\sum_{n=1}^{N} p\left(z_{nk} | \boldsymbol{x}_n, \Theta\right)}{-\lambda}.$$

Where λ can be computed by using the constraint that the $\hat{\pi}_k$ must

sum to one

$$1 = \sum_{k=1}^{K} \hat{\pi}_k = \sum_{k=1}^{K} \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)}{-\lambda}.$$

Which results in the following expression for λ

$$\lambda = -\sum_{k=1}^{K}\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right) = -N.$$

As a result, the maximisation step for π_k computes the following

$$\hat{\pi}_k = \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)}{N}.$$

We can follow an analogous procedure to optimise $\hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k$:

$$\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right) \log p\left(\boldsymbol{x}_n|\hat{\boldsymbol{\mu}}_{z_{nk}}, \hat{\Sigma}_{z_{nk}}\right).$$

Computing the gradient, setting it equal to zero and solving for $\hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k$ yields the following optimum values

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right) \boldsymbol{x}_n}{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)},$$
$$\hat{\Sigma}_k = \frac{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right) \left(\boldsymbol{x}_n - \mu_k\right)\left(\boldsymbol{x}_n - \mu_k\right)^T}{\sum_{n=1}^{N} p\left(z_{nk}|\boldsymbol{x}_n, \Theta\right)}.$$

These are the update equations that we discussed previously and concludes our discussion of EM based optimisation of the parameters in a GMM.

EM for hyper-parameter optimisation in Linear Regression

The EM algorithm can also be used to tune the hyper-parameters in Bayesian linear regression by assuming that $\boldsymbol{w}$ is the latent variable and α, β are the model parameters. Hence, in the maximisation step, we have to optimise

$$\mathbb{E}_{p(\boldsymbol{w}|X,\boldsymbol{y},\alpha,\beta)}\left[\log p\left(\boldsymbol{y}, \boldsymbol{w}|\hat{\alpha}, \hat{\beta}\right)\right].$$

To keep the notation uncluttered, we will drop the subscript in the expectation operator and we re-write the expected data log likelihood as follows

$$\mathbb{E}\left[\log p\left(\boldsymbol{y}, \boldsymbol{w}|\hat{\alpha}, \hat{\beta}\right)\right] = \mathbb{E}\left[\log p\left(\boldsymbol{y}|\boldsymbol{w}, \hat{\beta}\right)\right] + \mathbb{E}\left[\log p\left(\boldsymbol{w}|\hat{\alpha}\right)\right].$$

Hence, when we search for optimal values for β (α) we only have to consider the first (second) part. However, to be able to compute the expectation, we have to have the parameters of the posterior distribution on $\boldsymbol{w}$ given $X, \boldsymbol{y}, \alpha, \beta$. This posterior is characterised by the mean $\boldsymbol{w}_*$ and covariance A

$$\begin{aligned}\boldsymbol{A}^{-1} &= \beta \boldsymbol{X}\boldsymbol{X}^T + \alpha \boldsymbol{I},\\ \boldsymbol{w}_* &= \beta \boldsymbol{A}\boldsymbol{X}\boldsymbol{y}\left(\boldsymbol{c}\right).\end{aligned}$$

Plugging this in the expectation, computing the gradient and solving gives us the final update equations

$$\begin{aligned}\hat{\alpha} &= \frac{D}{\boldsymbol{w}_*^T\boldsymbol{w}_* + \mathrm{Tr}\left(\boldsymbol{A}\right)},\\ \hat{\beta} &= \frac{N}{\mathrm{Tr}\left(\boldsymbol{A}^{-1}XX^T\right) + \left(X^T\boldsymbol{w}_* - \boldsymbol{y}\left(\boldsymbol{c}\right)\right)^T\left(X^T\boldsymbol{w}_* - \boldsymbol{y}\left(\boldsymbol{c}\right)\right)}.\end{aligned}$$

These update equations make a trade-off between a peaked prior distribution, which results in small weights and obtaining a good fit for the model, which results in a low variance on the predictions (i.e. a low mean squared error between the targets and our predictions).

2.4 Conclusion

In this chapter, we have briefly summarised the most important machine learning concepts such as generalisation, over-fitting, under-fitting, cross-validation and regularisation. Additionally, we have introduced the EM algorithm, Gaussian Mixture Models, Linear Discriminant Analysis and Linear regression. This knowledge will be used in the following chapters to develop and analyse our machine learning framework for ERP based BCI.

3
A unified probabilistic model for ERP based BCI

Since the inception of the original ERP based BCI by Farwell and Donchin (1988), many researchers within the BCI community have strived to increase the accuracy and spelling speed. This effort has resulted in the development of improved and well performing but rather complex (supervised) machine learning models (Cecotti and Gräser, 2010; Rakotomamonjy and Guigue, 2008). Most of these methods have to be fine-tuned on rather large quantities of subject-specific data. In turn, this limits their applicability in practical experiments.

What's more, the current state of the art models are not limited to a basic ERP decoder. Instead, they are composed of a basic decoder, e.g. swLDA, that is combined with language models (Speier et al., 2012). Additionally, dynamic stopping strategies are included in the model (Speier et al., 2012; Schreuder et al., 2011b; Höhne et al., 2010; Lenhardt et al., 2008; Liu et al., 2010; Jin et al., 2012; Verschore et al., 2012). The dynamic stopping methods attempt to increase the efficiency of the entire BCI setup by stopping the stimulus presentation when the classifier has received enough data to be confident in its prediction. We have discussed dynamic stopping methods in Section 1.3.6. For a more detailed comparison we refer to the excellent work of Schreuder et al. (2013).

The problem with this new generation of decoders is that, in most cases, the basic classifier, the dynamic stopping strategy and the language models are separate entities, which are tied together by additional heuristics. In turn, this increases the number of tunable parameters which increases the engineering effort and the data dependency

during the cross-validation procedures to make them work reliably.

In this chapter, we will introduce a unified probabilistic model, which integrates language statistics and dynamic stopping in a natural way (Kindermans et al., 2013). The proposed model has to be calibrated for each individual subject but it does not require the tuning of additional parameters or extensive cross-validation.

This probabilistic model is the backbone of this thesis. In the following chapters, we will demonstrate how this model can be trained without label information (Kindermans et al., 2012b) and how it can be extended to share decoding information between subjects (Kindermans et al., 2012a, 2014). While the model in this chapter is a basic but powerful supervised model, it can easily be extended and will become a true zero-training method by the end of this dissertation.

Despite the unsurpassed capabilities of the unified model, it is rather simple and based on just three assumptions. The first assumption is based on the work of Blankertz et al. (2011), who reported that ERP features are approximately Gaussian with class dependent mean and shared covariance. We have made this assumption less strict. We assume that there exists a one-dimensional projection of these features which is Gaussian with class dependent mean and shared variance. The second assumption considers the prior probability of the desired symbol. In a basic model, each symbol and therefore each stimulus is assumed to be equiprobable. However, we can alter this assumption easily to incorporate language statistics. Our third and final assumption is introduced to limit the complexity of a model. Indeed, a machine learning approach can only be reliable when it is properly regularised (see Section 2.1 and (Farquhar and Hill, 2013; Tomioka and Müller, 2010; Bishop, 2007)). For this reason, we introduce a Gaussian zero mean isotropic covariance prior on the weight vector of the projection. These three assumptions result in a probabilistic model that can be trained efficiently that exhibits a natural dynamic stopping strategy based on probabilistic inference.

In the remainder of this chapter, we will present our unified model, which will be validated in an extensive offline study.

3.1 Building a unified model

Before we define the model, we will introduce our notation. Each selection in an ERP speller corresponds to a trial $t = 1, \ldots, T$. During this trial t, the attended symbol is $c_t = 1, \ldots, C$. Each trial consists of several stimulus iterations $i = 1, \ldots, I$, in which each stimulus $j = 1, \ldots, J$ is presented once to the user. The data recorded after stimulus j in iteration i of trial t is $\boldsymbol{x}_{i,j,t}$. Using this notation, we will define the probabilistic model.

3.1.1 Defining the basic model

Our first assumption considers only the attended symbol. In the basic version of this model, we assume that all symbols are equally likely and this can be formalised as follows:

$$p\left(c_t\right) = \frac{1}{C}, \tag{3.1}$$

where c_t is the desired symbol in trial t and C is the number of symbols to choose from, which is 36 in the standard matrix speller used in our validation.

Next, we turn to our assumption on the data. We assume that it is possible to project the data into a single dimension, in which it will be Gaussian with a class dependent mean. The projection is denoted by $\boldsymbol{w}$. The projection of the data $\boldsymbol{x}_{t,i,j}^T \boldsymbol{w}$ is assumed to be Gaussian with variance β^{-1}. The mean depends on the class and is encoded by the function $y_{t,i,j}\left(c_t\right)$. If the desired symbol c_t for trial t is intensified during stimulus j in iteration i then $y_{t,i,j}\left(c_t\right) = 1$, otherwise $y_{t,i,j}\left(c_t\right) = -1$. Furthermore, we use the distribution of the projection of the data to approximate the distribution of the EEG

features[1]:

$$p\left(\boldsymbol{x}_{t,i,j}|c_t, \boldsymbol{w}, \beta\right) = \mathcal{N}\left(\boldsymbol{x}_{t,i,j}^T \boldsymbol{w}|y_{t,i,j}\left(c_t\right), \beta^{-1}\right).$$

The assumption of the Gaussian projection in one dimension is backed by Blankertz et al. (2011), where the authors argue that the EEG features are approximately Gaussian distributed with a class dependent mean and shared covariance. Consequently, each one-dimensional projection is Gaussian with a class dependent mean and shared variance. Finally, the approximation of the distribution of the EEG by that of the projection might seem strange, since it is not a true generative model (i.e. we cannot sample from this probabilistic model). To make it generative, we would have to model the data in the original features space, similar to the approach of LDA. However, our approximation does not reduce the discriminative power of the model compared to the standard LDA model because that is also a linear model. Moreover, this approximation will allow us to incorporate (discriminative) transfer learning directly into the model, which would not be possible in a standard LDA model. We would like to point out that this formulation implicitly assumes that for each class the EEG features are i.i.d. given the true symbol. Which, in practice, might not always be the case, for example when the desired symbol is highlighted during 2 consecutive stimulus presentations because the stimulus onset asynchrony is typically shorter than the ERP response itself. Hence, subsequent feature values might be correlated or even dependent. However, most machine learning approaches, e.g. Linear Discriminant Analysis, employ this simplification.

Last but not least, we will place a prior on the projection such that we can regularise:

$$p\left(\boldsymbol{w}\right) = \mathcal{N}\left(\boldsymbol{w}|\mathbf{0}, \alpha^{-1} I\right),$$

where $\alpha^{-1} I$ is the covariance matrix on the weight vector. This prior

[1]Please note that this approximation can be replaced by a distribution on the projected features, which is analogous to a regression model. In that case, we would have to perform inference by conditioning on the projection of the EEG and not on the EEG itself. We opted for this approximation because the model becomes more intuitive since it allows us to condition directly on the observed EEG.

with zero mean and isotropic covariance is introduced to keep the weights of the classifier small, which in turn keeps the complexity low, see Section 2.2.2. Bringing these three assumptions together gives us the following model:

$$
\begin{aligned}
p\left(c_t\right) &= \frac{1}{C} \\
p\left(\boldsymbol{x}_{t,i,j} | c_t, \boldsymbol{w}, \beta\right) &= \mathcal{N}\left(\boldsymbol{x}_{t,i,j}^T \boldsymbol{w} | y_{t,i,j}\left(c_t\right), \beta^{-1}\right) \\
y_{t,i,j}\left(c_t\right) &= \begin{cases} 1 & \text{if } c_t \in \text{stimulus } j \\ & \text{of iteration } i \\ -1 & \text{otherwise} \end{cases} \\
p\left(\boldsymbol{w}\right) &= \mathcal{N}\left(\boldsymbol{w} | \boldsymbol{0}, \alpha^{-1} I\right)
\end{aligned}
$$

Our model is closely related to Linear Discriminant Analysis (LDA) and Bayesian Linear Regression (see Chapter 2 and (Bishop, 2007)). The assumption that the projected EEG has a normal distribution with a class dependent mean is slightly more general than LDA, where the EEG itself is assumed to have a normal distribution with class dependent mean but with shared covariance. The Bayesian Linear Discriminant Analysis (BLDA) model (Hoffmann et al., 2008), which is often used in an ERP speller, is very similar to our model but differs slightly in the underlying assumptions. Despite its name, the BLDA model is not an enhanced LDA model but it is based on regression, whereas we modified the assumptions such that it becomes a real classifier. This is reflected in the inference of the desired symbol. BLDA uses regression outputs, which are averaged across iterations and the symbol associated with the maximum value is predicted. As a result, combining BLDA with language models and dynamic stopping is quite cumbersome. The model presented here uses Bayesian inference to predict the desired symbol, which makes it more natural to use it in combination with language models and dynamic stopping.

3.1.2 Training the model

To train the model, we will use an Expectation Maximization (EM) based training algorithm (see Section 2.3.2), that is completely analogous to EM for Bayesian Linear Regression. We refer to Section 2.3.2 for the derivation of the update equations. In this EM based training, the latent variable is the weight vector $\boldsymbol{w}$ and the parameters to be estimated are α, β. During training, we alternate between updating the estimate of the weight vector and updating the hyper-parameters until convergence. The advantage of this approach is that it does not require cross-validation procedures and it results in a properly regularised highly accurate model, as we will demonstrate later.

The update for the weight vector consists of computing the posterior distribution of the weight vector given the training data, which has mean $\boldsymbol{w}_*$ and covariance $\boldsymbol{A}$ according to:

$$\begin{aligned} \boldsymbol{A}^{-1} &= \beta X X^T + \alpha \boldsymbol{I}, \\ \boldsymbol{w}_* &= \beta \boldsymbol{A} X \boldsymbol{y}(\boldsymbol{c}), \end{aligned}$$

where the columns of X are the individual data-points $\boldsymbol{x}_{t,i,j}$ and $\boldsymbol{y}(\boldsymbol{c})$ is a column vector that contains the different labels $y_{t,i,j}(c_t)$.

The updates for the hyper-parameters are as follows:

$$\begin{aligned} \alpha &= \frac{D}{\boldsymbol{w}_*^T \boldsymbol{w}_* + \mathrm{Tr}(\boldsymbol{A})}, \\ \beta &= \frac{N}{\mathrm{Tr}\left(\boldsymbol{A}^{-1} X X^T\right) + \left(X^T \boldsymbol{w}_* - \boldsymbol{y}(\boldsymbol{c})\right)^T \left(X^T \boldsymbol{w}_* - \boldsymbol{y}(\boldsymbol{c})\right)}, \end{aligned}$$

where D is the dimensionality of the weight vector and N is the number of feature vectors used to train the model.

3.1.3 Inferring the attended symbol

After training, we can use the model in the BCI setup. The goal is always to infer the most likely symbol given the data. In our model, this is a straightforward application of Bayes rule. We will use the mean $\boldsymbol{w}_*$ of the posterior distribution on $\boldsymbol{w}$ as parameter of our model. Taking the fully Bayesian treatment and integrating over $\boldsymbol{w}$ is also a

possibility, but in our experience this does not improve the results. We will drop the subscript and refer to the mean of the posterior simply as $\boldsymbol{w}$. Thus, given this weight vector, predicting the desired symbol for trial t conditioned on the EEG is done as follows:

$$\begin{aligned} p\left(X_t|c_t, \boldsymbol{w}, \beta\right) &= \prod_{i=1}^{I}\prod_{j=1}^{J} p\left(\boldsymbol{x}_{t,i,j}|c_t, \boldsymbol{w}, \beta\right), \\ p\left(c_t|X_t, \boldsymbol{w}, \beta\right) &= \frac{p\left(X_t, c_t|\boldsymbol{w}, \beta\right)}{p\left(X_t|\boldsymbol{w}, \beta\right)}, \\ &= \frac{p\left(c_t\right) p\left(X_t|c_t, \boldsymbol{w}, \beta\right)}{\sum_{c_t} p\left(c_t\right) p\left(X_t|c_t, \boldsymbol{w}, \beta\right)}. \end{aligned}$$

The predicted symbol c_t is the most likely one. Furthermore, it is important to note that $p\left(X_t|c_t, \boldsymbol{w}, \beta\right)$ can be computed for any number of iterations. Therefore, we are able to estimate the most likely symbol after each iteration. Hence, using the likelihood of the most probable symbol results in a intuitive dynamic stopping strategy, which we will discuss in Section 3.1.5.

3.1.4 Incorporating language information

Initially, we assumed that all symbols receive the same prior probability. However, in practice this is not the case. For example, when the user has already spelled the text: "I WOULD LIKE TO DRIN", then you can be fairly certain that the next symbol will be "K". To incorporate such knowledge into the model, we have to make use of language models. In this work, we make use of N-gram letter models, which take the $N-1$ previously spelled characters into account to compute the prior probability of the next symbol:

$$p\left(c_t|c_{t-N+1}, \ldots, c_{t-1}\right). \tag{3.2}$$

We will use the shorthand h_t^{N-1} to denote the history of length $N-1$: $c_{t-N+1}, \ldots, c_{t-1}$, thus, Equation 3.2 becomes:

$$p\left(c_t|h_t^{N-1}\right). \tag{3.3}$$

To exploit statistical knowledge about language in the model, all

we have to do is replacing Equation 3.1 by Equation 3.3 in the definition of the model.

To keep inference simple, we will assume that the user has to correct erroneous mistakes with a backspace. Consequently, we know that text spelled previously is either correct or has to be erased. This allows us to compute the attended symbol as follows:

$$p\left(c_t|X_t, h_t^{N-1}, \boldsymbol{w}, \beta\right) = \frac{p\left(c_t|h_t^{N-1}\right) p\left(X_t|c_t, \boldsymbol{w}, \beta\right)}{\sum_{c_t} p\left(c_t|h_t^{N-1}\right) p\left(X_t|c_t, \boldsymbol{w}, \beta\right)},$$

where c_t can be either a *symbol* or *backspace*. First, it is important to note that the complexity of this operation does not depend on the number of previous symbols that the language models considers. Thus, embedding the language model into the classifier does not result in a computational penalty, which is a key aspect of this approach, as it allows us to utilise language models with a long history (up to 8-gram models in our work). However, when there is uncertainty about the previously spelled symbols then the inference becomes much more involved because we have to rely on a forward backward algorithm (Bishop, 2007) to infer the most likely symbol. That algorithm scales exponentially with the length of the language model. Nevertheless, that approach will be discussed in the context of unsupervised learning in Chapter 5. As the goal of this chapter was to keep the model simple, we will only consider the former setting.

We would like to point out that our method is not limited to a specific language model implementation. The only requirement is that we must be able to obtain the probability of each symbol in the grid conditioned on the history. The language models in this work are N-gram letter models, with N ranging from 1 to 8. The most direct way to learn such a model is to count how many times a specific sequence occurs, followed by normalizing these counts:

$$\hat{p}\left(c_t|h_t^{N-1}\right) = \frac{\text{count}\left(h_t^{N-1}, c_t\right)}{\sum_{c_t} \text{count}\left(h_t^{N-1}, c_t\right)}.$$

Clearly, a sequence that is not included in the training corpus will receive no probability mass and we will not be able to spell this se-

quence with the BCI. On top of that, the memory consumption scales exponentially with the length of the history. Both these problems can be solved at once by using Kneser-Ney smoothing (Kneser and Ney, 1995).

This technique stores only sequences occurring K or more times, the probability mass assigned to unseen sequences is proportional to D occurrences. Thus, for sequences occurring K or more times:

$$\hat{p}\left(c_t|h_t^{N-1}\right) = \frac{\text{count}\left(h_t^{N-1}, c_t\right) - D}{\sum_{c_t} \text{count}\left(h_t^{N-1}, c_t\right)}$$

The remaining probability mass divided across sequences that appear fewer than K times can be computed as follows:

$$\hat{p}\left(D|\, h_t^{N-1}\right) = 1 - \sum_{c_t} \hat{p}\left(c_t|h_t^{N-1}\right),$$

where we summed over all c_t for which count $\left(h_t^{N-1}, c_t\right) \geq K$. We will divide $\hat{p}(D)$ across the sequences that we counted less than K times proportional to the probability mass they receive in an $(N-1)$-gram model:

$$\hat{p}\left(c_t|h_t^{N-1}\right) = \hat{p}\left(D\right) \frac{\hat{p}\left(c_t|h_t^{N-2}\right)}{\sum_{c_t} \hat{p}\left(c_t|h_t^{N-2}\right)},$$

where we have summed over all c_t for which count $\left(h_t^{N-1}, c_t\right) < K$. In this work we have used the suggested values from literature for the discount $D = 2$ and the minimum number of occurrences $K = 4$. We trained the models on the corpora included in the NLTK toolbox (Bird, 2006).

Previously we pointed out that the user has to correct mistakes by using a backspace command. For that reason, we have to estimate the likelihood for the backspace command conditioned on the history. This is non-trivial because it can't be learned from data. Therefore, we have devised the following heuristic. The probability that we made a mistake in the last trial according to the language model is the following:

$$\hat{p}(\textit{backspace}|h_t^{N-1}) = \sum_{c_n \neq c_t} \hat{p}(c_n|h_t^{N-1}),$$

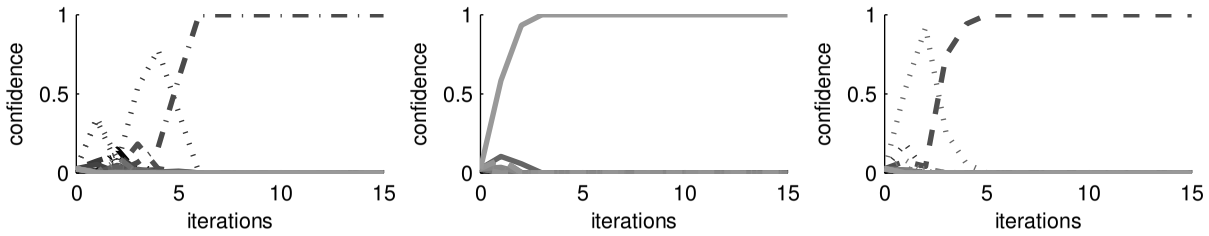
where c_t is the symbol spelled in trial t. We combine $\hat{p}(backspace)$ with the language model prior as follows. We compute the likelihood that the previous symbol was wrong according to the language model. Then, we add this as an additional state to the language model, which results in a total probability mass larger than 1. Therefore, we have to re-normalise the language model. This yields the following updated language model

$$p(backspace|h_t^{N-1}) = \frac{\hat{p}(backspace|h_t^{N-1})}{1 + \hat{p}(backspace|h_t^{N-1})},$$
$$p\left(c_t|h_t^{N-1}\right) = \frac{\hat{p}\left(c_t|h_t^{N-1}\right)}{1 + \hat{p}(backspace|h_t^{N-1})},$$

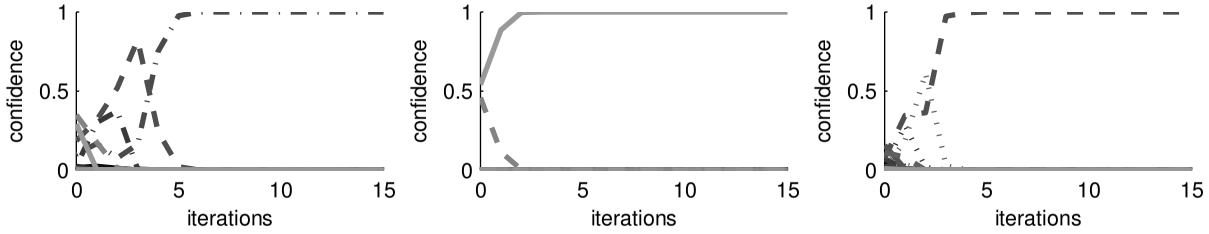
and we will use this distribution during symbol prediction. The estimation of the backspace prior has the following properties. The original language model probabilities remain proportional to eachother. The backspace probability is independent of the classifier output, which is essential in order to be able to correct mistakes where the user selected the wrong symbol. Additionally, less than half the probability mass can be assigned to the backspace command. This makes sense if we assume that the speller works accurately. Finally, spelling unlikely texts will increase the probability of a backspace.

3.1.5 Increasing the communication speed

Our final contribution in this chapter enriches the speller with dynamic stopping by using the approach presented originally by Verschore et al. (2012). Our goal was, similar to the incorporation of language models and the basic speller itself, to keep everything as straightforward as possible. As we pointed out in the previous sections, our classifier is able to output a likelihood for each symbol after each iteration. Therefore, after each iteration, we know which symbol the classifier prefers and how much confidence it has in this prediction. The evolution of likelihood of the symbols as a function of the number of iterations is plotted in Figure 3.1a. The first two plots show the typical behaviour: once the classifier settles on a symbol, the likelihood for that symbol approaches 1 quickly. Therefore it is



(a) Without language model



(b) With an 8-gram letter model

Figure 3.1: Evolution of character probabilities for three different trials as a function of the number of iterations. Each line denotes the estimated probability of a single symbol. In the top row, we have used a classifier without a language model, the bottom row uses an 8-gram language model. Even though both approaches predict the same symbol after 15 iterations, the language model based prediction requires fewer iterations to become confident about its prediction.

sensible to set a confidence threshold and stop presenting stimuli when this threshold is reached. But one has to select this threshold carefully. For example, the last plot shows some pathological behaviour, the classifier becomes very certain of a specific symbol, after which it starts to home in on another symbol. This shows how critical the threshold selection is. Setting it too low results in making additional mistakes. On the other hand, when we set it too high, we waste time during unneeded iterations. The default method to set the stopping threshold, or any parameter in a machine learning method, is cross-validation. For the dynamic stopping threshold, this turns out to be problematic. First of all, we cannot re-use the training data to optimise the threshold, because the classifier might be overly confident on this data. As a result, we need additional calibration data to optimise the threshold. Second, when we use few data-points, then a single mistake can effect the spelling speed drastically. Therefore, using a small dataset to optimise the threshold is not reliable. Recording a large additional dataset to optimise the threshold is also a bad idea, since doing so will waste more time recording this dataset than we gain by optimising the threshold. Therefore, we use a subject-independent fixed threshold. Remember that in Figure 3.1a, we saw that the likelihood for a specific symbol approaches 1 quickly when the classifier receives enough data to support this selection. Therefore we argue that setting the threshold to 0.99 for all subjects is a good choice. It is a very conservative threshold, thus it is very unlikely that by adding more iterations the classifier will predict a different symbol.

We will show that dynamic stopping by itself can improve the performance. However, it becomes much more powerful when combined with a language model. Recall that a language model rescales the posterior probabilities over the symbols based on the history. When we combine dynamic stopping with language models, the language model effectively lowers the confidence threshold for expected, common sequences. Unlikely or strange letter combinations will result in an increased threshold. In essence, we modulate how conservative the threshold must be with respect to the previously spelled text. This effect is illustrated in Figure 3.1b, where we have plotted the posterior probabilities computed on the same data as Figure 3.1a. It is clear that a symbol, that is likely according to the language model, reaches

the threshold more quickly. Equally important, symbols that receive a low prior probability can get spelled as well. The language model steers the symbol prediction but does not overrule it.

3.2 Offline evaluation

3.2.1 Experimental setup

Data

In our offline validation, we utilise three different publicly available datasets, comprising data from a 6x6 visual matrix speller. These datasets are: BCI Competition II (Blankertz et al., 2004), BCI Competition III (Blankertz et al., 2006b) and the Akimpech P300 dataset (Yanez-Suarez et al., 2012). Details on these datasets are given in Appendix C.

We argue that the best approach to compare methods is using publicly available data such that each method is evaluated on the same data. This eliminates influences/effects of non-stationarity from the subject's mental state. However, because these datasets do not contain real textual content, evaluating language model based approaches directly on these publicly available datasets is not possible. On top of that, we will show that the spelled text can greatly influence the accuracy of a language model based decoder. Therefore we argue that it is essential that each technique, that makes use of language models must be evaluated on different texts per subject. Hence, even when the datasets would have contained real text, the evaluation would have been suboptimal.

Dataset augmentation

To solve the aforementioned problem, we have utilised the dataset augmentation approach that we proposed in (Kindermans et al., 2012a). Recall that in an ERP speller, the task is to predict the desired symbol by detecting the attended stimulus. Furthermore, the mapping from stimulus to symbol is always done in post-processing, i.e. a look-up table. Thus, in order to select the correct symbol, we only have to

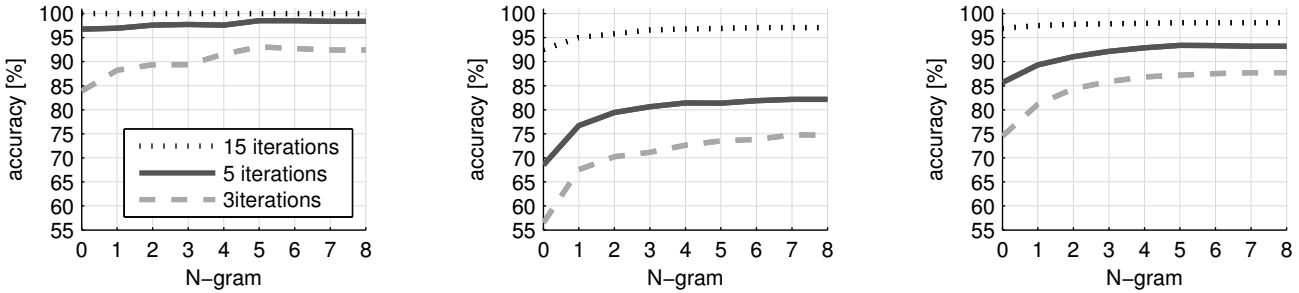
detect the stimulus that caused the ERP response, not the symbol itself. As a consequence, we can dynamically modify the underlying lookup table and this allows us to remap the spelling actions associated with the different stimuli on the fly. This approach enables us to modify the desired output without interfering with the raw classification accuracy (when no language model is used). On top of that, this dataset augmentation scheme enables us to simulate backspace functionality in offline experiments. The end result is that we can fully emulate a user who is trying to spell a text, thereby correcting his spelling mistakes. We would like to stress that we do not alter the intensification structure nor the EEG itself. We will only predict the correct symbol when the correct position in the grid is identified as the attended stimulus, just as in the unmodified dataset.

We want to emulate a user spelling a specific text and the user has to be able to correct his mistakes. Therefore, we will allocate the following symbols to the P300 grid (left to right, top to bottom): $A-Z$,$1-4$, a *backspace* command, $6-9$ and a *space* command. This grid will be cyclically shifted on a character by character basis, such that the desired command is assigned to the true P300 position. The cyclic shifts make sure that the neighbouring symbols/commands are always the same, which makes sure that errors where the neighbouring intensifications are selected do not depend on the desired symbol. During the offline simulations, we determined the next desired action and remapped the grid, after eac spelled symbol. This was done until we had processed all the data once.

To obtain evaluation texts, we sampled from a Wikipedia dataset (Sutskever et al., 2011). This dataset was first transformed to uppercase, then we dropped the symbols that were not present in the grid. For each subject we sampled 20 contiguous texts randomly, where each text has the same length as the original dataset. Each classifier was evaluated on all texts for all subjects to limit the influence of the desired text on the performance.

	Fixed Optimal				LM + Fixed Optimal				Dynamic Stopping				LM + Dynamic Stopping			
Dataset	I	LM	Acc.	SPM	I	LM	Acc.	SPM	I	LM	Acc.	SPM	I	LM	Acc.	SPM
BCI Comp. II	1	-	77.42	4.6	1	5	83.54	5.7	2.7	-	90.3	4.5	1.99	7/8	96.13	6.02
BCI Comp. III	9	-	85.50	1.7	3	7/8	74.57	2.6	7.6	-	88.5	2.2	4.74	7/8	92.50	3.40
Akimpech	4	-	81.80	2.9	2	7/8	80.22	4.3	5.5	-	95.0	3.3	3.50	7/8	96.34	4.68

Table 3.1: Results with and without a language model for a fixed optimal number of iterations and dynamic stopping. I is the number of iterations, LM designates language model, Acc. is the accuracy and SPM represents the number of symbols per minute.



(a) BCI Comp. II **(b)** BCI Comp. III **(c)** Akimpech

Figure 3.2: Influence of the language model on the accuracy for a fixed number of iterations per symbol.

Pre-processing

Throughout this thesis, we will use the following pre-processing method, which is not subject-specific, for the publicly available datasets. The preprocessing method is applicable online since the EEG is preprocessed on a character by character basis. The first step comprises the application of a common average reference filter, followed by a band-pass filter (0.5 - 15 Hz). Then we sub-sample the data by a factor 6, to around 40 Hz. Subsequently, we retain 10 samples per channel centred around 300 ms after the stimulus presentation. In the final step, we append a bias term to the feature vector.

Experiments

We ran experiments with and without a language model. The language models ranged from uni- to 8-gram models. We evaluated a fixed number of iterations, ranging from 1 to 15, and compare the results to dynamic stopping. In the dynamic stopping approach, we stopped the stimulus presentation when the classifier reached a certainty of at least 0.99. This parameter was selected as a conservative threshold with a clear intuitive meaning.

3.2.2 Results and Discussion

We begin by looking at the evolution of the accuracy for 3, 5 and 15 iterations as a function of the length of the N-gram model. The average results per dataset are plotted in Figure 3.2. In this figure, we observe that the language model is able to improve the accuracy, but the improvement is the largest for a low number of iterations. For example, on the BCI Comp. III dataset, 3 iterations without a language model results in an average accuracy of only 56.5%, and the 8-gram language model boosts this up to 74.8%. But when we use 15 iterations on the same dataset, the result without language models is already 92.5% and the margin for improvement is much smaller. The results obtained on the other two datasets corroborate these findings. A second important observation is that when we go from a classifier without a language model to one with a uni-gram embedded into it, we make a leap in selection accuracy. Going from a uni-gram to a

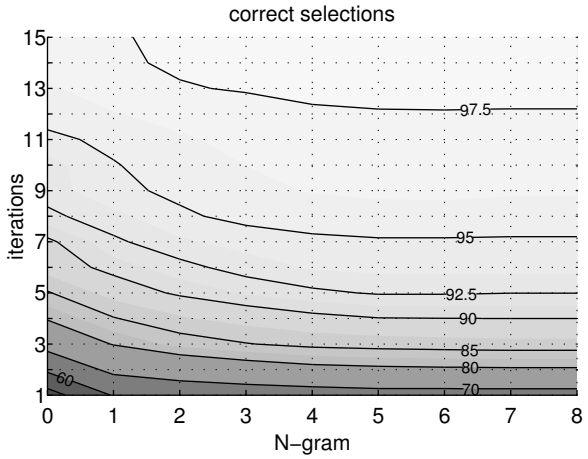


Figure 3.3: Spelling Accuracy averaged over all subjects. For a given number of iterations, using language models increases the spelling accuracy.

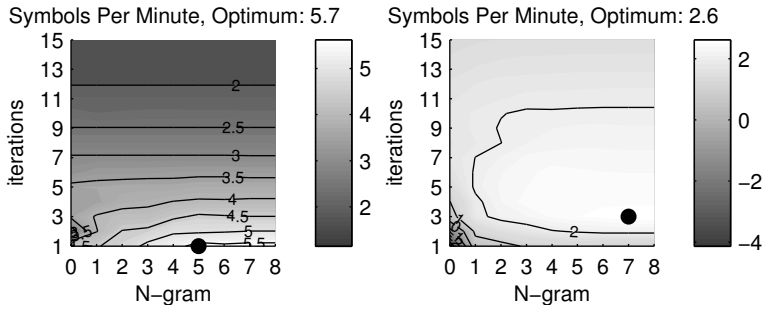
bi-gram yields a smaller performance gain. For 7 and 8-gram models we actually obtain the same selection accuracy. This has two causes. First, a uni-gram already conveys most prior knowledge about the language. Second, the longer the history of the language model, the more difficult it becomes to train it.

To illustrate the impact of a language model on the spelling accuracy in a different way, we have plotted the average accuracies over all datasets, for 1 to 15 iterations and over all language model settings, in Figure 3.3. The highest accuracies are obtained with either 5, 6, 7 or 8-gram models, depending on the number of iterations per symbol. Let us fix the desired minimum average accuracy and determine how many iterations are required per symbol. If the desired accuracy is 90%, then we can use a 7-gram language model and 4 iterations per symbol, only half of what is needed without a language model. We can obtain 85% accuracy without a language model and 6 iterations, but by using at least a 4-gram language model we need no more than 3 iterations. Once more, we observe that the language model has the most effect when few iterations are used to predict the symbol and, that going from no language model at all to a uni-gram model has a bigger effect than going from a 5 to a 6/7/8-gram model.

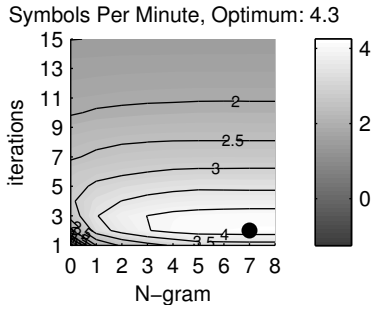
In the previous paragraph, we have concentrated on the aver-

age performance. However in BCI, the results are always subject-dependent and when language models are used, the desired text can have a huge influence on the performance. When the desired text is likely under the language model, then the language model will have a larger (positive) impact on the accuracy than unlikely sequences. This effect is illustrated in our experiments where we have evaluated the speller using 20 different sequences sampled from Wikipedia. We analyse the results in detail for the first subject from BCI Comp. III. Each sequence for this specific subject is 100 trials long and we use 5 iterations per trial. Our model without a language model obtains 64% accuracy, the best result from literature is 72% with the eSVM approach from Rakotomamonjy and Guigue (2008), a technique that requires very complicated subject-specific tuning. By adding an 8-gram language model to our classifier, we obtain an average accuracy of 74.8% with a standard deviation of 10.7. Out of the 20 sampled texts, only 4 resulted in a selection accuracy below 72%: 71%, 67%, 58% and 52%. From these 4, only 2 texts result in a performance drop compared to our classifier without a language model. On the other hand, 16 texts allow us to outperform eSVM, 11 texts yield an accuracy of over 80% and three of these got to at least 90%. This analysis illustrates the big influence a language model can have. For that reason, we conclude that one has to be very cautious when evaluating language models in a BCI. We argue that when you evaluate a method that utilises a language model, you must evaluate different texts for each subject and each classifier setting. Additionally, our results indicate that a language model adapted to the specific language employed by the BCI user, should yield additional performance improvements.

Next, we want to determine how the language model affects spelling speed. In Figure 3.4, we have averaged the Symbols Per Minute (SPM) results per dataset, and we have marked the optimum per dataset with a dot. Take note that we selected the optimum on the test-set. Therefore, the optimum is not representative of what you can attain in an online experiment. We use it to show the maximum level of performance. With the test-set based optimisation, we see that on the BCI Comp. III and the Akimpech dataset the maximum SPM value is obtained with 7/8-gram language model. Additionally, we



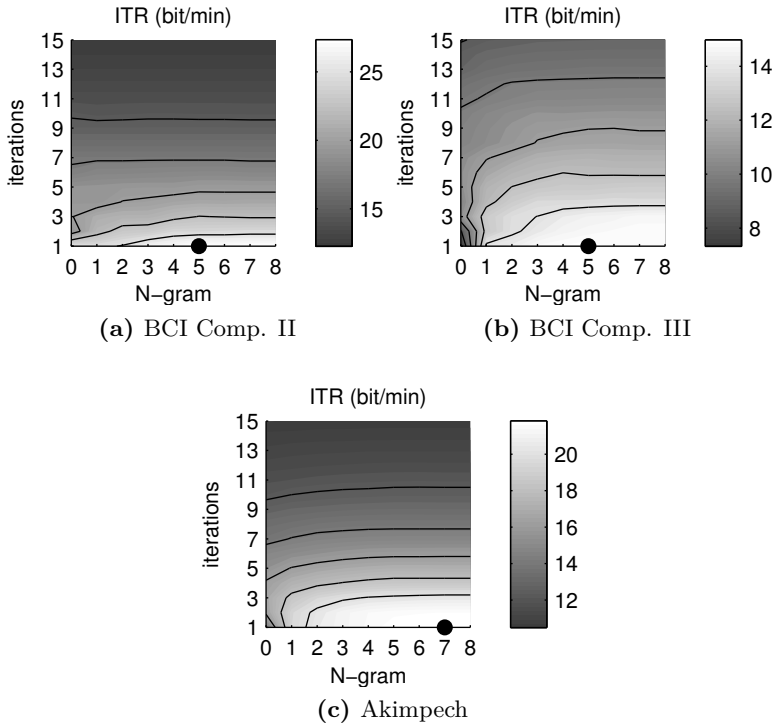
(a) BCI Comp. II **(b)** BCI Comp. III



(c) Akimpech

Figure 3.4: The number of symbols per minute spelled as a function of the language model and the number of iterations per symbol. The optimum is marked with a dot. Negative scores correspond to over 50% selection error, this results in an accumulation of mistakes that cannot be corrected.

see that for a fixed number of iterations per symbol, a more complex language model results in an increased number of symbols per minute, which corroborates our previous findings on selection accuracy. In Table 3.1, we have compared the optimum SPM value on the test set, with and without a language model. We see that on BCI Competition II, we obtain 4.6 SPM without a language model and a single iteration. The addition of a language model increases this to 5.7 SPM. The BCI Comp. III dataset is much more difficult and this is reflected in an optimum of only 1.7 SPM with 9 iterations without n-gram. The addition of a 7/8-gram increases this result to 2.6 SPM with just 3 iterations. Without a language prior, we can spell 2.9 symbols per minute with 4 iterations on the Akimpech dataset.



(a) BCI Comp. II **(b)** BCI Comp. III

(c) Akimpech

Figure 3.5: The average information transfer rate (ITR) for each dataset. A dot marks the optimum, note that this optimum does not coincide with the SPM optimum (Figure 3.4). The ITR over-estimates the information transfer as it does not consider the fact that spelling errors have to be corrected. Note that this figure is included to show that ITR can result in an unusable optimum.

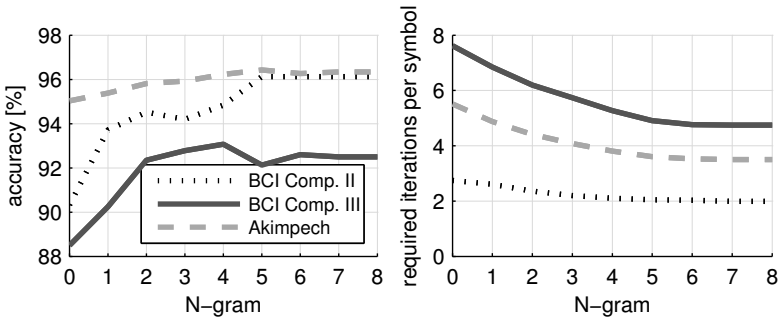
Incorporating a 7/8-gram increases to 4.3 SPM with just 2 iterations per symbol. Therefore we conclude that the language model increases the SPM drastically.

Next, we illustrate the deficit of the Information Transfer Rate, to demonstrate why it is not appropriate to use as an error measure for ERP-BCI. In Figure 3.5, we see that apart from BCI Comp II., which contains only a single subject, the parameters that maximise the ITR differ from those that maximise the SPM, e.g. on BCI Comp III. the optimal ITR setting results in just 1.25 SPM, which is less than half the true SPM optimum of 2.6. On all datasets the ITR is maximised

with a language model and a single iteration per symbol. This is caused by the ITR definition, ITR favours a low time per symbol because it assumes that a mistake equals no information transfer, whereas it is actually negative information that has to be corrected. We argue that because of this deficit and the fact that the ITR is not interpretable, ITR is not suited for ERP speller evaluation.

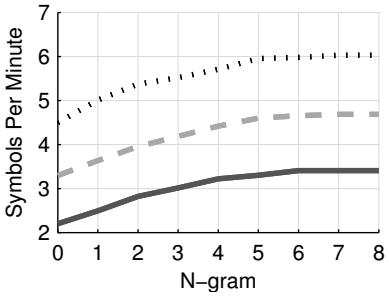
Now, we will compare dynamic stopping, with a certainty threshold of 0.99, to a fixed optimal setting for the number of iterations. We compare both techniques with and without a language model and present the results in Table 3.1, where in addition to the SPM results, we have included the (mean) number of iterations, the accuracy and the language model used. Without a language model, the fixed optimal number of iterations resulted in 4.6, 1.7 and 2.9 SPM on BCI Comp. II, BCI Comp. III and the Akimpech dataset, respectively . With dynamic stopping we obtain 4.5, 2.2 and 3.3 SPM, which is only a minor decrease for the BCI Comp. II dataset, but a steady increase on the other 2 datasets. Here, we would like to stress once more that the fixed optimal setting was optimised on the test-data, whereas the dynamic stopping uses a fixed threshold, which is interpretable and reflects the expected behaviour of the classifier. Therefore, the dynamic stopping results are truly representative of an online experiment. When we look at the accuracy, we see that dynamic stopping results in a higher accuracy than the fixed optimal setting on all datasets. We obtain 90.3%, 88.5% and 95.0% on the three datasets. Therefore, it is clear that our dynamic stopping criterion results in a more reliable speller. However, there is still room for improvement, as the SPM results for dynamic stopping without a language model are still worse than the fixed optimal with language model.

The effect of language models on our dynamic stopping approach is illustrated in Figure 3.6. We see that in general, there is an increase in accuracy and a decrease in the required number of iterations per symbol. Obviously, this results in an increase of the number of symbols spelled per minute. This brings us to our final result, the entire model attains accuracies of 96.1%, 92.5% and 96.3% on the BCI Comp. II, BCI Comp. III and Akimpech datasets. The average number of iterations is 1.99, 4.74 and 3.50 respectively. The number of symbols spelled per minute becomes 6.02, 3.40 and 4.68,



(a) Accuracy

(b) iterations per symbol



(c) Symbols Per Minute

Figure 3.6: Performance with dynamic stopping as a function of the language model. The average accuracy increases and the average number of iterations per symbol decreases when we use a language model with a longer history. As a result, we see an improvement in the number of symbols that we can spell each minute.

which is a relative increase of 30%, 100% and 60% over the standard fixed optimal model. This improvement is the result of the combination of language models and dynamic stopping, these two techniques cooperate to improve accuracy and spelling speed.

Comparing these results with other research is difficult, as most BCI researches use their own proprietary datasets. Combined with the fact that results are highly subject- and even session-dependent, this makes it impossible to compare with the numbers given in these papers. On BCI Comp III. data, the best published result for 5 iterations is an accuracy of 73.5% or in 1.8 SPM using eSVM (Rakotomamonjy

and Guigue, 2008). Here, we would like to point out that this approach uses subject-specific pre-processing parameters and that incorporating dynamic stopping in the eSVM approach is difficult, because it uses an ensemble of SVM classifiers. Our own semi-supervised approach (Kindermans et al., 2011), also uses subject-specific pre-processing and achieves 76%, which results in 2 SPM. With dynamic stopping and 7/8-gram, we used on average 4.74 iterations per symbol to obtain an accuracy of 92.5% which boils down to 3.4 SPM. On the Akimpech dataset, the default classifier in BCI2000 (Schalk et al., 2004), swLDA, achieves 98.8% with 15 iterations. There are no results published for a lower number of iterations. With dynamic stopping and a language model, we used 3.5 iterations on average to obtain 96.34% accuracy. Even with this significantly reduced number of iterations, we are very close to the swLDA results.

3.3 Conclusion

In this chapter we have shown how an unified probabilistic model can be used to detect the ERP response reliably. During development, we did not only focus on performance, but also on simplicity. The result is a single probabilistic model, which has to be trained in a supervised manner, but which does not require subject-specific fine-tuning of the parameters or extensive cross-validation procedures. Furthermore, this probabilistic model, which is based on three straightforward assumptions, can incorporate language statistics and the probabilistic nature of the output results in a direct dynamic stopping strategy with an interpretable threshold.

The experimental validation of this model has shown that language models are effective at improving the performance of ERP based spellers. Furthermore, we have shown that the biggest gains can be made with simple (i.e. uni-gram models) but that extending the history that is considered by the language model can further enhance the performance. However, there is an important remark to be made. The influence of the language models depends heavily on the difficulty of the EEG data itself and on the desired text. A user for whom the ERP response is easily detected will benefit less than a more difficult

user. Furthermore, a text that is likely, according to the language model, will result in a higher performance gain than an unlikely text. Therefore, we argue that it is necessary to evaluate language model based decoders on a wide array of texts, such that the performance bias from the text itself can be limited. The final performance improvement comes from the inclusion of dynamic stopping, which can only reach its true potential in combination with a language model. With dynamic stopping and language models, we achieved a relative increase of 30%, 100% and 60% in SPM over the fixed optimal number of iterations without a language model on respectively the BCI Comp. II, BCI Comp III. and Akimpech datasets.

However, it is not the simplicity, nor the performance of this model which is the main contribution of this chapter. The model presented here is above all a building block, which will be used throughout the remainder of this thesis. First, we will introduce an unsupervised learning algorithm, which allows the model to be trained on the fly, without label information. Second, an extension of this model allows us to incorporate transfer learning to share decoding information across the different subjects. Third, we will analyse how and why this model actually allows for unsupervised and transfer learning.

4 Unsupervised Learning in an ERP based BCI

In the previous chapter we have presented a unified probabilistic model for ERP based BCI, that integrates a dynamic stopping strategy and language information. Despite the gains in spelling speed and accuracy, the usability of the BCI remains rather limited due to the dependency on a tedious calibration session. This is a major hindrance for ERP spellers, and indeed for BCIs in general. The calibration session is required to obtain the labelled data that common supervised algorithms use to learn to decode the subject's data. During this calibration recording, the user is instructed to focus on specific stimuli. As a result, the calibration session is not productive, it does not provide a direct benefit for the user. This becomes problematic, especially for patients with a limited attention span who truly need a BCI (Li et al., 2011).

For this reason, the BCI community has spent a lot of effort on reducing the dependency on labelled data. The goal of this research is to build a so-called zero-training or calibration-less BCI. A first approach to reduce the dependency on labelled data is to increase the signal to noise ratio of the signal. Examples of this approach for visual ERP include the optimisation of the visual stimuli (Tangermann et al., 2011, 2012; Kaufmann et al., 2011). A second approach is the introduction of adaptive methods, which are able to improve a suboptimal supervised trained classifier during online usage (Kindermans et al., 2011; Dähne et al., 2011; Li et al., 2008; Vidaurre et al., 2011a). Furthermore, these adaptive methods can cope with non-stationarity in the data, where the distribution of the recorded signals changes,

e.g. due to fatigue (Shenoy et al., 2006; Quionero-Candela et al., 2009; Sugiyama and Kawanabe, 2012). As a result, these adaptive methods can outperform static methods during long-term use.

What we propose here goes beyond basic adaptation. Instead of using a pre-trained model that is adapted online, we propose to forego the initial calibration session and use a system which learns online without any label information. At the start of a session, such a system is oblivious to the information content in the EEG data. It has to figure out how to decode the signals while the user is interacting with the BCI. This is a challenging unsupervised learning problem. Nevertheless, we were able to solve this problem, and by the end of a spelling session, the unsupervised model is as reliable as state of the art supervised decoders.

The key to solving this unsupervised learning problem is the exploitation of constraints (Kindermans et al., 2012b). In the ERP based BCI, the constraints are imposed by the paradigm, but it has also been shown that constraints can be used to learn efficiently in the context of natural language processing (Chang et al., 2007; Mann and McCallum, 2008; Liang et al., 2009). Additionally, constraints are known to improve decoding in the prediction of finger flexion based on ECOG (Wang et al., 2011). We will show that such an unsupervised model can be as reliable as a state of the art supervised decoder.

The concept of exploiting application constraints to facilitate unsupervised learning will be discussed in the next section. Afterwards, we will detail how the constraints are embedded in the unified probabilistic model. This is followed by information on how to train the model. After the technical discussion, we will evaluate the model offline on visual ERP data from three different datasets. On top of that, we will present the results from an online study using the auditory ERP paradigm AMUSE (Schreuder et al., 2010). This online study was executed at the TU-Berlin in collaboration with Michael Tangermann, Martijn Schreuder and Klaus-Robert Müller. In the final section, we will give a concise overview of the advantages and limitations of unsupervised spelling for ERP based BCI. The next chapter is devoted to improving upon this basic unsupervised model.

4.1 Learning without label information

Exploiting the repetitive structure of the ERP paradigm is key to unsupervised training in an ERP based BCI (Kindermans et al., 2012b). To understand this, it is best to look at the limitations of common unsupervised learning algorithms. Typically, unsupervised learning is used for clustering (i.e. dividing the data points into groups such that each data point in a group is similar) (Hartigan and Wong, 1979; Bishop, 2007), feature learning (Ranzato et al., 2007; Coates and Ng, 2012), dimensionality reduction (Jolliffe, 2005; Schölkopf et al., 1998), or as a general data analysis tool (Blei et al., 2003). Unsupervised learning is not commonly used for classification. These unsupervised methods are capable of uncovering structure in the data, but they are not able to assign the correct label to the different components. This is illustrated in Figure 4.1, where we generated data from a mixture model. This mixture model comprises two Gaussian components with a shared covariance structure but separate means. This generative process follows the assumptions made by Linear Discriminant Analysis (LDA), which is a supervised classifier, and those made by a Gaussian Mixture Model (GMM), which is an unsupervised clustering approach. After training, both the supervised LDA model and the GMM model can discriminate between the two groups. However, only the LDA model can point to the target class, the GMM model cannot since it does not have information about the labels. However, in the case of ERP, with a 5 to 1 ratio of non-target to target data-points and an unrealistically low amount of noise, then we could use this knowledge in the GMM to select the target class. This is a first constraint that aids unsupervised classification in ERP based BCI. However, when the noise level increases and the class overlap becomes larger, it is no longer possible to reliably discriminate between the target and non-target clusters.

Real EEG data can cause a standard GMM model to fail, but we have not used all the side-information that we can extract from the paradigm. To illustrate this, we present a second toy example, which can be seen in Figure 4.2. Here we present data from a toy ERP paradigm with 3 stimuli and 3 iterations per trial. The stimuli are

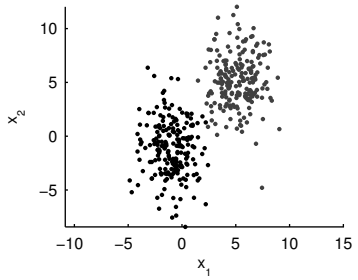


Figure 4.1: Discriminating between the two clusters can be done easily. However, assigning a meaning (e.g. which are the target and the non-target responses) is not possible without additional (label) information.

shown on the top line of panel A and the pictures in the second line represent the EEG data. When we look at the data from the first iteration, we observe a flower, a cactus and tree as ERP responses. These three responses are all distinct, they could all be targets, non-targets or outliers, and there is no information to determine which is which. By increasing the number of iterations, we obtain additional data points. This brings us closer to the previous example. However, the increased amount of data is not the only source of information that we can exploit. In panel B, we have grouped the responses per stimulus. Recall that in an ERP paradigm a single stimulus type elicits the target response and all other stimuli are followed by a non-target response. In our case, we see the *cactus-response* appearing each time we present stimulus 2. The other stimuli result in more general *plant-responses*. Consequently, by using the application constraints, we can (with high confidence) assign the target label to the *cactus-response*.

4.1.1 The probabilistic model

The next step is to modify the probabilistic model presented in the previous chapter, to incorporate the paradigm information. Even though we did not stress it in the previous chapter, these constraints are already present in our unified model. Moreover, this information is always used when the calibration dataset is labelled and when we perform inference in the model. Therefore, we simply re-visit the ba-

Figure 4.2: Illustration on how the constraints facilitate true unsupervised learning in ERP paradigms. The pictures represent EEG data, which can be either target or non-target ERP responses. The goal is to detect the target stimulus without using label information. The paradigm states that target and non-target ERP responses are distinct. Furthermore, there is only one stimulus that must result in the target response, all other stimuli have to elicit non-target responses. In panel **A**, we have shown the stimuli and their corresponding ERP responses. When we analyse the data from the first iteration we cannot make a confident descision because all responses are very distinct. However, adding data from additional iterations facilitates the problem. Especially when we group the data per stimulus (panel **B**).

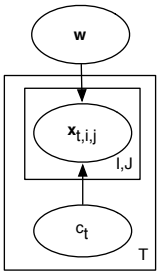


Figure 4.3: Graphical representation of the probabilistic model used for unsupervised training.

sic version of the unified probabilistic model, without the language model extension for simplicity.

Let $\boldsymbol{x}_{t,i,j}$ be the feature vector recorded during trial t in stimulus iteration i after stimulus presentation j. There are C different stimuli, which are all equally likely to be the attended stimulus c_t. The EEG is projected by $\boldsymbol{w}$ onto $y_{t,i,j}(c_t)$, where the value of $y_{t,i,j}(c_t)$ depends on whether it is a target or non-target stimulus. Furthermore, we assume that these projections have a per class variance β^{-1}. Finally, we include a zero mean isotropic covariance prior to regularise the model. The full model is visualised in Figure 4.3 and can be written as follows.

$$
\begin{aligned}
p(c_t) &= \frac{1}{C} \\
p(\boldsymbol{x}_{t,i,j}|c_t, \boldsymbol{w}, \beta) &= \mathcal{N}\left(\boldsymbol{x}_{t,i,j}^T \boldsymbol{w} | y_{t,i,j}(c_t), \beta^{-1}\right) \\
y_{t,i,j}(c_t) &= \begin{cases} 1 & \text{if } c_t \in \text{stimulus } j \\ & \text{of iteration } i \\ -1 & \text{otherwise} \end{cases} \\
p(\boldsymbol{w}) &= \mathcal{N}\left(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1} I\right)
\end{aligned}
$$

In this model, the constraints posed by the repetitive structure of the paradigm are contained in the function $y_{t,i,j}$. This function

couples all data points that belong to the same trial and allows us to perform inference at the level of the attended stimulus, which is easier than performing inference for each individual data point. A single trial in a paradigm with 6 different stimuli and 15 iterations per trial generates 90 (15 iterations per trial and 6 stimuli per iteration) data points for a single trial. There are $2^{90} > 10^{27}$ possible options to label the individual data points. However, a labelling is only feasible when there is a single stimulus that always results in a target response. All other stimuli must result in non-target responses. This reduces the number of possible labellings to just 6, regardless of the number of iterations in the trial. Hence, by using the application constraints, we have significantly reduced the search space. We would like to stress once more that using these constraints in itself is not new. It is used since the inception of the ERP speller to label calibration data and it is always exploited during inference. On the other hand, the use of these constraints to build an unsupervised algorithm for ERP spellers is new and this is the main contribution of this chapter.

Since the model does not have to be modified, inference remains unchanged. We assume that $\boldsymbol{w}$ and β are known and apply Bayes's rule.

$$p\left(c_t|X_t, \boldsymbol{w}, \beta\right) = \frac{p\left(c_t\right) p\left(X_t|c_t, \boldsymbol{w}, \beta\right)}{\sum_{c_t} p\left(c_t\right) p\left(X_t|c_t, \boldsymbol{w}, \beta\right)}.$$

And the predicted symbol will be the most likely one.

4.1.2 Unsupervised training

The unsupervised training procedure consists of a combination of the Expectation Maximization (EM) algorithm (Dempster et al., 1977) and direct maximisation of the likelihood. In the EM algorithm, we treat the attended stimulus as latent variable and the weight vector $\boldsymbol{w}$ and the class-wise variance of the projection β^{-1} as parameters of the model. This is different from the supervised setting, where $\boldsymbol{w}$ was the latent variable. Using the EM algorithm, we optimise the expected data log likelihood of the model:

$$\boldsymbol{w}, \beta = \arg\max_{\boldsymbol{w},\beta} \sum_{\boldsymbol{c}} p\left(\boldsymbol{c}|X, \boldsymbol{w}, \beta\right) \log p\left(X, \boldsymbol{c}|\boldsymbol{w}, \beta\right) + \log p\left(\boldsymbol{w}|\alpha\right).$$

To obtain the update equations, we compute the gradient with respect to the parameters and set it equal to zero. The derivations for the update equations are available in Appendix A and we will only discuss the resulting updates.

First, the update for $\boldsymbol{w}$

$$\hat{\boldsymbol{w}} = \sum_{\boldsymbol{c}} p\left(\boldsymbol{c}|X, \boldsymbol{w}, \beta\right) \left(XX^T + \frac{\alpha}{\beta} I\right)^{-1} X\boldsymbol{y}\left(\boldsymbol{c}\right).$$

The resulting updated classifier is equal to a weighted sum of regularized linear regression classifiers. The weight of each classifier equals the probability that the attended stimuli were correct given the previous estimate of $\boldsymbol{w}$. The EM algorithm estimates how likely each stimulus is to be the attended one and subsequently modifies the model such that it maximizes its belief.

The update for β^{-1} is similar. The variance of the projections is updated such that it equals the expected mean squared error between the target projections and the actual projection:

$$\hat{\beta}^{-1} = \left\langle \sum_{c_t} p\left(c_t|X, \boldsymbol{w}, \beta\right) \left(\boldsymbol{x}_{t,i,j}^T \boldsymbol{w} - y_{t,i,j}(c_t)\right)^2 \right\rangle_{t,i,j}.$$

Finally, the additional hyper-parameter α is optimised by direct maximum likelihood, which is possible because it only depends on $\boldsymbol{w}$. The update for α^{-1} is equal to the average squared classifier weight

$$\hat{\alpha}^{-1} = \frac{\boldsymbol{w}^T \boldsymbol{w}}{D},$$

where D is the dimensionality of the weight vector.

We would like to pay special attention to regularisation in this model. First, when we regularise too much and α goes to infinity, the model will collapse onto the degenerate solution $\mathbf{0}$. But the regularisation does not only depend on α, it also depends on β. When β is large, which means that β^{-1}, the variance of the projection is small, we will regularise less. This allows us to use larger weights and a more complex model when we obtain a good fit for the data. However, when we are not able to fit the data well, we will regularise more and keep the solution simple. This approach makes sense in the

unsupervised setting. The better we understand the data, the better we can specify our model.

4.1.3 Post-hoc classification

The EM procedure introduces adaptation in the model. After each EM iteration, the estimation of the most likely attended stimuli can change. During the first few iterations in a batch setting, or during the first few trials in an online setting, these estimates are usually off. The classifier can revise its initial faulty output as more and more data is collected and used during subsequent classifier updates. We will denote the updated estimate as post-hoc or updated spelling predictions. Furthermore, this approach allows us to use the speller in an online setting, where only user errors have to be corrected. A user error occurs when the user focusses on the wrong stimulus, not when the decoder is at fault. When post-hoc classification is used, the user can decide to rely on the classifier to correct the decoding mistakes later. As a result, he can gain time by continuing spelling as if there was no mistake at all.

4.1.4 Unsupervised classifier ranking

There is one final aspect that we did not yet consider. The EM algorithm optimises the likelihood of the data. Hence, there is no guarantee that the classifier will actually learn how to solve the correct task. By exploiting the application constrains in the optimisation process, we force the algorithm to choose a single stimulus as target for each trial. All the other stimuli must be non-target stimuli. This partially enforces the algorithm to discriminate between the two classes and prevents it (in most cases) from collapsing on the prior. But this alone does not allow to detect whether the classifier has been trained correctly.

We have observed that there is a high correlation between the data log likelihood of the optimised model and the accuracy. To illustrate this, we made scatter plots of the performance of the classifier versus the data log likelihood (Figure 4.4). From this figure, it is clear that it is possible to select the good performing classifiers simply by look-

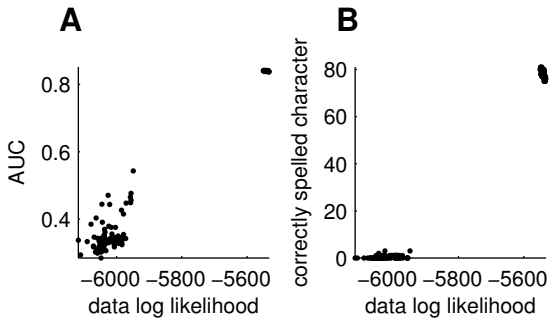


Figure 4.4: Scatter plots showing the quality of the classifier. Quality is measured in either AUC or characters predicted correctly versus the data log likelihood. The data used in this plot is created in the UB experiment on the data from the second subject from BCI Competition III using 5 repetitions.

ing at the data log likelihood, which can be computed without label information. This plot was generated based on a total of 100 different initialisations of $\boldsymbol{w}$. Data from subject B in BCI Competition III with just 5 iterations per trial was used.

A final important observation is that, due to local minima, we find many classifiers that have very low AUC values on this challenging data-set. For this reason we will use groups of initialisations to ensure that we maximise the probability of finding a classifier that performs well. Obviously, the number of initialisations required depends on the signal to noise ratio of the data. When an easier dataset is processed, we will require less initialisations. The same applies for the number of iterations per trial, the higher this number the easier it becomes to solve the problem.

We would like to point out that the data log likelihood is not the only heuristic that can be used to select the best classifier. The expected variance of the projection β^{-1} heavily influences the likelihood. As a result, selecting the classifier with the smallest expected variance is also a valid strategy.

4.2 Offline evaluation

The initial evaluation of the novel unsupervised approach will be performed on visual ERP data. The goal of this evaluation is to find out the capabilities and limitations of unsupervised learning for ERP based BCI. Here, we will focus on the accuracy and reliability. In what follows, we will briefly describe the datasets, the preprocessing and the different variations of unsupervised learning which will be evaluated. Afterwards, we will discuss the results of our experiments.

4.2.1 Experimental Setup

Data

For this evaluation, we will use the same datasets as in the previous chapter, i.e. is the BCI Competition II (Blankertz et al., 2004), the BCI Competition III (Blankertz et al., 2006b) and the Akimpech dataset (Yanez-Suarez et al., 2012). All three datasets were recorded using the standard 6×6 visual speller with 15 iterations per trial. These datasets contain 25 different subjects in total. For more information on these datasets, we refer the reader to Appendix C. The advantage of using the BCI Competition III dataset is that many approaches have been published that use this dataset. This allows for a direct comparison with other (supervised) approaches to ERP spelling. On top of that, the BCI competition III dataset is a very challenging dataset even though it contains data from a visual ERP paradigm, which are in general less challenging.

Pre-processing

Pre-processing is identical to the supervised model from the previous chapter. We process the EEG trial per trial such that we can apply the method online. For each trial, we apply a common average reference filter, a bandpass filter (0.5 - 15 Hz). The resulting features are subsampled by a factor 6. A total of 10 samples per channel are retained, with the middle sample positioned 300 ms after stimulus presentation.

To facilitate the implementation of the decoder, a bias term is added afterwards.

(Un)supervised learning methods

Five different variants and two additional post-hoc re-evaluations of the unsupervised classifier are put to the test. The five approaches can be divided into two groups, where the first group contains batch training methods which are trained on the entire dataset at once. The goal here is to determine how well the model performs when many data points are available. The second group contains the online methods, which simulate real online experiments. In these online simulations, the classifier is fed the data one trial at a time, which significantly increases the difficulty of the learning problem.

Before discussing the specific aspects of the different methods, we will summarize the aspects the different methods have in common. All experiments will be executed on the same data but restricted to 5, 10 and 15 iterations per trial. This allows us to investigate the influence of the amount of data on the resulting performance. The unsupervised methods are always initialised randomly. To account for the variability caused by the initialisations, we will repeat each experiment 10 times. Furthermore, because the variability of the initialisation can impact the results heavily, we will use 10 classifier pairs per run (i.e. 20 classifiers per run). Each time we have to predict a symbol, we will select the best classifier based on the data log likelihood, as discussed in Section 4.1.4. The initialization of the model is very simple: $\boldsymbol{w}$ is drawn from $\mathcal{N}(\mathbf{0}, I)$, $\beta = 1.0$ and $\alpha = 0$. Each draw of $\boldsymbol{w}$ will be used to initialise two classifiers. One classifier will receive $\boldsymbol{w}$ as initial weight vector, the other classifier will be initialised with $-\boldsymbol{w}$. The idea of this approach is to ensure that one classifier performs better than guessing and one classifier performs worse. An overview of the evaluated unsupervised methods and their properties is given in Figure 4.5.

UB: Unsupervised batch training

In this first experiment, which is named UB, we perform unsupervised training on the test set. While the use of the test set

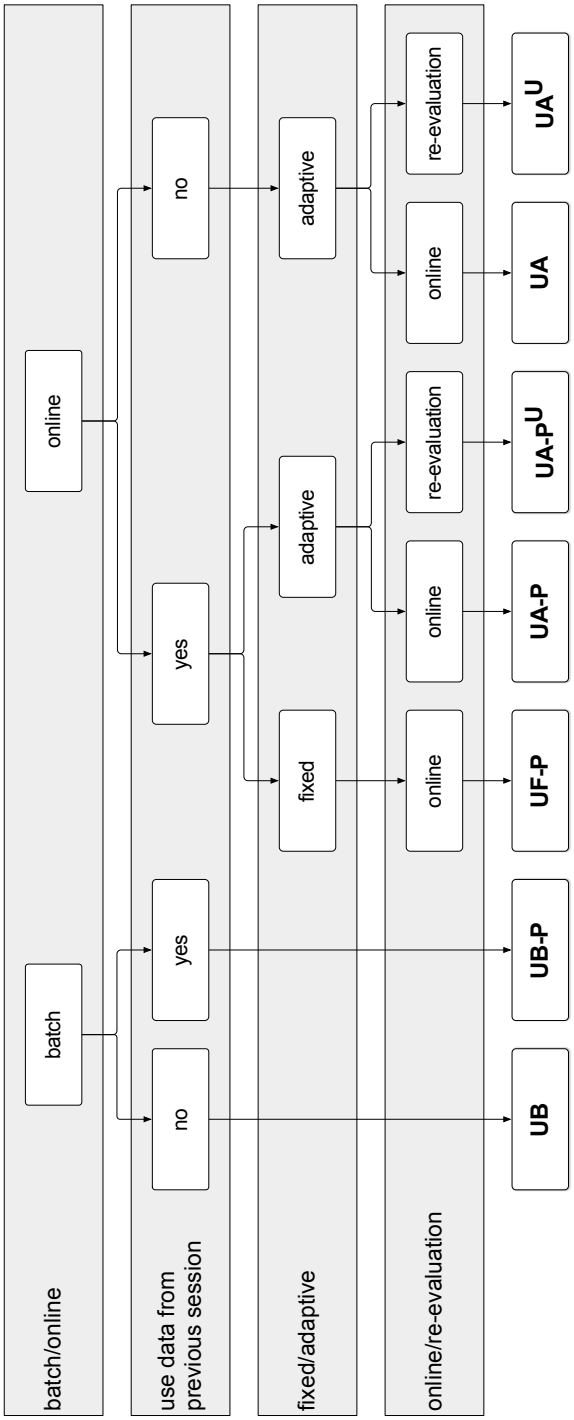


Figure 4.5: Overview of the unsupervised methods, their properties and the naming convention.

for training purposes may seem confusing at first, we would like to remind you that we do not use label information.

The UB approach is not applicable in an online BCI, but it allows us to examine the quality of the decoder after unsupervised training and it enables us to assess how good the resulting classifier is at labelling datasets for which no ground truth is available. Such unlabelled datasets are generated each time the user truly utilises the BCI system in a so-called free spelling mode.

The classifier initialisation is standard and follows the approach introduced above. All classifiers are optimised until convergence or 500 EM iterations, whichever comes first. After training, we select the classifier with the highest data log likelihood for evaluation. The accuracies of all 10 experiments are averaged to obtain the final result.

UB-P: Unsupervised batch training with an extended unlabelled dataset

In the second experiment, we want to determine how the performance can be improved by using previously recorded (but unlabelled) data. The setup is almost identical to the UB experiment. The only difference between UB and UB-P is the data used for unsupervised training. In UB only the test set is used, in UB-P both the train set and the test set are used to train the model. We would like to stress that the label information of the train set was not used during training and only the test set was considered during evaluation.

UF-P: Unsupervised pre-training

UF-P is our first setup that can be applied on an online setting. Here, we will train the classifiers on the training set without using the label information. This experiment is designed to analyse the generalisation properties of the proposed model. This cannot be assessed using only the data on which the model is trained (even though it is unsupervised training). For example, even when all characters are predicted correctly, there might be some over-fitting due to the application constraints forcing

the model to assign the correct label to data points with outlier characteristics. Furthermore, if this approach is successful, it allows us to use the previous session(s) of a subject to pre-train an unsupervised model which can subsequently be used in a real online experiment. This experiment is implemented as follows. First, we train the classifiers using the UB approach on the train set. Afterwards, we select one classifier from each experimental run. The resulting 10 classifiers are subsequently evaluated on the test set.

UA-P: Unsupervised pre-training and adaptation

The logical next step is to introduce unsupervised adaptation to the UF-P approach. For this reason, we re-use the UF-P classifiers from the previous experiment in combination with EM updates to make it adaptive. Using this approach we want to investigate whether it is beneficial to update the classifier during an online experiment. During evaluation, the classifier will receive the EEG data trial per trial. Before predicting the attended symbol, the classifier will add the data from the last trial to the unsupervised training set. Hence, during the online simulation, the (unlabelled) dataset used for training will grow continuously. After adding the data point to the dataset, the classifier will update the model by executing 3 EM iterations. Subsequently, the classifier predicts the attended symbol. By comparing the UA-P and the UF-P result, we can see whether unsupervised adaptation is beneficial. However, this experiment does not show how the UA-P model, we obtained after processing the entire test compares to the non-adaptive UF-P model. To determine the reliability of the final model, we have to apply the post-hoc approach to spelling with the UA-P method, the post-hoc evaluation of UA-P will be denoted by the superscript "^{U}": UA-P^U.

UA: Online unsupervised adaptation from scratch

In our final experiment, we make the task as challenging as possible. The classifier is initialised randomly, and has to learn on the fly. This represents the use case where no data is present for the current user.

Analogous to the UA-P experiment, the classifier receives the data trial per trial and after each trial a symbol prediction has to be made. Just as in the previous experiments, we perform 10 experimental runs of this experiment. Learning with next to no data is extremely difficult, therefore, we made some changes. First of all, the regularisation parameter α is limited to 10^3. When little data is available, the risk of collapsing onto the degenerate solution, which is a vector of zeros, is rather high. Like in the previous experiments, we continue to use 10 classifier pairs per run. We perform 3 EM updates of each classifier after we add data from a new trial to the classifier. After these EM iterations, we select the best classifier based on the data log likelihood, we make our symbol prediction and move on the next trial. Before moving on to the next trial, we execute the following procedure to maximise the probability of obtaining a model which is above trained correctly. We re-initialise half the classifiers after each trial. Per pair, we select the classifier with the highest log likelihood. The other classifier within that pair, i.e. the one with the lowest log likelihood, is re-initialised using the values for α, β from the best classifier of the pair. The weight vector is re-initialised to $-\boldsymbol{w}$, where $\boldsymbol{w}$ is the weight vector of the best classifier. As a result, before starting the optimisation for the next trial, half the classifiers will obtain an AUC above 0.5 (i.e. better than random performance) and the remaining classifiers will obtain an AUC below 0.5.

SF: Supervised fixed

Finally, as a reference, we will also include the results obtained using the supervised training method from the previous chapter.

I	SF	UB	UB-P	UF-P	AF-P	UA-P^U	UA	UA^U
5	96.7	98.7 (1.7)	96.8 (0.0)	96.8 (0.0)	96.8 (0.0)	96.8 (0.0)	56.5 (5.5)	96.8 (0.0)
10	100	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)	83.5 (1.1)	100.0 (0.0)
15	100	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)	92.3 (1.7)	100.0 (1.0)

Table 4.1: BCI Competition II Spelling Accuracy. Percentage of correctly predicted characters. The first column indicates the number of repetitions per character. The values in braces are the standard deviation.

	I	eSVM	CNN-1	MCNN-1	SF	UB	UB-P	UF-P	AF-P	UA-P^U	UA	UA^U
A	5	72	61	61	64	46.8 (4.0)	69.0 (0.0)	64.2 (0.9)	66.5 (0.5)	69.0 (0.0)	9.0 (7.4)	18.0 (17.9)
	10	83	86	82	86	89.4 (1.1)	91.0 (0.0)	86.0 (0.0)	87.0 (0.0)	88.0 (0.0)	62.4 (4.1)	87.4 (0.5)
	15	97	97	97	94	95.8 (1.3)	96.0 (0.0)	94.0 (0.0)	96.0 (0.0)	96.0 (0.0)	86.6 (1.6)	96.0 (0.0)
B	5	75	79	77	73	76.3 (1.6)	79.0 (0.0)	75.0 (0.0)	75.0 (0.0)	79.0 (0.0)	53.0 (2.1)	80.5 (0.5)
	10	91	91	92	91	92.1 (1.3)	95.0 (0.0)	91.0 (0.0)	94.0 (0.0)	95.0 (0.0)	87.9 (0.6)	92.0 (0.0)
	15	96	92	94	91	95.2 (0.6)	95.0 (0.0)	92.0 (0.0)	94.0 (0.0)	95.0 (0.0)	87.3 (1.1)	96.9 (0.3)

Table 4.2: BCI Competition III Spelling Accuracy. Percentage of correctly predicted characters. The first column indicates the subject, the second column the number of repetitions per character. The mean spelling accuracy and the standard deviation over the 10 experimental runs is given.

I	SF	UB	UB-P	UF-P	AF-P	UA-P^U	UA	UA^U
5	85.6	87.6 (11.9)	88.3 (11.9)	86.8 (13.0)	87.9 (12.2)	88.6 (11.5)	61.2 (25.8)	85.7 (14.7)
10	93.4	96.9 (5.2)	96.5 (5.2)	95.6 (5.9)	96.9 (4.5)	96.7 (5.2)	85.6 (14.6)	97.3 (4.8)
15	96.9	98.8 (2.8)	98.8 (2.8)	97.8 (3.7)	98.3 (3.4)	98.8 (2.8)	93.1 (6.0)	98.8 (2.7)

Table 4.3: Akimpech Spelling Accuracies. Percentage of correctly predicted characters averaged out over subjects from the Akimpech dataset. The first column indicates number of of repetitions per character. The values in braces are the standard deviation computed over the means of the different subjects.

4.2.2 Results and Discussion

In our discussion we will address the experiments in the order they were introduced above, starting with the offline experiments and finishing with the online ones. We will use both the selection accuracy and in one experiment the Area Under Curve (AUC) as an error measure. The spelling results for BCI Competition II are given in Table 4.1, for the BCI Competition experiments III in Table 4.2. Where applicable we have given the mean and standard deviation for 10 different spellers. The results for the Akimpech dataset are available in Table 4.3, where we have averaged out the means over all subjects and the standard deviation over the subject means is given.

Batch Training

To start, we compare UB, which is trained on the unlabelled test set, to eSVM, the winner of BCI Competition III and the best performing method on this dataset (Rakotomamonjy and Guigue, 2008). When evaluated on the BCI Competition III with 15 iterations per trial, eSVM predicts 97 % of the symbols correctly for subject A and 96 % is spelled correctly for subject B. Our proposed unsupervised method obtains a slightly lower accuracy on average: 95.8 % for subject A and 95.2 % for subject B. The individual experimental runs showed very little variance and the accuracy ranged from 95 % to 98 % for A and 95 % to 97 % for B. This is a little higher than the supervised model SF, which we introduced in the previous chapter. This indicates that the unsupervised approach is able to learn reliable models when a large amount of data is available. Furthermore it shows that using the data-log likelihood is a good criterion to select the best performing classifiers.

When evaluated on the less challenging but smaller dataset from BCI Competition II, UB makes no mistake at all. Moreover, the UB approach can predict the attended stimuli without fault even when we reduce the number of iterations per trial to 10. When we reduce the number of iterations per trial to 5, the classifier makes at most a single mistake. However, reducing the number of iterations per trial on the more difficult BCI Competition III dataset has a larger im-

pact. When we present the classifier with only 10 repetitions per trial then UB achieves an average accuracy of 90.8 % and thus it performs marginally better then eSVM and the CNN based methods (Cecotti and Gräser, 2010). Additionally, the variance on the performance is still very low. By further reducing the number of iterations per trial to 5 the performance starts to suffer heavily and the variance on the accuracy starts to increase. For subject A, we obtain only 46.8 % accuracy while other (supervised) approaches get up to 72% accuracy. For subject B, the outcome is more positive and with 76.3% accuracy UB outperforms eSVM but not the CNN based method. The performance drop is a very important result. It demonstrates a major limitation of the proposed method. When the data is very challenging, such as the data from subject A in BCI Competition III, the method breaks down when we present it with a limited number of repetitions. Recall that exploiting the constraints posed by the paradigm is key to the unsupervised learning method. This constraint is contained in the repetitive structure of the iterations. By reducing the number of iterations, we reduce the influence of the constraints. We will show later that we can compensate for the lack of repetitions by analysing more trials.

The average accuracy obtained on the Akimpech dataset[1] with the standard swLDA classifier, which is used in BCI2000 (Schalk et al., 2004), is 98.1% for 15 iterations. UB performs slightly better with an average accuracy of 98.8%. Furthermore, in the subject-specific results we found that spelling is perfect for 18 out of 22 subjects. This is also slightly better than the swLDA method which predicts the attended stimuli correctly for 16 out of 22 subjects. After reducing the number of iterations to 10, the accuracy drops to 96.9 and no mistakes are made for 15 subjects. Even with just 5 iterations per trial, the average accuracy remains rather high at 87.6 % and we spell perfectly for 5 subjects.

For the final UB experiment, we want to determine how it compares to a classifier which is trained with label information on this data. To investigate this, we have trained a regression based classifier without regularisation on this data. This is identical to fixing

[1]These results are included in the subject-specific description of the dataset. Only exact figures are available for 15 epochs.

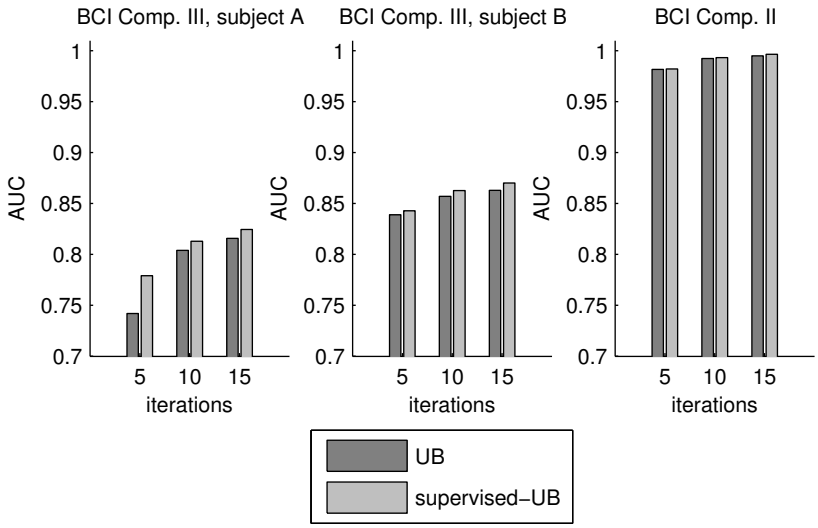


Figure 4.6: Bar graph showing the performance, measured in AUC, on the test. The classifier UB is trained unsupervised on the test set. The supervised-UB method uses label information and is allowed to over-fit on the test set by switching off regularisation.

the inferred probabilities for the characters to the true value, setting $\alpha = 0$ and training $\boldsymbol{w}$. In this experiment, we are not concerned with generalisation to unseen data, we want to determine how close the performance is to the maximum attainable level. We will evaluate this method using the AUC (for the discrimination on a single stimulus level). Furthermore, the evaluation is performed on the entire test set, even when the classifier is trained on limited number of trials. As a result only the classifiers trained by using 15 iterations per trial will have seen the entire test set before evaluation. We present the result in Figure 4.6, where we see that the UB approach is actually quite close to the supervised (over-fitting) classifier. Only on the data from subject A of BCI Competition III there is an increased performance gap. This result demonstrates that the unsupervised classifiers are almost as reliable as supervised classifiers trained on the same data. Therefore we can conclude that using labels is not really necessary when enough data is available.

After our initial experiment investigated how the performance is influenced by the availability of more unlabelled data. For this reason, we performed the unsupervised training jointly on the unlabelled train data and the test data in the UB-P experiment. A global observation is that by increasing the number of data points the performance improves too. Especially for the experiments where we limit the number of iterations per trial. However, this improvement is always rather small across all three datasets, apart from the experiment with subject A from BCI Competition III where 69 % of the symbols is spelled correctly by UB-P compared to only 47 % for the UB approach. Finally, there is an additional advantage to using more data. The number of initialisations that result in low quality spellers is reduced when we use more data. Giving a solid estimate for the number of initialisations required to obtain reliable performance is not possible because this is strongly subject-specific. We found that using 10 initialisations per experiment resulted in very stable results. Therefore we argue that using more initialisations is probably not beneficial in most cases. Only when we reduce the number of iterations per trial, the variance on the performance increases and using more initialisations might become beneficial. The exact number of initialisations used depends on the computation power that is available.

Simulation of Online Usage

Even though the previous experiments were very informative and have enhanced our understanding of the unsupervised approach, they are not truly representative of real BCI usage because they cannot be applied in a real online BCI. For this reason, we present a second set of experiments. In these experiments we want to demonstrate the flexibility of an unsupervised algorithm for ERP-BCI by evaluating three (main) variations on the unsupervised approach: the fixed pre-trained unsupervised classifier UF-P, its adaptive counterpart UA-P and the UA model that has to learn from scratch. Furthermore, the pre-processing technique used in these experiments is applicable online. This ensures that our simulations are truly representative of online experiments.

The first type of unsupervised model UF-P is trained offline and without label information on data from a previous session. After training the classifiers, we use the data log likelihood to select the best one and we apply it to the test set. This approach is identical to how supervised classifiers are used. As a result, there is no computational penalty for using the unsupervised approach.

The classifiers from the UF-P experiments are used as initialisation in the UA-P experiments. The difference is that the UA-P classifiers are adapted to the test-set during online usage by adding the EEG trial by trial to the unsupervised training set and performing 3 EM updates in between trials. The time needed to perform a single EM iteration (i.e. classifier update) scales linearly with both the number of characters and the number of iterations. In our experiments on the Akimpech (BCI Competition) dataset an EM updates takes 0.85 ms (3 ms) per iteration. When 15 iterations per trial are available, then we can execute 3 EM iterations for up to 52 (14) characters on the Akimpech (BCI Competition) data with a non-optimized Python implementation on a standard laptop. This difference between the two data-sets is caused by the difference in dimensionality of the feature vectors by using 64 and 10 channel EEG data.

The UA experiment has somewhat higher computational requirements for the EM updates. The classifier selection and spelling of the last character is almost instantaneous. But in contrast to the UA-P

approach, the UA classifier runs different models in parallel. This is necessary to obtain stable performance, especially on the more difficult datasets such as the one from BCI Competition III. However, the updates for the individual classifiers can be executed in parallel such that they do not increase the overall update time, but doing so would make the implementation more difficult. Additionally, when the classifier updates take too much time, they can be executed during the stimulus presentation of the next trial as well.

The results from the UF-P experiment indicate that delaying the classifier updates is a valid option because the unsupervised classifiers can generalise well to unseen data. Overall we observe a slight decrease in performance compared to the batch training scenario. Nevertheless, the results are near perfect on the data from BCI Competition II. At most one mistake is made when the number of iterations is reduced to 5. The data from BCI Competition III is clearly more challenging. On this dataset with 15 iterations per trial, UF-P (93.0 %) performs slightly worse than eSVM (96.5 %). When the number of iterations is reduced to 10, we observe the opposite effect: UF-P obtains 89 % accuracy and eSVM 87 %. For subject B the performance of eSVM and UF-P is identical in the most challenging setting with 5 iterations per trial (75 %). However, UF-P falls behind on subject A 64.2 %, whereas eSVM obtains 72%. In the analysis of the Akimpech dataset we notice the same effects: the performance is slightly lower than in both batch experiments with accuracies of 86.8%, 95.6% and 97.8% for 5, 10 and 15 repetitions. But for 15 iterations, this approach can compete with a supervised approach. All in all this experiment demonstrates that it is possible to build a classifier, that generalises well, without using label information.

In the second online experiment, we want to investigate whether adaptation is beneficial. The UA-P experiment is an adaptive version of the UF-P experiment discussed above. Furthermore, since this method is adaptive, we can use the post-hoc re-evaluation UA-P^U as well. Overall we observe that the adaptation helps and the UA-P^U re-evaluation performs nearly as well as our best setup UB-P. This is corroborated by the traces of the AUC on the entire test set during the adaptation process, which are shown in Figure 4.7. At the start, the classifier is equal to the UF-P classifier. Furthermore, these classifiers

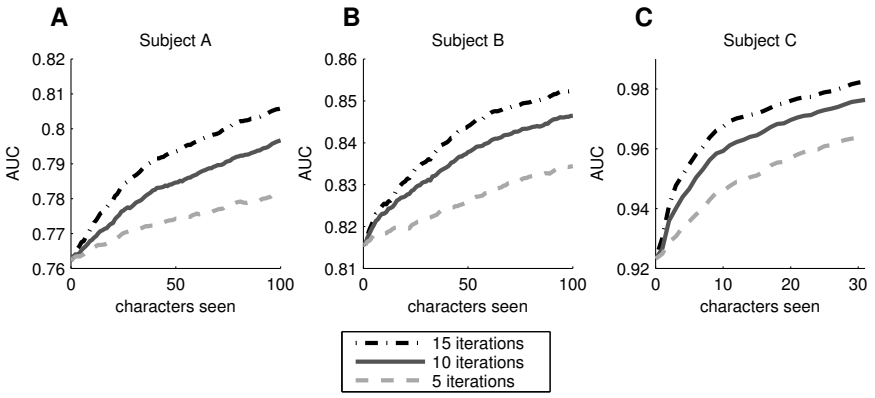


Figure 4.7: Classifier improvement trough adaptation. The initial classifier was trained unsupervisedly on the train set with 5 iterations. The classifier was adapted to the EEG by feeding it the EEG character by character and performing EM on the original training set combined with the new EEG.

are identical for 5, 10 and 15 iterations. During adaptation, we see that almost each additional trial improves performance. It is also clear that the improvement correlates with the number of trials per iteration.

This brings us to our final experiment, UA, which contains the real challenge: unsupervised learning without a pre-trained classifier. Instead, the classifier is initialised randomly and receives the EEG data one trial at a time. When we use 15 repetitions, we obtain 86.6%, 87.3% and 92.3% on the data from BCI Competition III subject A, B and BCI Competition II, respectively. This is slightly worse than the previous experiment, where we started with a pre-trained classifier. When we reduce the number of iterations per trial to 10, performance on subject A of BCI Competition III becomes poor: only 62.4 % of the trials was predicted correctly. Furthermore, when only 5 iterations are used, the method fails completely on this subject and 9.0 % of the trials is correctly classified. However, keep in mind that this is still above chance level (2.8 %). The results on subject B (53.0%) and BCI Competition II (56.5%) are significantly better but far from state of the art. The Akimpech data is less difficult and we obtain 61.2 %, 85.6 % and 93.1 % for 5, 10 and 15 iterations. The individual

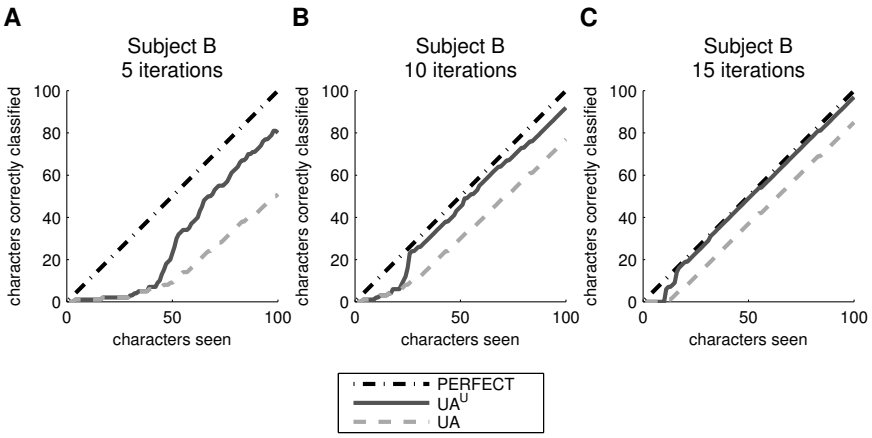


Figure 4.8: Plots showing the performance obtained by 3 single online initializations on subject B, each using a different number of repetitions to predict a character. The horizontal axis represents the number of characters processed. The vertical axis represents how many of these characters were predicted correctly. The dashed line shows us how many characters the online classifier has predicted correctly (starting with an initially untrained classifier). The solid line shows how many characters the current classifier can predict correctly if we re-test it on all of the previously processed characters. The dash-dot line represents the upper bound on the performance which equals the number of characters seen.

spelling accuracies on this dataset and 5 repetitions range from 23.1 % to 93.5%. By increasing the number of repetitions to 15, the lowest accuracy is raised to 75.3% and spelling is perfect for subject ASR. Additionally, we investigated the influence of the number of initialisations used (which is 10 in the aforementioned results). When only a single initialization is used per experiment, the performance on the Akimpech dataset drops to 57.8% for 5, 76.1% for 10 and 81.9% for 15 iterations.

We saw that when the data is really hard and the number of iterations per trial is limited, then the UA approach starts to fail. Obviously, when less (unlabelled) data is available, it becomes harder to learn. Figure 4.8, where we used data from BCI Competition III subject B, shows how the classifier's performance evolves as more and

more trials are processed. The horizontal axis represents the number of trials processed, the vertical axis denotes the number of trials predicted correctly. Obviously, we cannot predict more characters correctly than the number of characters processed. This upper bound is represented by the dash-dot line. The dashed line represents the UA classifier performance and the solid line represent UA^U which is the post-hoc re-evaluation. From this plot it is clear that the classifier suffers initially, both the UA and UA^U approach fail. After a number of characters, which depends on the subject and the number of iterations per trial, the UA classifier starts to predict symbols correctly and the post-hoc approach makes a big performance leap by correcting its mistakes. The result is a non-linear transition at this *eureka-moment*. Furthermore, from this point onwards, the classifier makes almost no additional mistakes. The UA and UA^U methods will be analysed in detail in the next section, where we present an online study comparing these methods to a state of the art supervised LDA classifier.

4.2.3 Summary

In this offline study we have shown that it is not only possible to train a model without having access to labelled information; these models perform on the same level as supervised models when there is enough (unlabelled) data available. On top of that, we have shown that these unsupervised models are also able to generalise to subsequent spelling sessions. Additionally, our results indicate that the unsupervised method can be used in an online experiment such that a calibration session is no longer required. There is, however, a warm-up period during which the classifier is unreliable but the post-hoc re-evaluation is able to correct these initial mistakes eventually.

We used data from 25 different subjects (3 different datasets) and our method performed stably across all datasets and subjects without specific tuning for the individual dataset. This indicates that the proposed method is robust and can potentially be used in a real online setting. This is definitely much more important than obtaining state of the art performance after many hours of parameter tuning and engineering.

4.3 Online evaluation

In the remainder of this chapter, we focus on the evaluation of the proposed unsupervised method in an online setting. In this online study, we have used AMUSE, which is an auditory ERP paradigm. The goal of this study is to prove that the proposed method is not only applicable in an online setting, but also that it can be applied to the more challenging auditory ERP paradigms. The evaluation will focus on a comparison between a supervised regularised LDA classifier and the UA/UA^U methods, which learn from scratch.

4.3.1 Experimental setup

Experimental paradigm: AMUSE

In the online study, we have used the spatial auditory ERP paradigm AMUSE. This paradigm was originally proposed by Schreuder et al. (2010) and has already been discussed in the first chapter (see 1.2.2). What follows is a brief overview. The paradigm uses six different tones as stimuli, each of which can be recognised by its unique pitch and the location of the speaker. These six tones/directions can be used in a two-step procedure to select one out of 36 symbols, see Figure 4.9. In the first step, the user is able to select a group of 6 symbols, in the second step the user can select a symbol from the previously selected group. In the current study, these symbols included 26 letters, German umlauts, whitespace and punctuation marks. The grouping of the symbols is shown in our illustration of the user interface.

The current setup deviated from the original AMUSE study as follows. To reduce the likelihood of front-back confusions, the user was placed 10 cm behind the centre of the ring of speakers. For the same reason, the speakers are not spaced equally on the ring, but they are shifted to the front. This enhances the spatial discrimination between the tones. Even though it is possible to use the AMUSE paradigm in a BCI which works completely in the auditory domain, we have simplified the spelling interface and we supported the user by indicating during the pre-trial cue on which tone he had to focus. However, during the actual stimulus presentation, the information on

the screen was static such that it could not influence the recorded signals.

Classification methods

The state of the art of calibration based methods is represented by a subject-specific LDA classifier with shrinkage regularisation on the sample covariance matrix (Ledoit and Wolf, 2004). This classifier is the standard option in the Berlin BCI group and is known to perform well on a wide range of ERP paradigms, including visual, auditory and tactile stimuli (Treder and Blankertz, 2010; Höhne et al., 2011; Acqualagna and Blankertz, 2013; Thurlings et al., 2012).

During the online experiment, the randomly initialised UA method and the post-hoc re-analysis UA^U were evaluated. The hyper-parameters of the unsupervised UA approach were optimised using the data from the original AMUSE study comprising 21 subjects. Details on this dataset are available in Appendix C. Overall, the results from this analysis corroborated our previous experimental findings. Furthermore, we found that performance was stable over a wide range of parameter values. In the end, we decided on using 5 classifier pairs, and 5 EM iterations per trial, β was initialised to 1 and α to 100. To prevent the classifier from collapsing on the prior, we limit α to 200. The number of iterations was not optimised but set to 15, which is a value often used used in BCI studies and matches the original AMUSE study. The number of trials during calibration is set to 30. Including more data points in the calibration would not improve supervised performance.

Data acquisition and preprocessing

To record the EEG data, we used BrainProduct BrainAmp amplifiers. The data was recorded at 1 Khz from 31 passive Ag/AgCl electrodes[2]. Additionally, both vertical and horizontal EOG was recorded but not used during decoding. The amplifiers used built-in band-pass filters with a 0.1 Hz lower and 250 Hz upper cut-off frequency. Before the

[2]Following the extended 10-20 naming scheme, these were channels `Fp2`, `F9`, `F5`, `F1`, `F2`, `F6`, `F10`, `FT7`, `FC3`, `FCz`, `FC4`, `FT8`, `C5`, `C1`, `Cz`, `C2`, `C6`, `TP7`, `CP3`, `CPz`, `CP4`, `TP8`, `P9`, `P5`, `P1`, `P2`, `P6`, `P10`, `POz`, `O1` and `O2`.

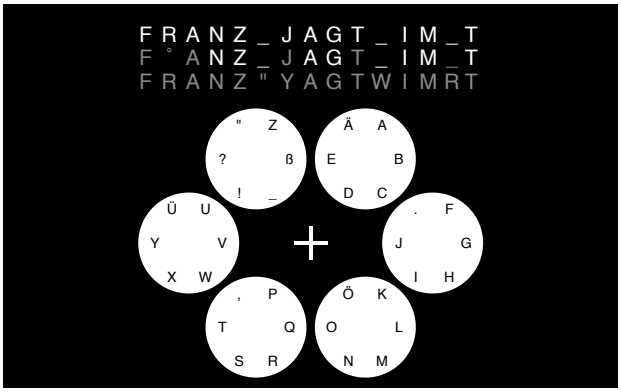


Figure 4.9: User interface used during the online study. The circles present the grouping of the symbols. The location of the circles and the symbols within the circles corresponds *nbf*.

actual pre-processing started, the signal was low-pass filtered at 45 Hz and downsampled to 100 Hz.

The entire online setup was implemented in the Berlin BCI toolbox and we used nearly identical pre-processing for both the unsupervised method and the supervised baseline. After an additional low-pass filter with a cut-off frequency of 40 Hz (Chebyshev type 2 filter of order five, stop-band attenuation of 20 dB), the data was epoched from 200 ms before the stimulus until 700 ms after stimulus presentation. The average baseline activity was estimated on the pre-stimulus interval and subtracted from the post-stimulus interval. For supervised calibration, we performed outlier removal based on a variance criterion. However, during the online experiment no outlier removal was used. To compute the features, we used twelve intervals to average the signal[3]. These intervals were concatenated and formed a 372-dimensional feature vector. For the unsupervised classification method two minor additional steps had to be included for technical reasons. First, normalization to zero mean and unit variance was applied feature-wise per trial. Second, the inclusion of a bias term was necessary.

[3]The intervals (in ms) were `[100 130]`, `[130 160]`, `[160 190]`, `[190 220]`, `[220 250]` for earlier, more transient ERP components, and `[250 300]`, `[300 350]`, `[350 400]`, `[400 450]`, `[450 500]`, `[500 600]`, `[600 700]` for later, slower components.

Participants

A total of 10 healthy subjects, which will be represented using the anonymised codes *nbb, nbc, nbd, nbe, nbf, nbg, nbh, nbi, nbj, jh*, partipated in the online study. All of them were external to the lab and were recruited using an online advertisement. For their effort, they were compensated with 8 EUR/h. All subjects stated to have normal hearing, to be non-smokers and to have no known neurological disorder or history. They also claimed not to take any psychoactive or EEG-altering substances. But this information was not systematically verified. Out of those 10 subjects, 6 subjects were male and four were female. The age ranged from 20 to 58 with an average of 34.2. All but two subjects had no prior BCI or EEG experience. Subject *nbj*, however, had taken part in an EEG experiment before, but this was not BCI related. Participant *jh* on the other hand had prior BCI experience as he had participated in a motor imagery based experiment at the TU-Berlin. Aside from these 10 subjects, a single extra subject participated in a pilot study featuring a shortened version of the experiment. This pilot study is not considered in the following analysis.

Following the Helsinki Declaration, all subjects received information on the experiment and they declared written informed consent. About one week in advance, the participants received information about the experiment, including task instructions, the request to have a good night's rest the night before, and a morning hair wash on the day of the experiment.

Course of the experiment

The entire experiment took each subject about 4 hours to complete. This includes everything from informing the participant, receiving concept, EEG cap setup, detailed instructions, the actual recording, and a hair wash. The experiment follows the structure visualised in Figure 4.10. The first three blocks are identical for all participants: familiarisation with a standard oddball experiment, a standard oddball recording and familiarisation with the concept of spatial auditory attention. After these three blocks, the participants were divided into two groups. Group A started with a supervised cali-

bration session followed by familiarisation with the copy spelling application and then alternating between unsupervised and supervised evaluation. The members of group B start with familiarisation with the copy spelling method, followed by unsupervised evaluation. Then we proceeded with the supervised calibration and afterwards they alternated between evaluation of the supervised and the unsupervised method. Both groups completed three evaluation blocks of each condition. What follows is a more detailed description of these experimental blocks.

Standard oddball familiarization and recording

After setup of the system (EEG cap, gelling, ...), the users were shown their EEG signals and they were educated about typical EEG artefacts, e.g. eye-blinks, and how to avoid them. Afterwards, we introduced the concept of an auditory oddball recording and proceed with two real recordings. These oddball recordings last five minutes during which a stream of short tones is presented. The tones are one of two pitches and originate from a single speaker. The stimulus onset asynchrony (SOA) is 1000 ms and there are on average 4 non-target tones per target tones. A total of 40 targets is presented, which they have to count silently and without motion.

Familiarization with the spatial tones

After the standard oddball recording, we introduced the spatial auditory paradigm AMUSE to the participants. To explain the paradigm, the participants heard examples of the tone sequences from the ring of loudspeakers around them. Initially, these tone sequences had a slow SOA of 1000 ms, which was reduced to 175 ms during this block. This equals the SOA used in the real BCI experiment.

Classifier calibration

The concept of spatial auditory attention was put to work in the calibration block. During this block, the subjects performed 30 trials with 15 iterations per trial. Each trial began with a cue indicating on which stimulus to focus. This cue consisted of a visual and an auditory component. The visual information was

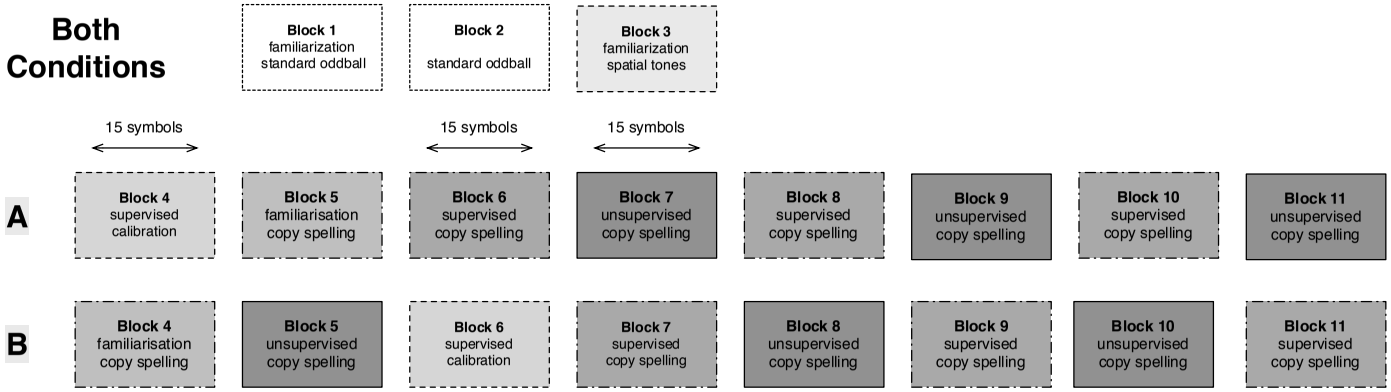


Figure 4.10: Schematic display of the course of an experimental session.

the highlighting of the desired stimulus direction on the screen. The auditory cue was three examples of the target tone. After the cue, there was a 2 second break before the stimulus sequence started. This stimulus sequence contained 15 repetitions of all 6 stimuli in pseudo random order. The stimulus duration was 40 ms and the stimulus onset asynchrony was 175 ms. Similar to the oddball recording, the subjects were requested to count the 15 target tones without making a sound or moving, while at the same time neglecting the 75 non-target tones.

Familiarization with copy spelling

The evaluation blocks introduce an additional layer of complexity compared to the calibration session. For this reason, we introduced a familiarisation block where we explained the copy spelling application. As mentioned above, the BCI is controlled using a two step process, where in the first step one of the six groups is selected by focussing on the corresponding tone. In the second step, a symbol from within this group is selected.

(Un)supervised copy spelling

After familiarisation with the copy spelling application, we started the evaluation runs. In each of the six runs either the supervised or unsupervised method was evaluated in an online setting. The participant was asked to copy spell a text of 15 symbols. Backspace functionality was not enabled and mistakes had to be ignored. Hence, to spell these symbols, 30 trials were required. To ensure fairness between the methods, the same texts were used for both methods. In the first two blocks we predefined this text (`FRANZ_JAGT_IM_T` and `AXI_QUER_DURCH_`) and in the last block the subjects were able to choose the text themselves.

A depiction of the user interface is shown in Figure 4.9. Each circle corresponds to a group of symbols in the first step of the selection process. Each symbol within a circle corresponds to the options in the second step of the selection. The position of the circles (and symbols) corresponds to the position of the speaker assigned to the stimulus in the ring. The target string, which had to be copy-spelled, is shown on the top of the screen

(first line). The results from the online decoding is shown on the second line. In this line errors were marked as follows. A single error in the two step selection process resulted in a grey letter, two errors resulted in ∘, two correct selections were shown in white. The bottom line (in grey) depicts the revised classifier output from the UA^U approach, but participants were requested to ignore this line.

4.3.2 Results and discussion

Basic Neurophysiology

We begin our analysis of the results with a visual inspection and comparison of the ERP responses in the supervised and unsupervised evaluation blocks. The averaged responses are shown in Figure 4.11. We observe that the ERP responses are similar for both conditions and that we can discern attention-related and class-discriminative difference between 100 and 200 ms post stimulus and from 250 ms post stimulus. This is a first indication that the experimental setup produced data of similar quality and difficulty in both conditions. Furthermore, it is a valid reproduction of the target and non-target ERP responses from the original AMUSE study.

Online performance

To obtain a baseline for the performance, we start by analysing the results from the supervised classifier LDA, see the top row in Figure 4.12. Please note that we consider the selection accuracy and not the spelling accuracy. Averaged over all experimental runs, which comprises 30 experimental blocks (10 users times three blocks) with 30 trials per block. Overall, we observe a very stable performance level for the supervised classifier. The pre-calibrated baseline method LDA obtains a selection accuracy of 92.1 %. The lowest accuracy recorded was 73 % and five blocks are decoded without a single selection error. In 7 out of 10 runs, the accuracy was at least 90 %. Increased fatigue was reported by a number of participants and this is reflected in a performance drop for the final supervised evaluation block. The performance loss is small but significant (paired t-test

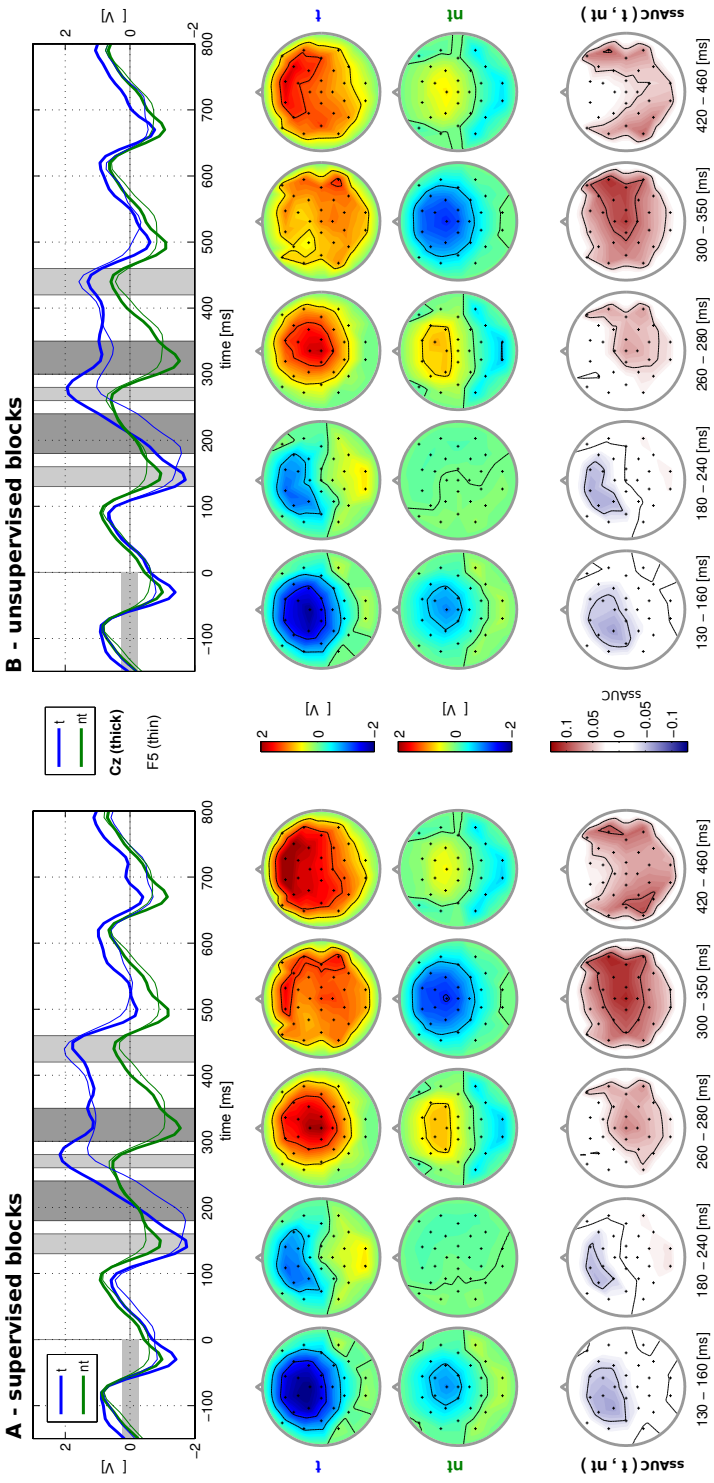


Figure 4.11: Averaged ERP responses over all 10 users for supervised (left) and unsupervised online spelling blocks (right). The top row shows the response evoked by target (blue) and non-target (green) stimuli for the channels Cz (thick) ang F5 (thin). The middle row gives the scalp plots for averaged target (t) and non-target (nt) responses at selected time intervals, which are indicated by the grey markings in the top row. The bottom row shows the univariate class-discriminative information, represented by a rescaled version of the AUC, such that 0 is equal to no discrimination, positive values correspond to a positive correlation between the class label and the feature and negative values indicate a negative correlation.

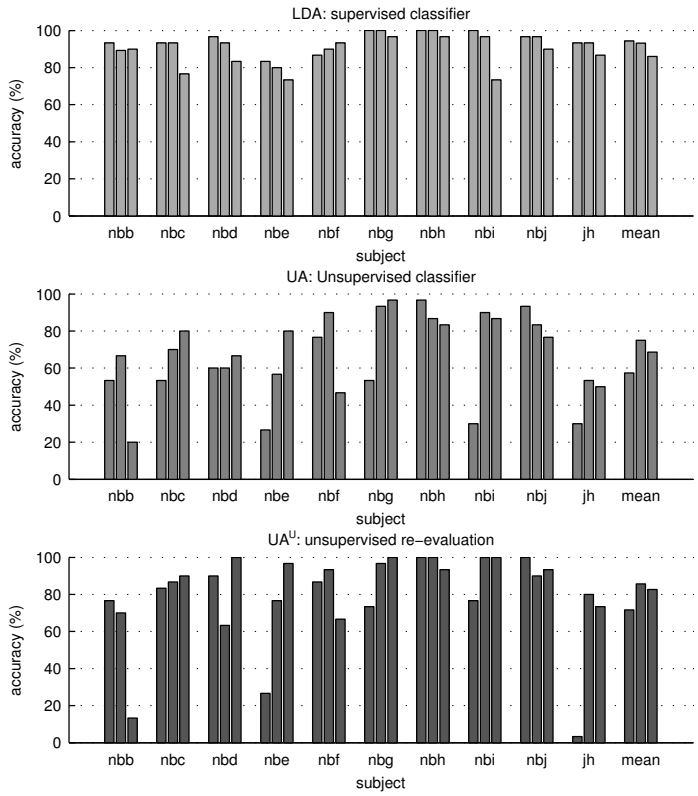


Figure 4.12: Performance comparisons between the supervised LDA model, the unsupervised UA model and the post-hoc re-evaluation UA^U. Each bar represents an experimental block of 30 trials in the online experiment. The methods are presented in the following order: top plot LDA, middle plot UA, bottom plot UA^U. Chance level is at 16.7 %.

t(9)=2.91, p=0.02). Solely based on the supervised results, we cannot determine whether the performance drop is caused by a less informative signal or by other effects of non-stationarity that cause the supervised fixed model to break down. In a subsequent analysis, we will show that the information content of the signal is good, but it is indeed the non-stationarity that troubles the supervised model.

Now, we turn to the evaluation of UA. The method was challenged by the short duration of the online evaluation blocks and UA is outperformed by LDA. Our experiments have shown that learning was successful with an accuracy of 67 % accuracy on average, which is far above chance level (1/6). The best six unsupervised blocks were completed with an online selection accuracy of no less than 90 %.

In contrast to the stable performance of LDA, the results for UA show a large amount of inter-subject and inter-block variability. Participant *jh*, for example, was not able to control the BCI in the first UA block but achieved 80% selection accuracy in the second UA block. In contrast, user *nbh* his first run was nearly perfect, with only a single selection error. For user *nbb*, the final block was not as successful and resulted in the lowest performance recorded with only 20 % of the selections correct. Similar to the supervised method, the third (last) unsupervised block has on average a decreased selection accuracy compared to the middle block, but contrary to the LDA case, this is not statistically significant t(9)=0.90, p=0.39. In addition, the average unsupervised performance is increased from the first to the second block. An effect that was statistically significant t(9)=-2.54, p=0.03, but this was not observed for the supervised method.

As we discussed before, the UA^U re-analysis uses the final updated classifier after processing all trials. The block-wise selection accuracy for UA^U is shown in Figure 4.12. This re-analysis results in a remarkable improvement of 13 % over the UA method to 80 %. Moreover, 90 % of the trials was decoded correctly by UA^U in half of the experimental runs. Finally, during seven out of 30 blocks, the UA^U method was able to obtain an error-free decoding. Unfortunately, for the three blocks that did not result in control in the UA setting, the UA^U re-analysis failed too. Later, we will show in a simulated online experiment of extended duration that this was caused by the warm-up effect we discussed in the offline study. This warm-up effect is caused

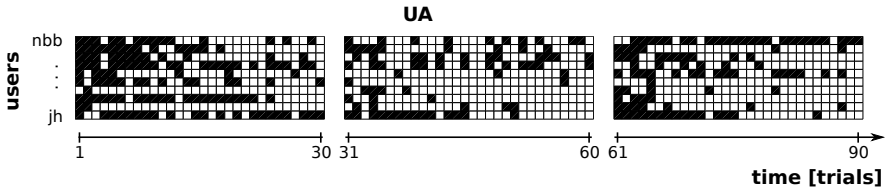


Figure 4.13: Overview of the individual trials in unsupervised evaluation blocks. Time goes from left to right and each row corresponds to a user. The order of these users equals that of Figure 4.12, with *nbb* in the top row and *jh* in the bottom row. Black squares mark selection errors, white squares indicate a successful trial. From this picture it is clear that the majority of mistakes made by UA appear at the beginning of an experimental block. Furthermore, nearly all users were able to control the BCI by the end of the block.

by the random initialisation. Consequently UA suffers at the beginning of an experimental block (Figure 4.13), however, UA^U is able to correct most of the mistakes by the end of the run (Figure 4.14).

The length of the warm-up period limits the usability of UA. To find out how long a user takes before he obtains control over the BCI. We define that the user has taken control over the BCI when three consecutive selections/trials are decoded without mistake. The probability to do so by guessing is only $\frac{1}{6^3} = 0.0047$. So, the exact point in time where the user has control is the first of these three error-free trials. Using this definition, all but three experimental runs resulted in control. The runs where control was not obtained were *nbe* and *jh* in the first unsupervised block and the run of *nbb* during the third unsupervised block. For the other runs, the average number of trials necessary to achieve control was 8.9. Two runs resulted in control in the very first trial. Furthermore, in 50 %, 70 % and 90 % of all runs, the users were able to control the BCI within 5, 15 and 25 trials respectively.

For two of these three cases where no control was possible, we received specific comments by the users. User *nbe* reported after the first online block that during this block, she could not ignore one very salient tone (front-left). Later she communicated that this problem did not persist and that she found a better strategy to focus on the target tones. User *jh* reported that during his first (unsupervised)

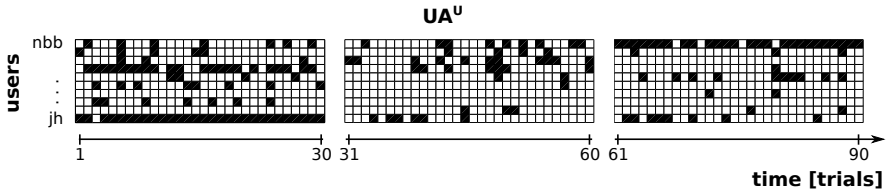


Figure 4.14: Overview of the selection errors committed during the post-hoc re-evaluation by UA^U at the end of an experimental block. These results are obtained on the same data as Figure 4.13. Bar three experimental runs, the UA^U re-evaluation is able to correct the initial mistakes and can decode the data with high accuracy. However, during the first blocks of users *nbe* and *jh*, and the third block of user *nbb*, the unsupervised method was not able to select a good classifier.

online spelling block he had trouble ignoring one very salient tone (front-right). This problem disappeared during the following blocks. We did not receive specific information from user *nbb* that could shed a light on the performance breakdown in his final unsupervised block.

We will show, using a simulated experiment, that the performance breakdown is not caused by an uninformative signal. Both the unsupervised and the supervised methods are able to decode these blocks reliably. Therefore we hypothesise that in these runs without control, the signal to noise ratio might be slightly lower. As a result, the warm-up effect was amplified. This is similar to our findings in the offline study on visual ERP data. The less data that is available, the harder it is to learn a reliable model without label information. Similarly, the more difficult the signal is to decode, the longer the warm-up period becomes.

Online text entry

Now, we will turn to the spelling accuracy, which is less forgiving than the selection accuracy since two subsequent selections must be correct in order to spell a symbol correctly. As a result, when using UA, only 7.8 out of 15 (52 %) symbols are spelled correctly on average. Not surprisingly, UA^U, the updated classifier after processing all data, is able to improve performance. This leads to an average of 10.4 out of 15 symbols (69 %) spelled correctly per block. As a comparison,

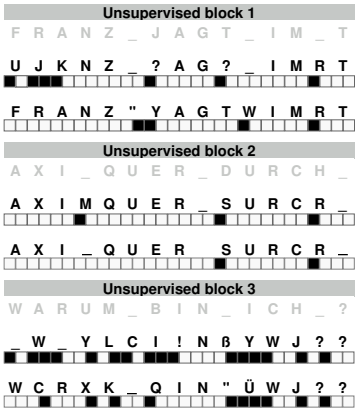


Figure 4.15: Spelling results for subject *nbf* during the unsupervised blocks. Per block, the top line represents the desired text, the middle line displays text produced online by the unsupervised classification. Text predicted by the post-hoc reanalysis at the end of the block is shown at the bottom line. Two trials are needed to determine a symbol. Individual selection errors (wrong trials) of both methods are marked by black squares directly below each symbol. Please note that the classifier was re-initialized randomly at the beginning of each block.

the pre-trained supervised classifier manages to spell 12.9 out of 15 symbols (86 %) without error.

To give the reader a feeling of the spelling quality of the unsupervised approach, Figure 4.15 presents the texts spelled (in German) by an average-performing subject *nbf* during the three unsupervised blocks. It is clear that even with a selection accuracy of nearly 80 % in the first block, it is difficult for a human observer to make sense of the spelled text. A selection accuracy of around 90 % in the second block results in better readability. In the first two blocks, the post-hoc classifier was able to revise a substantial number of symbols which had been predicted erroneously during the course of the experiment. Even though it introduced new errors, the number of wrongly decoded symbols is reduced by 40 %, from ten to six.

To judge the value of the three methods we should not be restricted to the spelling accuracy on short blocks, especially since UA is reset at the beginning of each block. The invested amount of time is an important factor, especially for patients. At the moment of the post-

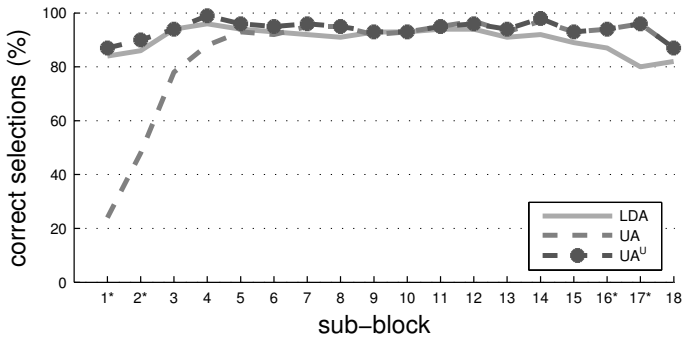


Figure 4.16: Comparison between the supervised LDA classifier, the unsupervised approach UA and the re-evaluation UA^U after processing all data. The results are presented per sub-block of 10 trials (5 symbols). The selection accuracy is averaged over all 10 users and the results are given per sub-block of 10 trials (5 symbols). Statistical significant differences (paired t-test,$p<0.05$) between the supervised LDA method and the unsupervised performance UA approachare indicated by asterisks.

hoc re-analysis (e.g. at the end of an unsupervised block), a user has spent the same amount of time interacting with the BCI as if he would have performed one full calibration run. While the calibration recording cannot result in any usable text output, the unsupervised block can. On average, it allows a user to communicate straight away with 2/3 of the symbols decoded correctly. We are aware, that this rate is not yet high enough to communicate in practical situations. On the other hand, the remedy is simple: as we will show later on in a simulated time-extended experiment, most of the errors can be corrected by post-hoc if the spelling duration is prolonged.

Simulated online experiment of extended duration

The online experiment allowed us to investigate the warm-up and learning speed of the UA method. However, the online spelling accuracy is not yet sufficient to be used for communication after these short 30 trial blocks. On the other hand, the goal of the BCI is to use it in longer sessions. Evaluating this in the online study was not possible, as the current experiments already lasted 4 hours. For this reason, we emulated a long spelling session by concatenating the EEG

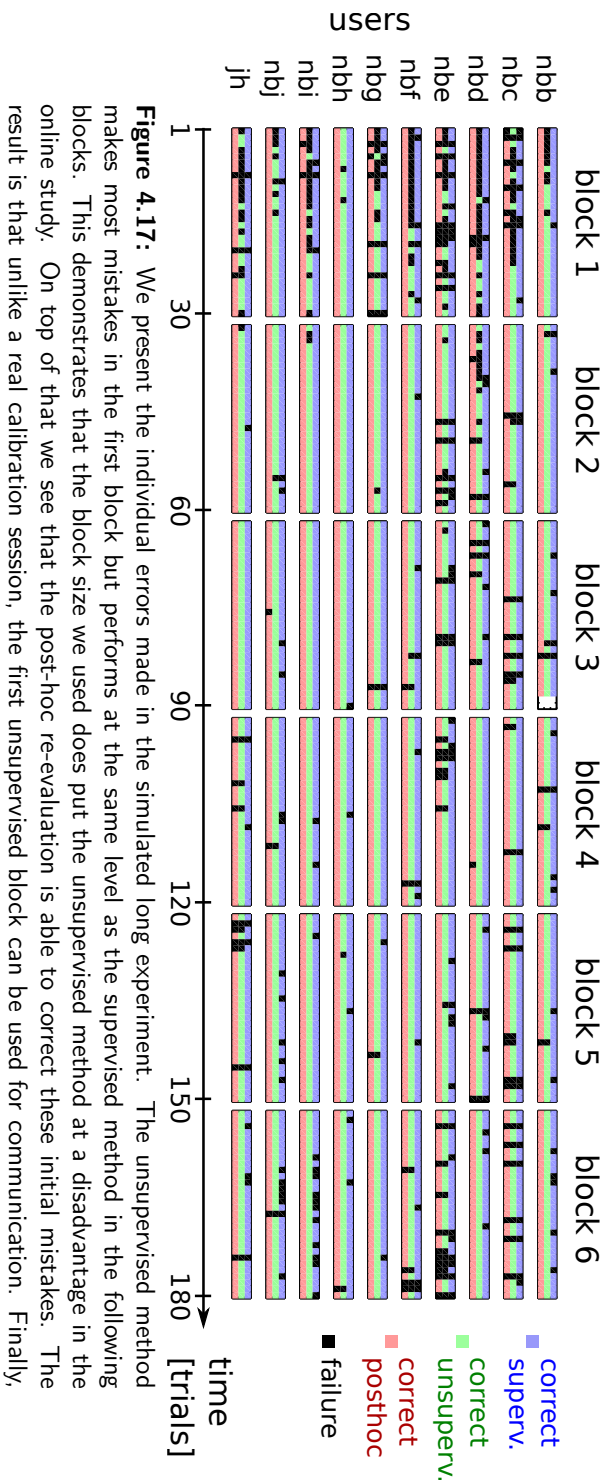


Figure 4.17: We present the individual errors made in the simulated long experiment. The unsupervised method makes most mistakes in the first block but performs at the same level as the supervised method in the following blocks. This demonstrates that the block size we used does put the unsupervised method at a disadvantage in the online study. On top of that we see that the post-hoc re-evaluation is able to correct these initial mistakes. The result is that unlike a real calibration session, the first unsupervised block can be used for communication. Finally, during the last block, the unsupervised methods perform slightly better than the supervised one.

from all experimental blocks in chronological order. This allows us to compare the performance of UA to that of LDA, when we go beyond the limitation of using only 30 consecutive trials. On top of that, it also allows us to investigate the effects of non-stationarity, e.g. due to fatigue, in a more elaborate manner. Furthermore, the UA^U evaluation will have seen the data from all 6 blocks, i.e. 180 trials. This enables us to assess how good a model can be built on this challenging auditory ERP data.

The average result, computed over all subjects on a sub-block level (10 trials per sub-block) is shown in Figure 4.16. In the first three sub-blocks, UA is outperformed by LDA, which confirms our findings from the true online study. However, the difference in performance is only significant for the first two sub-blocks (paired t-test, p=0.05). The updated classifier UA^U on the other hand performs as well as the LDA classifier. After the first two sub-blocks and up to the last two sub-blocks, performance for all three methods LDA, UA and UA^U is nearly identical. During the last three sub-blocks, effects of non-stationarity have reduced the performance of the fixed LDA classifier but not for adaptive methods UA and UA^U. A paired t-test (p=0.05) indicates that the difference is only statistically significant for the 16th and 17th sub-block, not for the last sub-block. When we combine this observation with the fact that UA^U performs stably over the entire dataset, we can conclude that the drop in performance for the supervised method is not caused by a less informative signal, but by shifts in the data distribution that are not present in the calibration data. This gives us additional support for the use of adaptive methods in longer experiments.

Before we form our conclusions, we will analyse the individual errors in this simulated experiment in more detail, on the level of the individual subjects. The individual errors are shown in Figure 4.17. Overall, we see that in this simulation 49 % of the mistakes made by LDA were made by UA^U, as well. In the opposite direction we observe that 55 % of the UA^U mistakes were committed by LDA too. This provides support for the intuition that there are trials which are "objectively" difficult, but we cannot make hard claims about this. Our global analysis has shown that averaged over the subjects, the methods perform comparably from the second block onwards. Now

we observe that this holds for all subjects individually, even for those blocks where the unsupervised methods failed in the online study. The blocks were the first block for users *nbe* and *jh* and the sixth block for user *nbb*. Remarkably, in this extended experiment UA^U makes only a few mistakes on these blocks. This is a very clear indication that the warm-up effect limited the performance in the online study and that there was not a reduced signal to noise ratio. This corroborates our main conclusion from the offline study. It is indeed possible to train a reliable classifier without label information. Furthermore, such a classifier can outperform state of the art supervised methods when non-stationarity is present in the data. However, a substantial amount of data (approximately 30 trials) is still required to build a good decoding model.

4.4 Conclusion

In this chapter, we have proposed an unsupervised method for ERP based BCI, which to the best of our knowledge is the first fully unsupervised method for this type of BCI. We have evaluated the proposed method extensively in offline experiments using visual ERP data from three different datasets (BCI Competition II, BCI Competition III and Akimpech), comprising 25 subjects in total. On top of that, we have presented the findings from an online study on 10 healthy subjects using the auditory ERP paradigm AMUSE. During this online study, we have compared our model to a supervised regularised LDA classifier.

Our results indicate that, when enough unlabelled data is available, the proposed unsupervised model can compete with state of the art supervised models in terms of accuracy. However, when used in an online setting, the proposed method exhibits a warm-up period during which the classifier is initially unreliable. This warm up period is caused by the lack of data, thus making it difficult to learn a reliable model. However, thanks to the post-hoc updated predictions, the mistakes made during warm-up can be corrected afterwards. This allows the user to effectively communicate during the warm up period, which is not possible during a real calibration session.

We would like to stress that the proposed model has performed

reliably across a wide array of data-sets and paradigms, even when an Emotiv EPOC[4] was used in a demonstration at NIPS2012, which demonstrates the robustness and practical applicability of the method.

Nevertheless, the warm-up period must be addressed in order to build a truly usable and user friendly BCI. In the following chapter, we will discuss how information can be shared across subjects to achieve this.

[4]A consumer grade EEG recording device with a very bad signal to noise ratio

5
A true zero-training BCI based on transfer learning

In the previous chapter, we have alleviated the dependency by proposing an unsupervised approach to ERP based decoding. This approach does not require labelled data. However, it does require a substantial amount of data before it is able to learn a high quality model. This limitation manifests itself in the form of a warm-up period at the beginning of an online experiment. During this warm-up period, the classifier is unreliable, which should come as no surprise given the random initialisation. Even though the classifier can correct the mistakes it made during the warm-up period, it prevents the basic unsupervised model from being a highly accurate zero-training approach. Hence, to transform the unified model into a true zero-training model, we have to eliminate this warm-up period.

We will solve the warm-up problem through the use of transfer learning. More specifically, inter-subject transfer learning. The concept of transfer learning in the field of machine learning denotes the sharing of knowledge between related tasks (Pan and Yang, 2010).

In BCI, each user represents a task, and the goal is to share knowledge on how to decode the brain-signals between the different users. This stands in stark contrast with the common approach of training subject-specific decoders within the BCI community. However, recently, there has been a strong increase in interest in transfer learning for ERP based decoding (Colwell et al., 2013; Herweg et al., 2013; Jin et al., 2012). Nevertheless, our approach is the first one that can work without using a single labelled example (Kindermans et al., 2012a, 2014).

5.1 Building a zero-training model

Recall that the probabilistic model we used in the previous two chapters is based on three simple assumptions. The model describes an ERP paradigm where per trial a single type of stimulus elicits a target response and all other stimuli elicit a non-target response. Furthermore, we assume that the EEG data can be projected into one dimension, where it will be Gaussian with class-conditional mean and shared variance. In the most basic form of this model, we make no assumptions about the attended stimuli and we place a zero mean prior on the weight vector to regularise the model. To keep the notation uncluttered, we have dropped the subscript that identifies the individual iterations and we will indicate the stimuli within a trial with i. Furthermore, to make the sharing of information between subjects more explicit, we have introduced the subject-specific identifier s. This minor modification results in the following description of the probabilistic model.

$$\begin{aligned} p\left(\boldsymbol{w}_s\right) &= \mathcal{N}\left(\boldsymbol{w}_s|0, \alpha_s^{-1} I\right) \\ p\left(c_{s,t}, \ldots, c_{t-1}\right) &= \frac{1}{C} \\ p\left(\boldsymbol{x}_{s,t,i}|c_{s,t}, \boldsymbol{w}_s, \beta_s\right) &= \mathcal{N}\left(\boldsymbol{x}_{s,t,i}^T \boldsymbol{w}_s | y_{s,t,i}(c_{s,t}), \beta_s^{-1}\right) \end{aligned}$$

Here $\boldsymbol{x}_{s,t,i}$ is the $N \times 1$ dimensional[1] feature vector of subject s corresponding to stimulus i during trial t; $X_s = [\boldsymbol{x}_{s,1,1}, \ldots, \boldsymbol{x}_{s,T,I}]$ contains all feature vectors of subject s. The attended stimulus/symbol during trial t is $c_{s,t}$ and there are C different stimuli. The function $y_{s,t,i}\left(c_{s,t}\right)$ encodes the class-conditional target mean, $y_{s,t,i}\left(c_{s,t}\right) = 1$ if $c_{s,t}$ is contained in stimulus i during trial t, otherwise $y_{s,t,i}\left(c_{s,t}\right) = -1$. The vector $\boldsymbol{y}_s\left(\boldsymbol{c}_s\right) = \left[y_{s,1,1}\left(c_{s,1}\right), \ldots, y_{s,T,I}\left(c_{s,T}\right)\right]^T$ comprises all target means for a single subject. The weight vector used to project the features into a single dimension is $\boldsymbol{w}_s$, the shared prior mean on this weight vector is $\boldsymbol{\mu}_{\boldsymbol{w}}$. The precision of the prior is subject-specific and represented by α_s. The per-class variance of the projected ERP features is β_s^{-1}.

[1] This includes the bias term.

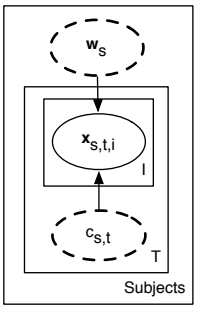


Figure 5.1: Graphical representation of the probabilistic model used for unsupervised training. The variables in the dashed ellipses are the points where we can introduce extra prior knowledge.

When we look at the graphical model representation (Figure 5.1), we see that there are two nodes where additional prior knowledge can be introduced. Similar to the first chapter, we can embed language statistics into this model. A different option is to introduce prior knowledge about the weight vector of the classifier. In our model, this prior knowledge will come from other subjects; we will use inter-subject transfer learning.

5.1.1 Transfer learning

Our approach to transfer learning assumes that the weight vectors are similar across subjects. This may seem rather odd because there is typically a huge amount of variability in the recorded signals between subjects. Furthermore, the BCI community has a strong tradition of using subject-specific models. We argue that the inter-subject differences are actually rather superficial. Moreover, we believe that the information contained in the brain signals is structured similarly for all subjects. We assume that the information resides in a low dimensional subspace of the original feature space, and that this subspace can be shared over subjects. For a detailed discussion on this hypothesis we refer to the next chapter. Furthermore, please note that this is specific to ERP data and conceptually different from the approach of Samek et al. (2013) for motor imagery. In their work, the authors

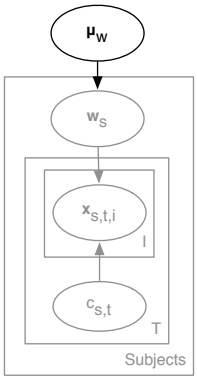


Figure 5.2: Graphical representation of the probabilistic model that incorporates transfer learning. The original graphical model is drawn in grey, the transfer learning component is drawn in black.

locate the uninformative subspace of the signal that is shared across users and remove.

To introduce inter-subject transfer learning into the model, we place a hyper-prior on the subject-specific distribution of the weight vector. The resulting graphical model is shown in Figure 5.2. The formal definition of the model, where the original model is in grey and the modifications are highlighted in black, is as follows

$$\begin{aligned} p\left(\boldsymbol{\mu}_{\boldsymbol{w}}\right) &= \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}|0, \alpha_p^{-1} I\right), \\ \alpha_p &= 0, \\ p\left(\boldsymbol{w}_s|\boldsymbol{\mu}_{\boldsymbol{w}}\right) &= \mathcal{N}\left(\boldsymbol{w}_s|\boldsymbol{\mu}_{\boldsymbol{w}}, \alpha_s^{-1} I\right), \\ p\left(c_{s,t}, \ldots, c_{t-1}\right) &= \frac{1}{C}, \\ p\left(\boldsymbol{x}_{s,t,i}|c_{s,t}, \boldsymbol{w}_s, \beta_s\right) &= \mathcal{N}\left(\boldsymbol{x}_{s,t,i}^T \boldsymbol{w}_s|y_{s,t,i}(c_{s,t}), \beta_s^{-1}\right). \end{aligned}$$

In this model, the subject-specific prior distributions on the weight vectors have no longer a zero mean but a mean equal to $\boldsymbol{\mu}_{\boldsymbol{w}}$. The hyper-prior on $\boldsymbol{\mu}_{\boldsymbol{w}}$ has zero mean and precision α_p. This results in an infinitely wide covariance matrix. The effect of this hyper-prior is that it couples the classifier for all subjects, while at the same time

allowing for enough flexibility to build good subject-specific models.

During inference, we assume that $\boldsymbol{w}_s$ is known. As a consequence, introducing transfer learning has no influence on the inference strategy. The transfer learning approach only influences the training procedure, and more specifically, the maximisation step for $\boldsymbol{w}_s$ and the update for α_s. Analogous to the basic model, training consists of a combination of the EM algorithm and maximum likelihood optimisation. During training we expect that $\boldsymbol{\mu}_{\boldsymbol{w}}$ is pre-computed. The derivation of the transfer learning update equations are rather similar to those of the basic unsupervised model, see Appendix A. For this reason, we present the original update equations in grey and highlighted the changes in black.

$$\hat{\boldsymbol{w}}_s = \sum_{\boldsymbol{c}_s} p\left(\boldsymbol{c}_s | X_s, \boldsymbol{w}_s, \beta_s\right) \left(X_s X_s^T + \frac{\alpha_s}{\beta_s} I\right)^{-1} \left(X_s \boldsymbol{y}_s\left(\boldsymbol{c}_s\right) + \frac{\alpha_s}{\beta_s} I \boldsymbol{\mu}_w\right)$$

$$\hat{\beta}_s{}^{-1} = \left\langle \sum_{c_{s,t}} p\left(c_{s,t} | X_s, \boldsymbol{w}_s, \beta_s\right) \left(\boldsymbol{x}_{s,t,i}^T \boldsymbol{w}_s - y_{s,t,i}(c_{s,t})\right)^2 \right\rangle_{t,i}$$

$$\hat{\alpha_s} = \frac{D}{\left(\boldsymbol{w}_s - \boldsymbol{\mu}_w\right)^T \left(\boldsymbol{w}_s - \boldsymbol{\mu}_w\right)}$$

Similar to the original model, the update for $\boldsymbol{w}_s$ consists of computing a weighted average over all possible regression classifiers. The weight per classifier is equal to the probability that the labels used to train the classifier were correct according to our previous estimate of $\boldsymbol{w}_s, \beta_s$. The key difference is that instead of regularising the classifier towards $\mathbf{0}$, we regularise the classifier towards the prior mean. The update for β_s^{-1} has not changed, and is equal to the expected mean squared error between the projection target and the actual projection. Finally, the updated α_s is simply the average squared difference between the prior weight and the classifier weight. From these update equations, it is clear that there is no computational penalty associated with using the transfer learning approach in an online setting. Actually, the opposite is true, since the transfer learning approach results in a reliable initialisation, we only have to update a single classifier in lieu of the multiple classifiers pairs that we used previously.

However, we do have to compute the estimate for $\boldsymbol{\mu}_w$ beforehand.

We begin by training the model without transfer learning on the data of a set of pre-recorded subjects $s = 1, \ldots, S$, where we set $\alpha_s = \alpha_p = 0$ and $\boldsymbol{\mu}_w = 0$. This is essentially the basic unsupervised approach. After we have trained on all of these subjects, we have S subject-specific Maximum A Posteriori estimates for $\boldsymbol{w}_s^{new}$, and also an optimised value for α_s^{new} per subject. These can be used to compute the general model, which is the mean and precision of the posterior distribution on $\boldsymbol{\mu}_{\boldsymbol{w}}$ given our estimates for $\alpha_s, \boldsymbol{w}_s$:

$$p\left(\boldsymbol{\mu}_w | \boldsymbol{w}_1^{new}, \ldots, \boldsymbol{w}_s^{new}\right) = \mathcal{N}\left(\boldsymbol{\mu}_w | \boldsymbol{\mu}_p^{new}, (\alpha_p^{new})^{-1} I\right),$$

$$\boldsymbol{\mu}_p^{new} = \frac{1}{\alpha_p^{new}} \sum_{s=1\ldots S} \alpha_s^{new} \boldsymbol{w}_s^{new}, \quad \alpha_p^{new} = \sum_{s=1\ldots S} \alpha_s^{new}.$$

Recall that in linear classifiers, the average absolute value of the weights corresponds to the complexity of the model. Furthermore, in our original optimisation, a model with low complexity will have a high precision on the prior. A model with high complexity will have a low precision. Hence, from the computation of the posterior we see that the models with low complexity and low weights will contribute more to the general model due to their high precision. As a result, highly complex models with large weights cannot dominate the general model.

The mean of the posterior will be used when we apply transfer learning to a new subject $S+1$. Here, we set $\boldsymbol{\mu}_w$ to $\boldsymbol{\mu}_p^{new}$ and keep it fixed. The precision of the subject precision prior α_{S+1} is initialised with α_p^{new}, and this value is also optimised during the online experiment. By optimising α_{S+1} we can switch between staying close to the prior when the data is difficult to fit or to build a very specific model when we find a good fit to the data (α_{S+1} becomes very small in this case).

5.1.2 Language models

Our second source of prior knowledge is language statistics. Language models were already used in Chapter 3 and will be re-introduced now. This results in a minor change of the model, which is depicted in

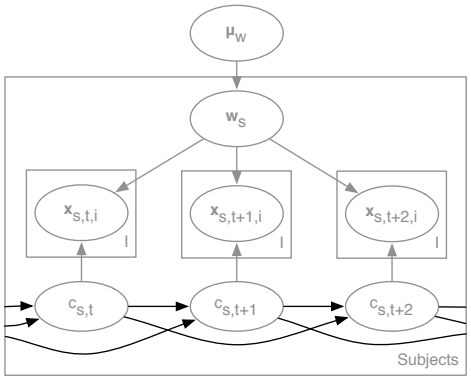


Figure 5.3: Graphical representation of the probabilistic model used for unsupervised training in combination with language models and transfer learning. The basic transfer model is shown in grey, the language model extension is highlighted in black.

Figure 5.3 and formalised below. Similar to the previous modification, we have highlighted the change in black.

$$
\begin{aligned}
p\left(\boldsymbol{\mu}_{\boldsymbol{w}}\right) &= \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}} | 0, \alpha_p^{-1} I\right) \\
\alpha_p &= 0, \\
p\left(\boldsymbol{w}_s | \boldsymbol{\mu}_{\boldsymbol{w}}\right) &= \mathcal{N}\left(\boldsymbol{w}_s | \boldsymbol{\mu}_{\boldsymbol{w}}, \alpha_s^{-1} I\right) \\
p\left(c_{s,t} | c_{t-n+1}, \ldots, c_{t-1}\right) &= \mathit{Multinomial}\left(\boldsymbol{\kappa}\left(c_{t-n+1}, \ldots, c_{t-1}\right)\right) \\
p\left(\boldsymbol{x}_{s,t,i} | c_{s,t}, \boldsymbol{w}_s, \beta_s\right) &= \mathcal{N}\left(\boldsymbol{x}_{s,t,i}^T \boldsymbol{w}_s | y_{s,t,i}(c_{s,t}), \beta_s^{-1}\right)
\end{aligned}
$$

In this model, $\mathit{Multinomial}\left(\boldsymbol{\kappa}\left(c_{t-n+1}, \ldots, c_{t-1}\right)\right)$ describes the probabilities of the stimuli based on the previously spelled text. This change to the model only influences the inference mechanism, which is used in the expectation step of our optimisation procedure. The maximisation step, on the other hand, does not have to be modified when language models are introduced. Furthermore, if the language model is kept simple, i.e. without a history: $p\left(c_{s,t} | c_{t-n+1}, \ldots, c_{t-1}\right) = p\left(c_{s,t}\right)$, computing the likelihood of a symbol given the EEG data re-

mains a direct application of Bayes's rule:

$$\hat{c}_{s,t} = \arg\max_{c_{s,t}} p\left(c_{s,t}|X_{s,t}, \boldsymbol{w}_s, \beta_s\right) = \frac{p\left(c_{s,t}\right) p\left(X_{s,t}|c_{s,t}, \boldsymbol{w}_s, \beta_s\right)}{\sum_{c_{s,t}} p\left(c_{s,t}\right) p\left(X_{s,t}|c_{s,t}, \boldsymbol{w}_s, \beta_s\right)}.$$

This still holds when the language models do utilise history but we assume that the previous symbol is either correct or has to be erased by a backspace action. This was the approach we took in Chapter 3. However, using that approach, we cannot use the post-hoc re-evaluation. Furthermore, the power of the language models is not fully utilised in that setting. Language models perform only at their full potential when they can use information from both subsequent and preceding trials. This is when uncertainty with respect to the previous symbols is allowed. But introducing this uncertainty in the language models makes inference much more involved, and we have to rely on the forward backward algorithm (Bishop, 2007) to perform inference. We will briefly summarize this method. To keep the notation uncluttered, we drop the subscript s and omit the conditioning on $\boldsymbol{w}_s, \beta_s$. The likelihood of a symbol in trial t given the data from all trials X can be written as follows.

$$\begin{aligned} p(c_t|\boldsymbol{X}) &= \sum_{c_{t-1},\ldots,c_{t-n+2}} \frac{p\left(X_1, \ldots, X_T, c_t, \ldots, c_{t-n+2}\right)}{p\left(X\right)} \\ &= \sum_{c_{t-1},\ldots,c_{t-n+2}} \frac{p\left(X_1, \ldots, X_T, c_t, \ldots, c_{t-n+2}\right)}{\sum_{c_t,\ldots,c_{t-n+2}} p\left(X_1, \ldots, X_T, c_t, \ldots, c_{t-n+2}\right)} \end{aligned}$$

In both parts of the fraction $p\left(X_1, \ldots, X_T, c_t, \ldots, c_{t-n+2}\right)$ appears, which we can decompose into a forward and backward component.

$$\begin{aligned} p\left(X_1, \ldots, X_T, c_t, \ldots, c_{t-n+2}\right) &= f\left(c_t, \ldots, c_{t-n+2}\right) b\left(c_t, \ldots, c_{t-n+2}\right) \\ f\left(c_t, \ldots, c_{t-n+2}\right) &= p\left(X_1, \ldots, X_t, c_t, \ldots, c_{t-n+2}\right) \\ b\left(c_t, \ldots, c_{t-n+2}\right) &= p\left(X_{t+1}, \ldots, X_T|c_t, \ldots, c_{t-n+2}\right) \end{aligned}$$

Both the forward and the backward component have to be computed recursively.

$$f\left(c_t, \ldots, c_{t-n+2}\right) = p\left(X_t | c_t\right) \sum_{c_{t-n+1}} p\left(c_t | c_{t-1}, \ldots, c_{t-n+1}\right) f\left(c_{t-1}, \ldots, c_{t-n+1}\right)$$

$$b\left(c_t, \ldots, c_{t-n+2}\right) = \sum_{c_{t+1}} p\left(X_{t+1} | c_{t+1}\right) p\left(c_{t+1} | c_t, \ldots, c_{t-n+2}\right) b\left(c_{t+1}, \ldots, c_{t-n+3}\right)$$

To initialise the forward recursion we use $p\left(X_1, c_1\right)$, which can be computed easily. The backward recursion is initialised with 1. A final important observation is that in an online setting, inference for the last trial can be done very efficiently. Indeed, when we cache the values from the previous forward recursion, we only have to perform an additional step of this recursion to compute the marginal likelihood of the symbol in the last trial.

5.1.3 Dynamic stopping

Transfer learning enables the classifier to work reliably from the very first trial. As a consequence, dynamic stopping can be re-introduced. The dynamic stopping strategy is identical to the one introduced in Chapter 3. When dynamic stopping is used, the stimulus presentation for the current trial is terminated as soon as the classifier assigns 99 % of all probability mass to a single symbol. We opted for 99 % as it is a conservative threshold with a clear interpretation, i.e. the classifier is very confident. Furthermore, we would like to stress that in a completely unsupervised setting, which the current setup is, there is no possibility to optimise the dynamic stopping threshold.

5.2 Offline evaluation

Analogous to the unsupervised study, we will begin with an extensive offline evaluation on visual ERP data to find out the capabilities and limitations of the proposed model. Afterwards we will present the results from the online study on auditory ERP data.

5.2.1 Experimental setup

Data

Once more, we will use the Akimpech dataset for our experiments. The BCI Competition datasets are not considered in this chapter because they encompass only three subjects. The Akimpech dataset, in contrast, comprises visual ERP data from 22 different subjects that were using the standard 6x6 matrix speller. The length of the test-data differs per subject. The average is 22.18 trials, with a minimum of 17 and a maximum of 29. For more details on this dataset please refer to Appendix C.

Analogous to the evaluation of language model based decoding of our unsupervised model, we will re-use the dataset augmentation approach presented in Section 3.2.1. The reason is that the target texts in the Akimpech dataset are limited to a small number of Spanish words. We have shown previously that the combination of a language model and a specific text can influence the end result drastically. The dataset augmentation procedure generates an extended dataset in which data from each subject is used in combination with 20 different texts, which are random samples from a Wikipedia dataset (Sutskever et al., 2011).

Language model

The language models, which are uni-, bi- and tri-gram models, are computed on the first $5 \cdot 10^8$ characters from the Wikipedia dataset. Note that this part of the dataset was not used to sample the evaluation texts. In the following experiments, we did not go beyond tri-gram models since inference scales badly with the length of the language models. To regularise the language model, we have used Witten-Bell smoothing (Chen and Goodman, 1999).

Pre-processing

We did not modify the pre-processing approach compared to our previous offline evaluation. To mimic online experiments, we pre-process the data trial by trial. Per trial, we apply a Common Average Refer-

ence filter and a bandpass filter (0.5 - 15 Hz). Then, we proceed with sub-sampling by a factor 6 and retain 10 samples centred at 300 ms post stimulus. Before feeding the data to the classifier, we append a bias term.

Experiments

SF: Supervised-fixed
The SF model is our basic unified probabilistic model from Chapter 3. It is trained on data from 16 trials with 15 iterations per trial, and 12 stimuli per iteration.

UA: Unsupervised adaptation
The UA model, which we introduced in the previous chapter, is the unsupervised model that is initialised at random at the beginning of an experiment. The EEG is processed trial per trial and when all data from a single trial is collected, we update the classifier. The current UA setup used 5 classifier pairs in parallel and 3 EM updates per trial. After the EM updates are executed, we select the best classifier based on the log likelihood. This classifier is used to predict the attended symbol. Before we move on to the next trial, we execute our re-initialisation procedure, where within each pair, the classifier with the lowest log likelihood is re-initialised with the weight vector from the classifier with the highest log likelihood (within that pair). The entire approach is visualised in Figure 5.5.

TF: Transfer-fixed
Our transfer learning model uses a leave-one-subject-out training. We began by training offline, without using label information, on all 21 subjects individually. These subject-specific classifiers are combined in a general subject-unspecific model. This model is then used without adaptation for a novel subject. We have opted for unsupervised training of our general model, as it results in the most flexible setup. Indeed, data from true free-spelling sessions, in which the ground truth is unknown, can be used to build the general model.

TA: Transfer-adaptive

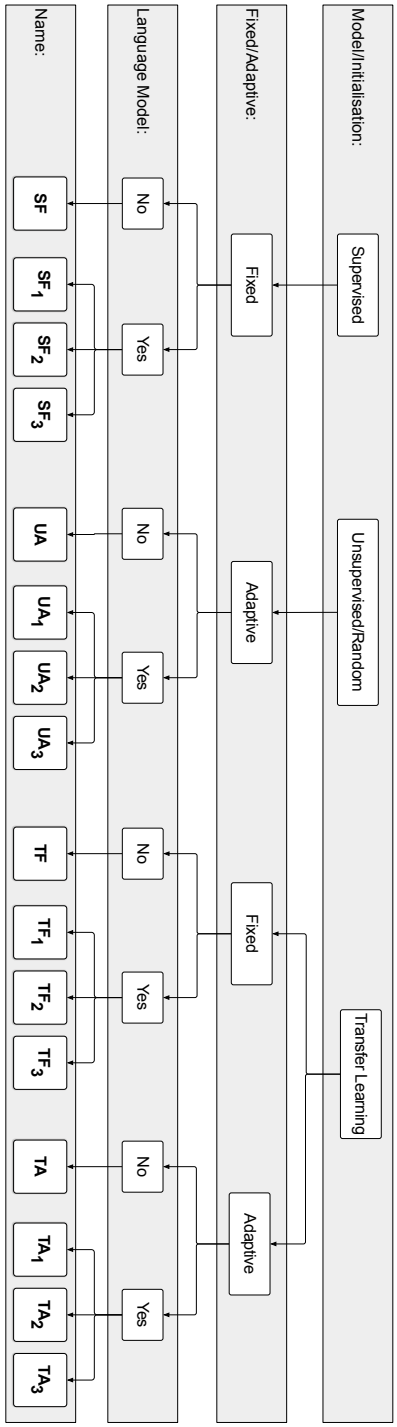


Figure 5.4: Overview of the different methods and their properties.

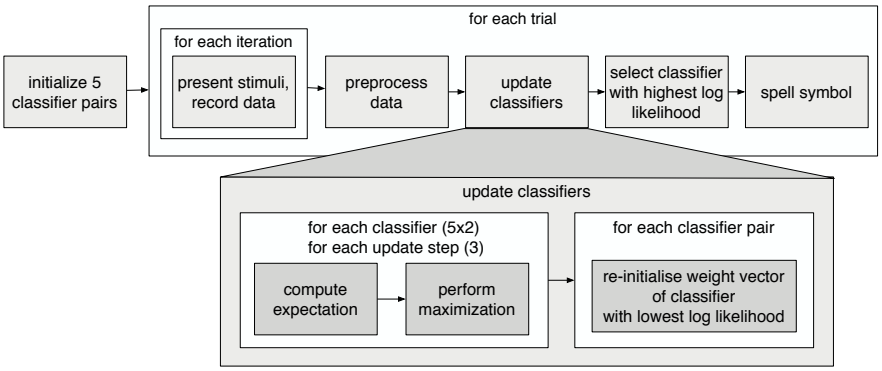


Figure 5.5: Representation of the algorithm used for the UA classifier.

The TF models are re-used in the TA experiments. The general model is used as an initialisation and as regularisation parameter during subsequent adaptation steps. As a result, only a single classifier has to be maintained and the selection and re-initialisation procedures from the UA model can be omitted in this experiment. Analogous to the UA model, we execute 3 EM updates per trial. However, updating the classifier before spelling the symbol is not feasible in combination with dynamic stopping. The updates in our experiment took at most 1.1 s, which would cause a large delay for dynamic stopping. Therefore we deferred the classifier updates, and executed those only after the desired symbol was predicted. A graphical representation of this approach is shown in Figure 5.6.

Finally, we have two extensions/modifiers for these models: language models and dynamic stopping

Language models

The language models can be used in combination with all aforementioned methods. When language models are used, we will denote this by using the subscript $1, 2, 3$ to denote uni-, bi- or tri-gram models. Furthermore, for all methods, the language models work under uncertainty of the previous update and use the forward-backward algorithm during inference. This differs from the approach in Chapter 3.

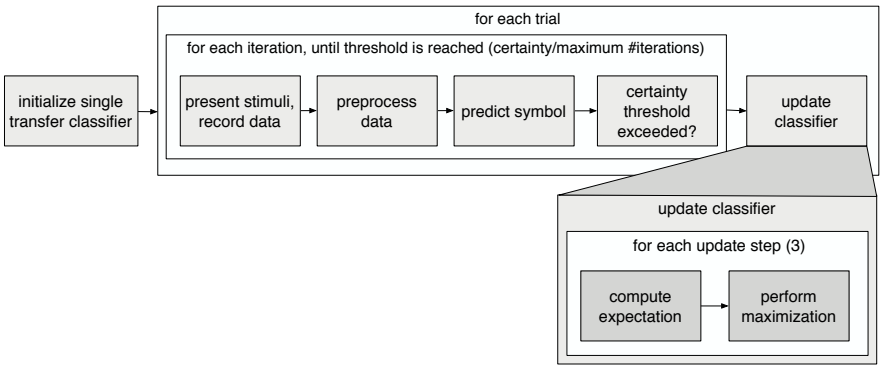


Figure 5.6: Representation of the algorithm used for the DS-TA classifier. The transfer learning initialization compromises unsupervised training on date recorded from other users, followed by combining these subject-specific models into a general model. Bypassing the *certainty threshold block* reduces this algorithm to the basic TA algorithm. Omitting the *update classifier block* reduces this algorithm to the TF version. Please note that the stimulus presentation and data recording can be executed in parallel (i.e. in separate threads) with the preprocessing and spelling components.

Dynamic stopping

The dynamic stopping approach will be denoted by the prefix *DS-*. The dynamic stopping strategy will only be used in combination with SF, TF and TA. It cannot be used in combination with UA, as the classifier is initially unreliably due to the warm-up. The certainty threshold was set to 0.99.

For clarity, we have included an overview of the methods and their properties in Figure 5.4.

5.2.2 Results and Discussion

In Table 5.1 we present the spelling accuracy for a subset of unsupervised and supervised models, both with and without dynamic stopping or language models. An overview of the results of all unsupervised approaches is given in Figure 5.7, the results for the re-evaluations are given in Figure 5.8.

It should come as no surprise that the most advanced method performs best. For 15 iterations per trial, the updated estimates from

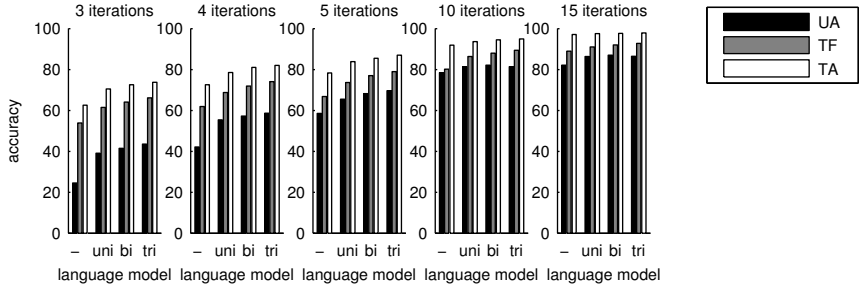


Figure 5.7: Accuracy for simulated online prediction using the unsupervised models.

TA_3^U are correct in no less than 99.5 % of the trials. Moreover, the online evaluation TA_3 achieves 97.9 % accuracy. The introduction of dynamic stopping (DS-TA_3) results in a slight performance drop (to 95.3 %), but, on the other hand, it reduces the number of iterations per trial by a factor 3. Our basic unsupervised model UA achieves 58.6 % accuracy when it is forced to use the same number of iterations per trial (5). In what follows we will investigate how the different components contributed to this impressive improvement. Afterwards, we will give a comparison based on the symbols per minute (SPM) measure to contrast it with a state of the art supervised model.

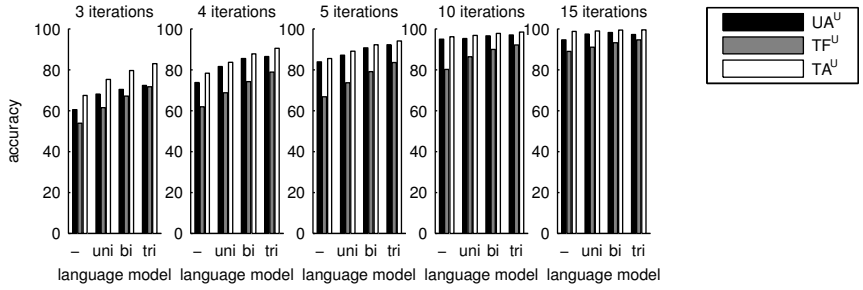


Figure 5.8: Accuracy obtained when the final updated prediction of the model after processing all trials is used.

Warm-up, an essential limitation to basic unsupervised learning

Our unsupervised model UA, which we introduced in the previous chapter, is able to learn a high quality decoder when a substantial amount of data is available. However, as we discussed in the previous chapter, it breaks down when the number of iterations is reduced drastically, especially when used in an online experiment. Online, only 24.5 % of the trials is classified correctly by UA when 3 iterations per trial are available. This result is above chance level, but far from usable in a real BCI. Increasing the number of iterations to 10 or 15 improves the decoding accuracy to 78.4 % and 82.1 %. However, most mistakes are made at the beginning of the simulation. This is caused by the warm-up effect and as a result, these mistakes can be corrected by the updated re-evaluation UA^U. The revised symbol predictions made by UA^U are correct in almost 95 % of the trials for both 10 and 15 iterations. Furthermore, it is this warm-up effect that prevents us from combining UA with dynamic stopping.

Influence of Transfer Learning

The key to eliminating the warm-up period is the use of transfer learning. A fixed transfer learning model TF is able to achieve 66.8 %, 80.1 % and 89 % accuracy for 5, 10 and 15 iterations. However, when the number of iterations is limited to only 3 or 4, the performance remains rather poor and TF cannot compete with state of the art supervised models. To enhance performance we can re-introduce unsupervised adaptation, dynamic stopping and language models. We will begin with the latter.

Influence of Language Models

Unsupervised learning is made possible by exploiting paradigm based constraints on the labelling of individual data points. The constraints posed by the paradigm are hard constraints. In each trial a single type of stimulus must result in a target ERP response. All other stimuli must result in non-target responses. When we introduce language models, we include an additional set of soft constraints. Consequently,

the language models are able to steer the optimisation process towards decoding *unsurprising* texts.

The combination of transfer learning with a tri-gram language model TF_3 is able to predict 79 %, 89.4 % and 92.3 % of the symbols correctly for 5, 10 and 15 iterations. A supervised approach with a tri-gram language model, SF_3, outperforms TF_3 and correctly decodes 92.2 %, 96.3 % and 97.7 % of the symbols. However, this online evaluation, where the symbols cannot be corrected once they are predicted, does not fully utilise the power of the language model. Indeed, only information from preceding trials was used during the prediction. By using the updated predictions, which makes use of the forward-backward algorithm during inference, TF_3^U gets 83.5 %, 92.2 and 94.6 % of the symbols correct. Furthermore, the supervised model, SF_3^U, is correct in 94.5 %, 97.3 % and 98.1 % of its revised predictions. This result clearly demonstrates that language models are only used to their full potential when they are able to revise past prediction.

Influence of (Unsupervised) Adaptation

Language models are able to boost performance, but a general subject-unspecific model will always be outperformed by a subject-specific one. Therefore, we re-introduce unsupervised adaptation, which is the TA model. The adaptive transfer learning model TA is able to predict 78.4 %, 91.9 % and 97.1 % of the trials correctly for 5, 10 and 15 iterations. Clearly, transfer learning and unsupervised adaptation are the two key techniques to achieve state of the art performance in a calibration-less ERP-BCI. Language models can enhance this result even more, especially when the number of iterations per trial is limited. By putting a tri-gram model on top of TA which results in the TA_3 model, we achieve 87.0, 95.0 % and 97.9 % accuracy for 5, 10 and 15 iterations, which is, as discussed in the beginning of this section, very close to the supervised SF_3 model. The final re-estimate for TA_3^U obtains an accuracy of 99.5 % for 15 iterations. Furthermore, even when the number of iterations is reduced, the TA_3 model does not break down. As a result, when using only 3 (4) iterations per trial, TA_3 spells 73.8 % (82.1 %) of the symbols correctly. The final

Iterations	UA	UA^U	TF	TF_3	TF_3^U	TA	TA^U	TA_3	TA_3^U	SF	SF_3	SF_3^U
3	24.6	60.5	53.9	66.1	71.7	62.5	67.5	73.8	83.0	74.5	85.3	89.4
4	42.2	73.7	61.9	74.1	78.9	72.6	78.3	82.1	90.5	81.8	89.5	92.8
5	58.6	83.8	66.8	79.0	83.5	78.4	85.5	87.0	94.1	85.6	92.2	94.5
10	78.4	94.9	80.1	89.4	92.2	91.9	96.2	95.0	98.4	93.4	96.3	97.3
15	82.1	94.6	89.0	92.3	94.6	97.1	98.8	97.9	99.5	96.9	97.7	98.1
DS	-	-	89.6	92.6	94.2	93.3	95.2	95.3	97.4	95.0	96.3	96.8

Table 5.1: Overview of spelling accuracy for a selected subset of methods. DS signifies dynamic stopping.

re-estimate from TA_3^U predicts 83.0 % (90.5 %) of the symbols correct. To put this into perspective, the updated estimate of SF_3^U is only slightly better with 89.4 % (92.4 %) correct symbols.

Influence of Dynamic Stopping

Our analysis in Chapter 3 has shown that a supervised model performs best when dynamic stopping is used instead of a fixed number of iterations. Therefore it is interesting to verify whether this is also the case for transfer learning methods. This question is rather non-trivial. Our experimental results show that the TA model performs well with a limited number of iterations, but the fixed transfer model TF performs quite poorly with only 3 iterations per trial. For dynamic stopping to work, the estimate of the classifier's confidence level must be accurate in a fixed transfer learning model, especially during the first few iterations and trials.

Our dynamic stopping based results are given at the bottom of Table 5.1. We begin our analysis with DS-TF, the fixed transfer learning model. In this model, each trial gives us a direct estimate of the reliability of a general transfer learning model in combination with dynamic stopping. The result is quite surprising, DS-TF uses a rather high number of iterations, 10.3 on average, and obtains an accuracy of 89.6 %. Although promising, this result is far from the one obtained by the supervised DS-SF method, which is much more accurate (95.0 %), and requires approximately half the number of iterations (5.5).

The re-introduction of unsupervised adaptation, the DS-TA model, reduces the average number of trials to 6.3 and increases the accuracy to 93.3 %. Clearly, the adaptation not only allows the classifier to improve its accuracy but also to speed up the decision making process. The difference between a supervised model and an unsupervised model diminishes by incorporating unsupervised adaptation. Finally, similar to our supervised study, Figure 5.9 shows that incorporating language statistics, increases the accuracy and reduces the required number of iterations for all techniques. This brings us back to our ultimate model, DS-TA_3, that gets 95.3 % of the symbols correct with only 4.8 iterations on average. This is incredibly close to the supervised model DS-SF_3, that performs marginally better at 96.3 %

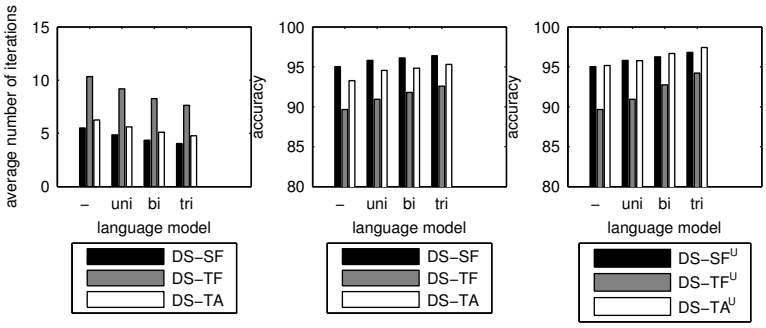


Figure 5.9: The effect of a language models on the required number of iterations and the spelling accuracy for dynamic stopping based methods.

accuracy and an average of 4 iterations per trial.

Impact on practical BCI usage

Now we will put our final results in a different perspective. The DS-TA approach uses slightly more (0.8) iterations per trial than the DS-SF model. This boils down to 1.7 s extra per trial. On the other hand, DS-SF requires a supervised calibration recording, which lasts 16 trials in our experiment. Such a calibration recording takes more than 10 minutes to complete. Hence, to make up for the time lost during calibration, the spelling session should last 345 trials. In this calculation, we have assumed that both models will achieve similar accuracy. However, our previous experiments have indicated that during long term use, unsupervised adaptation can outperform supervised (fixed) models, which would produce a result that is even more favourable for the transfer learning model.

To conclude our offline evaluation we compare DS-TA and DS-SF using the Symbols Per Minute (SPM) approximation (Schreuder et al., 2011a), which we also discussed in Section 1.2.3. The goal of SPM is to measure the spelling speed in an application in which the user has to correct his mistakes using a backspace command. This differs from our previous evaluation, where we assumed that the language model/unsupervised adaptation is used to correct the mistakes. Hence, to make the evaluation as general as possible, we will start with the

basic transfer model without language based extensions.

Using this metric DS-TA obtains 2.9 SPM and DS-SF achieves 3.4 SPM. Clearly, DS-SF is a little bit faster. However, this result did not take the calibration session into account. Imagine that we have two users, one using DS-TA and one using DS-SF. By the time that the DS-SF user can start spelling, i.e. after the calibration session, the DS-TA user will have spelled 29 symbols correctly. This increase in efficiency can have a significant impact, especially for patients with a limited attention span.

5.2.3 Summary

in our offline evaluation we have presented the results from a simulation study that allowed us to isolate the contributions of transfer learning, language models and unsupervised adaptation. We found that the combination of unsupervised adaptation and transfer learning are key to building a true zero-training ERP-BCI that can compete with state of the art supervised models. For this reason, we will investigate the use of transfer learning and adaptivity in a real online experiment using the challenging AMUSE paradigm.

5.3 Online evaluation

An online study presents the greatest challenge for BCI decoders. To validate the robustness of our proposed transfer learning model, we present the results from our second online study, which was also conducted at the TU-Berlin in collaboration with Michael Tangermann, Martijn Schreuder and Klaus-Robert Müller. Our offline simulations have shown that transfer learning and adaptation are essential to build a true zero-training BCI. Therefore, in our online study, we validate transfer learning in combination with unsupervised adaptation on the AMUSE paradigm.

5.3.1 Experimental setup

This study is an extension of our online evaluation of the unsupervised model. Therefore, we will briefly summarise the experimental setup and focus on the differences with the study presented in the previous chapter (see Section 4.3.1).

AMUSE

Once more, we have used the auditory ERP paradigm AMUSE. We detailed this paradigm in the first chapter (Section 1.2.2) and specific details on this study are given in the previous chapter (Section 4.3.1). What follows is a brief summary of the paradigm. AMUSE is an ERP paradigm that relies on auditory stimuli. There are 6 stimuli in total, which can be identified using their tone and location. Each tone is assigned to one of 6 speakers which are placed on a ring around the participant. By focusing on a specific stimulus, the user can perform an action. These actions are used in a two-step process to spell one out of 36 symbols.

Participants

All participants in this study were healthy and non-smokers. They did not take any EEG-altering or psychoactive medication. All of them were external to the lab and were recruited through an online advertisement. We followed the Helsinki Declaration and provided all subjects with information on the experiment. They received detailed information on the experiment about one week in advance and we requested them to rest well the night before and to have a hair-wash on the day of the recording. Before starting the experiment they declared written informed consent.

In this follow-up study, we have two distinct groups of participants. The first group consists of the participants from the first study, who were re-invited to take part in the transfer learning evaluation. All but one subject, *nbe*, participated in this follow-up study. The returning subjects are:*nbb, nbc, nbd, nbf, nbg, nbh, nbi, nbj, jh.* The second group, the new participants are represented by the following codes *nbq, nbs, nbu, nbv, nbw, nbx, jpc, nca, ncb, ncc, ncd.*

Methods

Only the transfer learning model with unsupervised adaptation TA, in which the EM updates were executed before making a prediction, was evaluated online. However, for comparison we will use a supervised LDA model, a basic unsupervised learning model UA and a fixed transfer learning model as references.

The transfer model was trained on the data from the basic unsupervised study. We performed unsupervised training for each subject individually on all 6 evaluation blocks. The resulting best classifier based on data log likelihood was used for transfer learning. For the returning subjects, we performed leave one subject-out transfer learning. For the novel users we combined all models from the original study. The transfer classifier was initialised with the general model, β^{-1} was initialised to 0.5 and the regularisation parameter α was initialised to 200, which is also its maximum allowed value.

Course of the experiment

The experiment was structured similarly to the previous study (see Section 4.3.1). Therefore, we will shortly summarise the setup. Each experiment began with an oddball familiarisation and recording phase. Afterwards, we proceeded with the familiarisation block on auditory spatial attention. Before starting the real experiment, we educated the users on the copy spelling application. Once this familiarisation was completed, we started the true BCI experiment. Each user performed three transfer learning evaluation blocks. Each block lasted 30 trials (15 symbols), where the users were free to select the target text. An overview of the experiment is given in Figure 5.10. For more details on the contents of the individual blocks, please refer to Section 4.3.1.

5.3.2 Results and discussion

Re-invited subjects: online analysis

We begin by presenting the results from the true online experiment, starting with the re-invited participants. Performance of the re-invited

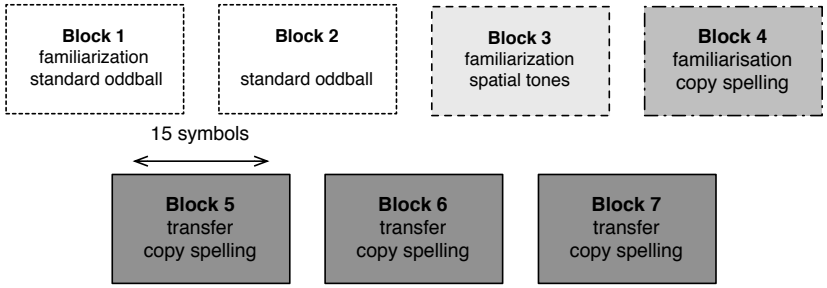


Figure 5.10: Overview of the structure of the transfer learning online study.

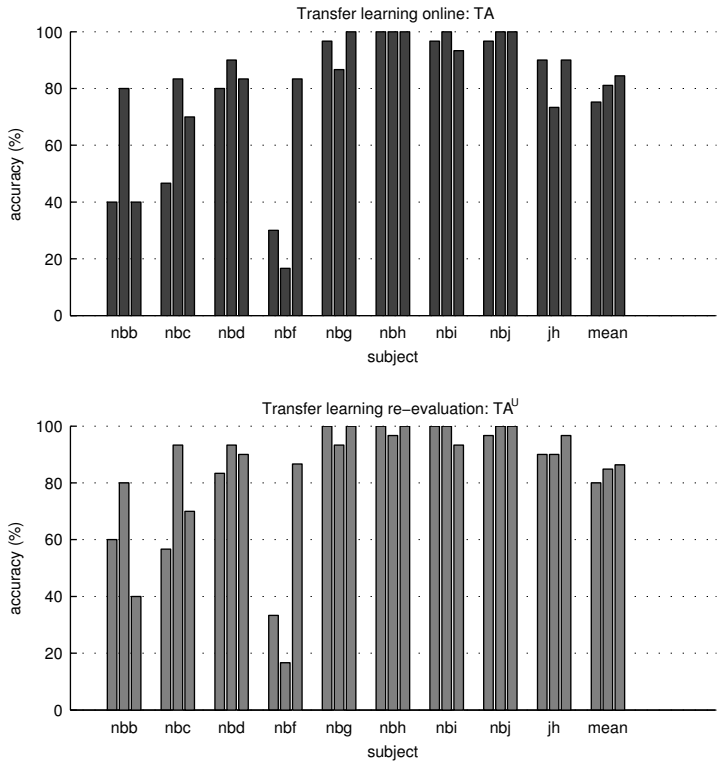


Figure 5.11: Performance comparison between the online adaptive transfer method TA and the re-evaluation TA^U in the online experiment with the re-invited subjects. Each bar corresponds to an experimental block.

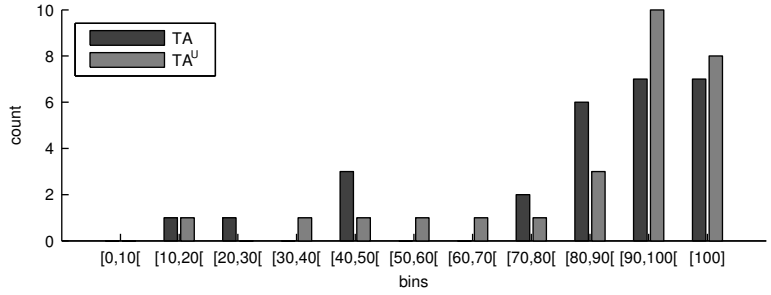


Figure 5.12: Histogram of the selection accuracy in the online experiment for the re-invited subjects.

;

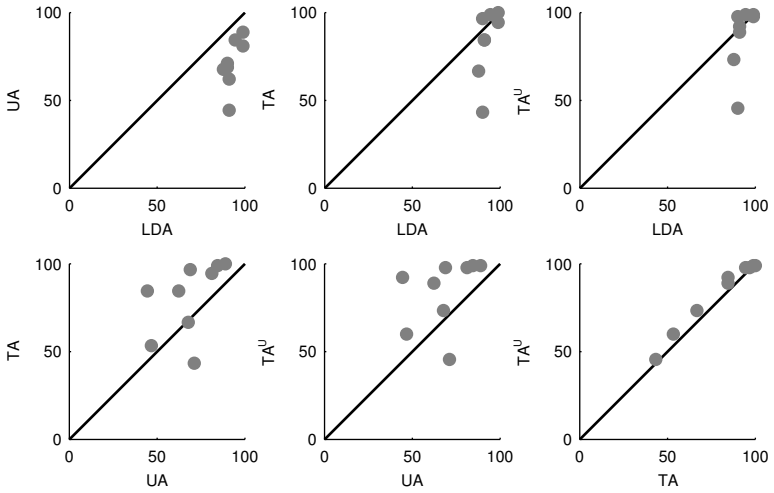


Figure 5.13: Subject-wise comparison on the selection accuracy for the re-invited subjects between the results obtained in the transfer study and those obtained in the unsupervised online study.

subjects is shown in Figure 5.11, a histogram of the selection accuracy is shown in Figure 5.12 and subject-wise comparison to the results from the unsupervised online study is given in Figure 5.13.

Overall, the re-invited participants achieve an average accuracy of 80.3 % with the adaptive transfer model TA. For reference, in the first online study, the supervised model SF obtained 92.6% and the unsupervised model UA was correct in only 68.5 % of the trials. The post-hoc re-evaluation TA^U resulted in an accuracy of 83.7 %, which is a minor improvement over UA^U, from the previous study, which obtained 81.5 %. Moreover, when we analyse the subject-wise comparison in Figure 5.13, we see that the performance of transfer learning is comparable to that of the supervised classifier in all but three subjects. Furthermore, when we zoom in on the individual subjects, we see that 4 out of 9 subjects spelled flawlessly in at least a single block and a total of 7 blocks was error-free. On the other hand, the performance for subjects *nbb, nbc* and *nbf* is below average and there is also a large amount of variability between the different runs for these subjects. The performance of the remaining subjects is actually quite close to that of the supervised model in the original study.

We observed an increase in classification accuracy over the three evaluation blocks. There is, however, no real general learning effect present. When subject *nbf* is excluded from the analysis then the second block achieves the best performance overall (almost 90 %). Similarly, subjects *nbb* and *nbc* are responsible for the performance drop of the first block.

Re-invited subjects: offline analysis

Additionally, we have performed a short offline analysis to compare performance between LDA, UA and TA on the same data. For this reason, we have re-run the simulated extended experiment on the data from the first unsupervised study. In our results, which are shown in Figure 5.14, we see that transfer learning has doubled the accuracy in the first 10 trials, but has not entirely solved the warm-up problem. Furthermore, from the third sub-block onwards, all unsupervised methods and the supervised LDA model perform comparably. The only exception to this is observed in the final sub-blocks, where

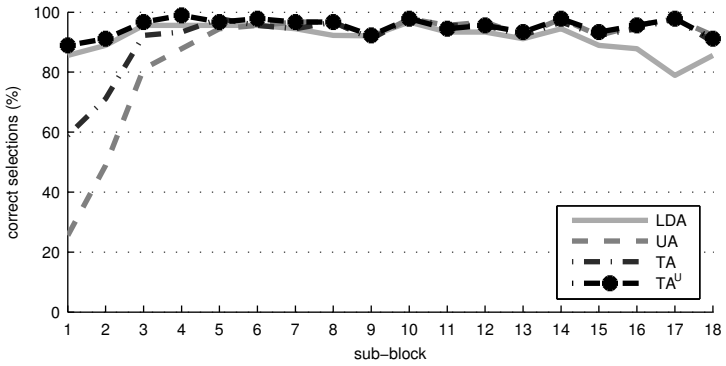


Figure 5.14: Performance evolution for the re-invited subjects on the data from the original online study. Each sub-block is 10 trials long.

the LDA performance is reduced compared to the unsupervised approaches. As we discussed in the previous chapter, this is probably caused by non-stationarity, which a fixed classifier cannot deal with. Furthermore, in Chapter 6, we will present the results on data from the original AMUSE study, comprising 21 different subjects, which corroborates this result.

Novel subjects: online analysis

Now, we turn to the new subjects, who have no prior experience in this paradigm. The block-wise results for transfer learning, both online (TA) and post-hoc (TA^U), are given for all subjects and all blocks in Figure 5.15. The average accuracy for TA is quite a bit lower compared to the re-invited subjects with only 63.0 %. This result is even below the average result for basic unsupervised learning in the previous online study. Here, we do observe a strong learning effect over the blocks. The first block is completed with an average TA accuracy of 54.2 % and the final block is completed with an accuracy of 72.1%. Furthermore, this learning effect is very clear in 8 out of 11 subjects. The post-hoc re-analysis improves the results to 68.3 %, but here the learning effect remains visible. The first block is completed with 60 % accuracy and the average of the last block is 78.2 % accu-

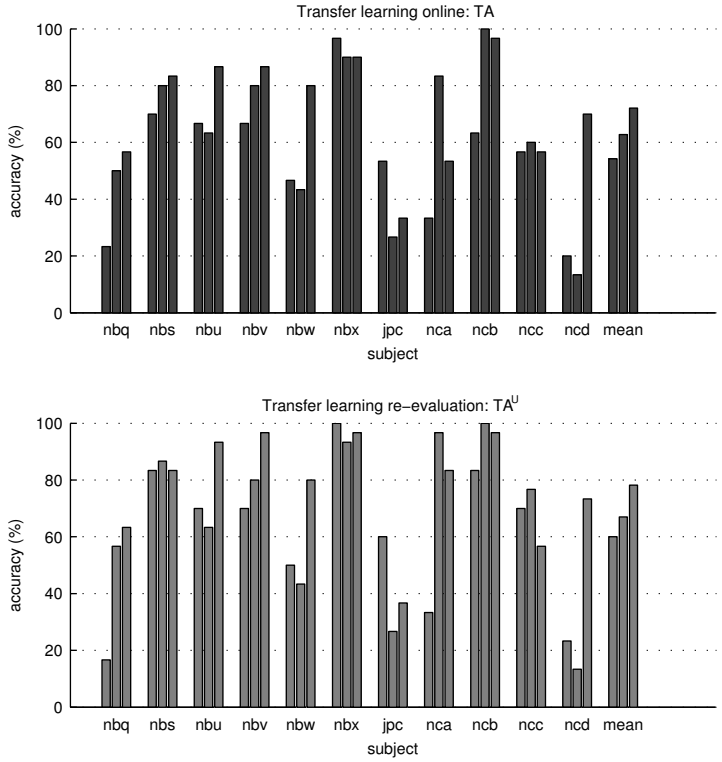


Figure 5.15: Performance comparison between the online adaptive transfer method TA and the re-evaluation TA^U in the online experiment with the novel subjects. Each bar corresponds to an experimental block.

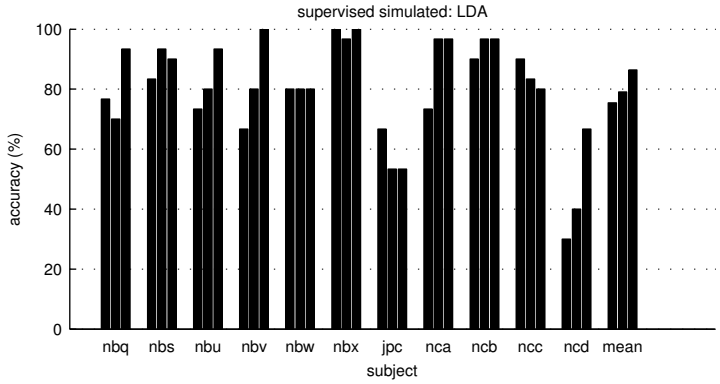


Figure 5.16: Accuracy for the novel subjects, obtained by a supervised simulation. Each bar corresponds to an experimental block.

racy. This amplified learning effect could partially explain the drop in performance compared to the re-invited subjects. However, for a better understanding, we have to resort to offline analysis.

Novel subjects: offline analysis

First, we want to determine whether the recordings for the novel subjects are of the same quality as those in the original unsupervised study. For this reason, we used the following cross-validation approach with the LDA classifier. The data from evaluation block 1 was used to train a classifier which was evaluated on the second experimental block, data from the second experimental block was used to train a classifier to evaluate the third block and the classifier trained on the third block was evaluated on the first block. The results for this experiment are shown in Figure 5.16. Here, we observe that overall, the accuracy for the supervised evaluation is 80.3 %, which is 10 % lower than the result obtained by the other subjects in the original unsupervised study. Furthermore, the learning effect is much stronger, the first block is completed with an accuracy of 75.5 % and the last one with an accuracy of 86.4 %. From these results, we conclude that the data from the novel subjects is much harder to decode without label information than the data recorded on the re-invited subjects.

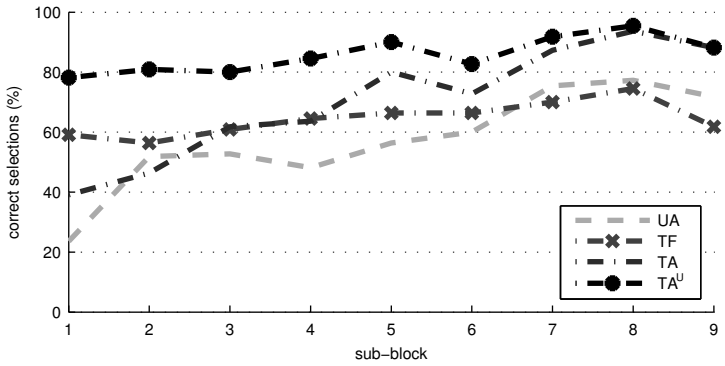


Figure 5.17: Simulated online experiment on the concatenated data from the novel subjects. Each sub-block lasts 10 trials.

However, it should not cause the transfer learning adaptation based method to break down.

To find out what happened, we started a second simulation, where we compare the basic unsupervised model UA to the fixed transfer learning model TF and to the adaptive transfer learning model, of an extended spelling session on all concatenated experimental blocks. The results of this simulation are shown in Figure 5.17, in which we computed the accuracy per sub-block of 10 trials. We see clearly that the fixed transfer model TF improves over time, from 59.1 % in the first sub-block to 74.6 % in the last sub-block. A similar performance improvement can be observed for our re-evaluation of the adaptive transfer learning model TA^U, which obtains 78.2 % accuracy in the first sub-block and up to 95.5 % accuracy in the second to last sub-block. Please note that the TA^U model is fixed during evaluation, the same classifier is used on all sub-blocks. This observation, combined with the TF evaluation and our supervised validation clearly indicates strong non-stationarity in the signal. Furthermore, we see that in the first few sub-blocks TF is more accurate than UA and TA, the reason for this is simple. The adaptive methods had started (over-)fitting on a non-informative component in the signal, presumably outliers[2]. However, when these adaptive methods had observed

[2]This is also a risk for supervised methods in the calibration session. Furthermore, we have anecdotal evidence that this happened during the original AMUSE

enough clean data, they recovered and were able to learn a good model for the data, that can cope with those erroneous effects at the beginning of the experiment. Moreover, the final adaptive transfer learning model is actually more accurate than the supervised LDA model's validation result, with 85.8 % of the attended stimuli detected correctly compared to 80.3 % for the supervised model.

Discovering the true cause of the differences in signal quality between the two user groups is hard. There is a much stronger learning effect for the novel subjects. Moreover, the concept of spatial auditory attention in AMUSE can be quite difficult for some people. As a result, the familiarisation phases, where the paradigm is introduced to the subject, might have influenced the results. Unfortunately, it is not possible to get a definitive answer from the experiments. However, it gives us an excellent opportunity to stress the following aspect of BCI research. BCI is an interdisciplinary field, it is important to develop novel paradigms that can be used efficiently, with a learning curve that is as short as possible. Hence, the optimisation of the paradigms, which is complementary to the machine learning optimisation, is equally important in the development of true zero training BCI. Improving the machine learning can result in big leaps forward in the field of BCI, but we should never forget that BCI is much more than a data-analysis problem. It is a collaboration between man and machine, which makes the problem much more difficult and unpredictably but also much more interesting.

5.4 Conclusion

In this chapter, we have proposed a novel transfer learning approach for ERP based BCI. This transfer learning model can be combined with unsupervised adaptation. It shares information on how to decode the brain signals across users. This shared information reduces the warm-up period in an unsupervised BCI. We have evaluated this method extensively, offline on visual and auditory ERP data and on-

study (Schreuder et al., 2011b). One of the subjects was not able to complete the experiment due to low (supervised) accuracy. The unsupervised method on the other hand is able to decode that data reliably.

line on auditory ERP data. In total, data from 42 subjects was used in our analysis.

Our results indicate that transfer learning is indeed possible but results in a decoder which is not as reliable as a subject-specific model. The combination of transfer learning and unsupervised adaptation results in an unsupervised model that can compete with state of the art subject-specific classifiers on visual ERP data. Our simulations show that by using transfer learning in combination with adaptation and dynamic stopping, a user can spell 29 symbols in the time normally used for calibration. This significant increase in efficiency can prove to be invaluable for BCI use by patients. The online results on auditory data show a similar increase in performance for the re-invited subjects. In contrast to the visual data, there is still a tiny warm-up effect, but the effect is much smaller. This indicates that true-zero training is indeed a possibility for ERP based BCI, even when a challenging auditory paradigm is used. However, our result on a novel set of BCI-naive subjects did not show the same level of reliability. Nevertheless, transfer learning was able to significantly reduce the warm-up period, but it is clear that there is a strong learning effect present in the users. This demonstrates that machine learning can improve the field of BCI, but we should not forget that it is a strong interdisciplinary field. Machine learning is only part of the solution to build true zero-training BCIs. Such a true zero training BCI is only possible when the user can quickly learn how to use the BCI.

Finally, we have neglected one important aspect of transfer learning. We did not answer the question: "Why does transfer learning work?". The next chapter is devoted to enhancing our understanding of transfer learning.

6

Understanding unsupervised transfer learning for ERP based BCI

In the previous chapters, we have introduced a unified probabilistic model for ERP based BCI, which can incorporate language statistics, a learning rule that enables us to train the classifiers reliably, and a transfer learning component that allows decoder information to be shared across users. We have also shown that these approaches work reliably in offline experiments and in an online auditory BCI setup.

However, we did not investigate why transfer learning actually works. For transfer learning to work, there has to be some similarity in the information content between the different subjects. The differential activation patterns (the average target activation minus the average non-target activation) shown in Figure 6.1 indicate, however, that there is a lot of variability between users. Furthermore, partially due to this high inter subject variability, the BCI community has a long history of using supervised subject-specific models (Cecotti and Gräser, 2010; Blankertz et al., 2006b; Hoffmann et al., 2008). For this reason, one could assume that transfer learning should not work, or should perform rather poorly. The results from the previous chapter and the related transfer learning work on ERP based BCI (Kindermans et al., 2014, 2012b; Colwell et al., 2013; Jin et al., 2012; Lu et al., 2009) indicate otherwise.

We argue that even though the subjects appear to be different, the difference is actually rather superficial. The dimensionality of the feature space is so vast that humans cannot comprehend the data. Such a high dimensional feature space is hard to interpret, and for this reason we have to rely on machine learning techniques. Inves-

tigating the differential activation patterns can be seen as a form of mass univariate analysis. The multivariate interactions are not taken into consideration, while they are actually very important in the computation of the direction of the decision boundary.

Nevertheless, it is still sensible to analyse these differential activation patterns. Recall the work of Blankertz et al. (2011), where it was shown that ERP features can be approximated by a class dependent Gaussian distribution, in which the means differ but the covariance matrix is shared between the classes. The mean target and mean non-target activations are *prototypes* of the two classes, which give a little bit of insight in the underlying neurophysiological process. However, we should not forget that to properly discriminate between target and non-target responses, we have to take the covariance structure of the data into account. It is the covariance structure that determines the direction along which $\boldsymbol{w}$ separates the two classes.

Given this knowledge, our transfer learning hypothesis is as follows. First, the informative subspace, i.e. the subspace that contains the class discriminative information, is significantly smaller than the original feature space. Second, the structure of the informative subspace is similar across different users. Third, the information that is present in this subspace is structured comparably between users. Or to put it in other words: the users are rather similar within this subspace. In the next section, we will elaborate on these hypotheses.

Given this hypothesis, several questions arise: "Is it possible that the features are very similar in the informative subspace but appear to be very distinct in the original (measured) feature space?", "How does this hypothesis relate to the transfer learning approach we proposed in the previous chapter?" and "How do we verify this claim using real EEG data?". On top of these questions, we would like to demonstrate that we can also infer the corresponding neurophysiological activation directly from the unsupervised classifier. In the previous chapter we discussed that we have no direct control over what the classifier actually learns. Here we will show that the classifier does truly succeed in learning the correct task and that, with a minor modification, we can recover the underlying activation patterns directly from the classifier, without using label information.

We will first discuss the transfer learning hypothesis and the rela-

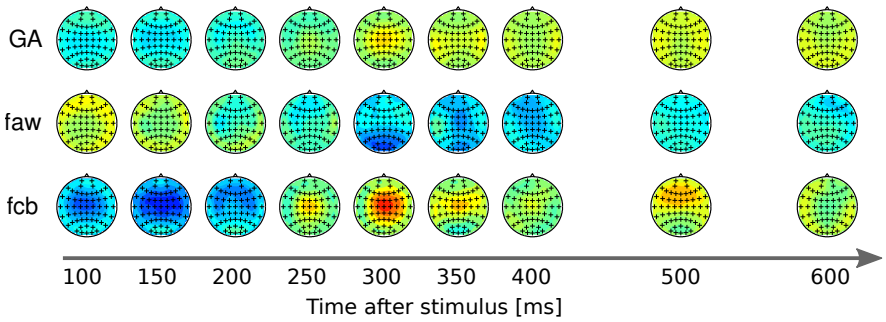


Figure 6.1: Differential activation patterns (mean target activity minus mean non-target activity) on the scalp surface after stimulus presentation. The top line shows the average over all subjects, the other two lines show the patterns for two selected subjects. Green corresponds to 0, red to a positive deflection, blue to a negative deflection.

tion to the transfer learning model from the previous chapter. Afterwards we will discuss Relevant Dimensionality Estimation (RDE), a method that is proposed by Braun et al. (2008) and is merely used as a data-analysis tool in this work. We will make a minute change to the probabilistic model, such that it can recover the activation patterns. For this reason, we will demonstrate on the original AMUSE dataset (Schreuder et al., 2011b) that this change does not reduce performance. Afterwards, we will discuss our investigation of the discriminative subspaces and the similarity between users.

6.1 Understanding transfer learning

Our transfer learning hypothesis states that the class-discriminative information is contained in a small subspace and that this subspace is similar for the different users. To make this more concrete, we have constructed a toy example, which is depicted in Figure 6.2, comprising 29 channel artificial data from two subjects. For more information on how we have generated this toy example, please refer to Appendix B. In panel A, we have shown the differential activation patterns for both subjects. This differential activation pattern is equal to the av-

erage target response minus the average non-target response. Panel B gives us three examples of target responses and three examples of non-target responses per subject. Clearly, the data from the subjects differs greatly in the feature space, i.e. the measured EEG. However, we have constructed this toy example such that transfer learning must be possible. Both subjects share the same two dimensional informative subspace. Panel C shows the data projected into this subspace, using the weight vectors $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2$, where we see that both subjects are identical within this subspace. As a consequence, within this subspace the linear classifier $[z_1, z_2]^T$ can be shared between the subjects. The projection into the informative subspace and the classification within this subspace are both linear operations. Therefore, they can be reduced to a single linear operation on the original feature space.

$$\boldsymbol{w} = z_1\boldsymbol{\nu}_1 + z_2\boldsymbol{\nu}_2$$

This is identical to the linear classifier that we use in the BCI's decoder.

This reasoning allows us to interpret the transfer learning approach from the previous chapter in several ways. The first interpretation is that the general model is a usable but suboptimal solution and that by slightly adapting this solution we obtain a subject-specific solution of high quality. However, we can also see it as a classifier which projects the data in the shared informative dimension and classifies the data in this subspace. The subject-specific model continues to use these shared dimensions, but additionally it projects the data in a few additional sub-dimensions which are only relevant for the novel subject. Then it performs classification in all the resulting dimensions at once.

6.2 Finding the informative subspace

To analyse whether the more general assumption about the shared subspaces actually holds for real data, we have to be able to determine the task relevant subspace for the different subjects. For this, we will use Relevant Dimensionality Estimation (RDE) (Braun et al., 2008),

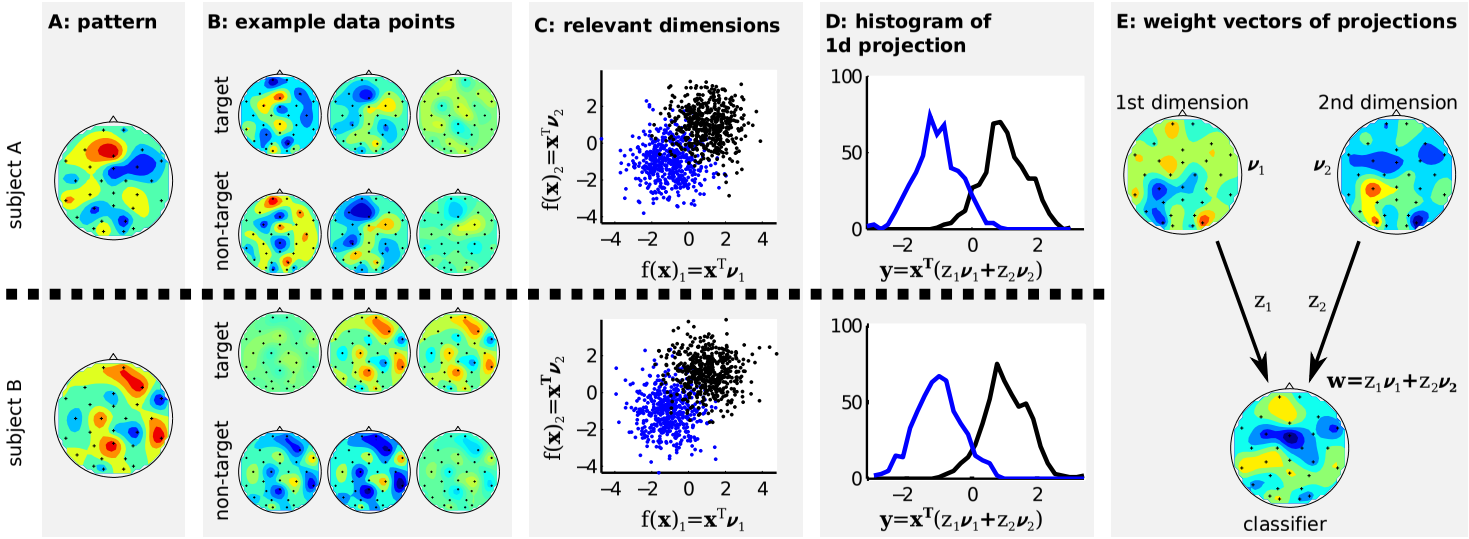


Figure 6.2: Illustration with artificially generated data of a discriminative subspace shared between two distinct subjects. In (A) we present the differential activation patterns, which are equal to the difference between the average target and non-target response. (B) gives three examples of a target and a non-target response per subject. The projection of the all data-points into the relevant subspace is given in (C) and the vectors used for this projection are given on top in (E). The projection of target responses is plotted in black, non-target responses are blue. Within the task relevant subspace, which is shared between the subjects, the subjects are identical. Therefore, we can classify the data of both subjects using the same weight vector. (D) shows the histogram of this projection. Finally, this classification can be executed in the original feature space by using a linear combination of the projections into the relevant dimension. The resulting weight vector is shown on the bottom of (E).

which is a supervised data analysis method,based on Kernel PCA (kPCA) (Schölkopf et al., 1998). We would like to stress that this method is presented in literature. We have merely used it as a tool to analyse the ERP data.

RDE estimates the dimensionality of the task-relevant subspace of the data, which contains the informative part of the data and excludes the noise components. For this, it uses a decomposition of the data based on kPCA and exploits the relation between the target labels

$$\boldsymbol{y} = (y_1, \ldots, y_n)^T$$

and the kPCA components

$$f(\boldsymbol{x})_m ,$$

which corresponds to the projection in the original subspaces in the toy example. This allows us to use RDE to distinguish between learning problems that are difficult because of the inherent complexity of the signal or because of the low signal to noise ratio. The mapping obtained by the decomposition depends strongly on the kernel function used. However, we know that ERP data is linearly separable, so we can use a simple linear kernel. Nevertheless, we have used kPCA and not standard PCA such that we can still rely on the findings from Braun et al. (2008) with respect to the relevant dimensionality estimation.

Now we will introduce RDE in relation to the transfer learning toy example from Figure 6.2. To estimate the RDE, we compute the (kernel) principal directions of the (training) data:

$$\boldsymbol{\nu}_m,$$

which corresponds to the projection vectors towards the (informative) subspaces in the toy example. Let $\boldsymbol{x}_i$ be a feature vector from the labelled dataset containing N examples and $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and $\phi(\cdot)$ a (possibly non-linear) feature map, which in our setting is the identity function:

$$\phi(\boldsymbol{x}) = \boldsymbol{x}.$$

By computing the eigenvalues and eigenvectors of the kernel matrix:

$$K = X^T X,\ KU = \Lambda U,$$

where $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ contains the eigenvectors and $\Lambda = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$ the eigenvalues, we can obtain the principal directions:

$$\boldsymbol{\nu}_m = \phi(X)\frac{\boldsymbol{u}_m}{\lambda_m}.$$

The m-th principal component for an unseen feature vector $\hat{\boldsymbol{x}}$ is the projection on the m-th direction:

$$f\left(\hat{\boldsymbol{x}}\right)_m = \phi\left(\hat{\boldsymbol{x}}\right)^T \boldsymbol{\nu}_m.$$

This projection of the data corresponds to panel C in Figure 6.2, the vector $\boldsymbol{\nu}$ corresponds to projection vectors shown in panel E. The m-th principal component for the i-th point in the training set is equal to the i-th component of the m-th eigenvector:

$$f\left(\boldsymbol{x}_i\right)_m = \boldsymbol{u}_m^i.$$

Because these eigenvectors are orthogonal, we can compute the coefficients of the labels $\boldsymbol{y}$ with respect to the basis U by taking the scalar product:

$$z_m = \boldsymbol{u}_m^T \boldsymbol{y},$$

Using the kPCA coefficient as a classifier in the relevant subspace corresponds to classifying the data in Figure 6.2C by projecting it in one dimension. This one dimensional projection was shown in panel D. It is clear that the kPCA coefficients are a linear classifier within the reduced d-dimensional feature space. Hence, we can predict the target label for a new feature vector in the original feature space by projecting $\phi\left(\hat{\boldsymbol{x}}\right)$ into the d-dimensional subspace, followed by classification in this subspace. In our specific case, this corresponds to using a linear classifier directly in the original feature space. This classifier, which is actually a combination of the two projections, is represented

on the bottom of panel E in Figure 6.2:

$$\hat{y}\left(\hat{\boldsymbol{x}}\right) = \sum_{m=1}^{d} z_m f\left(\hat{\boldsymbol{x}}\right)_m = \sum_{m=1}^{d} z_m \hat{\boldsymbol{x}}^T \boldsymbol{\nu}_m = \hat{\boldsymbol{x}}^T \sum_{m=1}^{d} z_m \boldsymbol{\nu}_m$$

This is equivalent to the linear classifiers used in our probabilistic model.

Furtmermore, Braun et al. (2008) have demonstrated that the task-relevant information is contained in the d leading kPCA components. These are the components that correspond to the highest eigenvalues. Additionally they have proposed the following algorithm to estimate the task-relevant dimensionality. First, they fit a zero mean Gaussian mixture model onto the kPCA coefficients:

$$z_i \sim \mathcal{N}\left(0, \sigma_1^2\right) \text{ for } i = 1, \ldots, d \;\; z_i \sim \mathcal{N}\left(0, \sigma_2^2\right) \text{ for } i = d+1, \ldots, n.$$

The dimensionality estimate equals d for which the likelihood is maximised.

6.3 Recovering the activation pattern

The linear classifiers used in this book are spatio-temporal filters, they filter the EEG feature vectors and the result is a value that tells us whether it was a target or a non-target response. However, as is clear from the toy example at the beginning of this chapter (Figure 6.2), the filter weights cannot be interpreted directly. Large weights can contribute directly to the classification result, but they can be present for noise cancelling purposes as well. Indeed, the same filter can be used for two subjects with distinct differential activation patterns.

To understand the "meaning" of the classifier, we have to compute the activation pattern that reflects the underlying neurophysiological process. It has been shown (Haufe et al., 2014) that under mild assumptions, the activation pattern $\boldsymbol{p}$ can be recovered by multiplying the filter $\boldsymbol{w}$, which is our classifier's weight vector, with the correlation matrix of the data

$$\boldsymbol{p} = XX^T\boldsymbol{w}.$$

The desired activation pattern for our application corresponds to the differential activation pattern (mean of the target response minus the mean of the non-target response). However, when we apply the above method to a regression based classifier we obtain the following.

$$\boldsymbol{h} = XX^T(XX^T)^{-1}X\boldsymbol{y} = X\boldsymbol{y},$$

When the targets -1 and 1 are used and the classes are balanced then the resulting pattern is proportional to the differential activation pattern. However, this is not the case in an ERP paradigm due to the 1:5 ratio of target to non-target data points. Hence, when we compute the patterns by left multiplying with the correlation matrix, we obtain a pattern that is proportional to the average target response minus five times the average non-target response. By modifying the targets such that we use 5 for the targets and -1 for the non-targets, we obtain a pattern that is proportional to the difference between the mean target and mean non-target activation. This will allow us to determine whether the classifier is not only able to classify the data correctly but whether it is able to recover the differential activation pattern without label information.

6.4 Experimental evaluation

6.4.1 Experimental setup

Since we have made a small modification to the transfer model (changing the targets for the projections), we will demonstrate that this does not have a detrimental effect on the resulting performance. For this demonstration, we will compare UA, TF, TA and TA^U to a supervised LDA model.

Data

The dataset used in this analysis was recorded in the original AMUSE study by Schreuder et al. (2011b). It contains 21 subjects. Note that this is not the same data as used in the previous chapters. More details on AMUSE, which is an auditory ERP paradigm that uses 6 stimuli

and a two step selection procedure, can be found in the introduction Section 1.2.2) and in the publication that reports the results using the data bySchreuder et al. (2011b). A description of the dataset can be found in Appendix C. The task in this study was to copy spell a pre-defined text, where errors had to be corrected by an undo-/back action. As a result, the total number of online trials performed varied between subjects. The smallest number of trials required was 64 and it took at most 165 trials for the slowest subject. However, a low classification accuracy prevented 5 subjects to finish the sentence. Nevertheless all subjects will be considered in our evaluation.

Pre-processing

To make the interpretation of the features easier, we have introduced a minor change to the sub-sampling compared to the pre-processing procedure used in the previous chapters. The EEG is still processed trial per trial, and we begin by applying a bandpass filter (0.5 -15 Hz) once in the forward and once in the backward direction. Per stimulus we compute a feature vector that is normalised to zero mean and unit variance. We sub-sample the data by taking 9 samples per channel at the following time (in seconds) after stimulus presentation:

$$[0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6].$$

Combined with a bias term, this results in a total of 550 features per epoch. It is important to note that this feature selection approach does not reduce the performance, even though the sampling frequency is below 30 Hz, which would be required to perfectly reconstruct the EEG signal of 15 Hz.

Classifiers

The different methods used in this experiment are the same as in the previous chapters, apart from the target outputs in our unsupervised/transfer models. We list them for completeness.

LDA: supervised baseline
: This is the classifier which was used in the online experiments during the original AMUSE study.

evaluation	LDA	UA	TF	TA
(simulated) online	77.4	74.7	72.3	80.5
post-hoc	-	84.5	-	86

Table 6.1: Results obtained during online simulation. We compare the zero training results with the online experiment conducted in Schreuder et al. (2011b). The results are in line with our previous observations and demonstrate once more that unsupervised learning can be (at least) as reliable as supervised models.

UA: unsupervised adaptation
The UA model is the randomly initialised at the beginning of the experiment and uses unsupervised adaptation during the experiment.

TF: Transfer learning fixed
The transfer learning fixed model, uses the leave one subject out transfer learning approach from the previous chapter. The basic classifiers are the final adapted classifiers from the UA experiment.

TA: Transfer learning adaptive
In the TA experiment, we combine transfer learning (TF) with unsupervised adaptation. We will also include the results from the post-hoc re-evaluation.

6.4.2 Results and discussion

Spelling Performance

To verify that the modification of the target labels did not cause the model to fail, we will analyse the results from the simulation of online spelling. The average performance for the five classification approaches are provided in Table 6.1, while the scatter plots in Figure 6.4 provides a subject-specific, pair-wise comparison of LDA against the four unsupervised methods. Clearly, TA^U performs best, but these

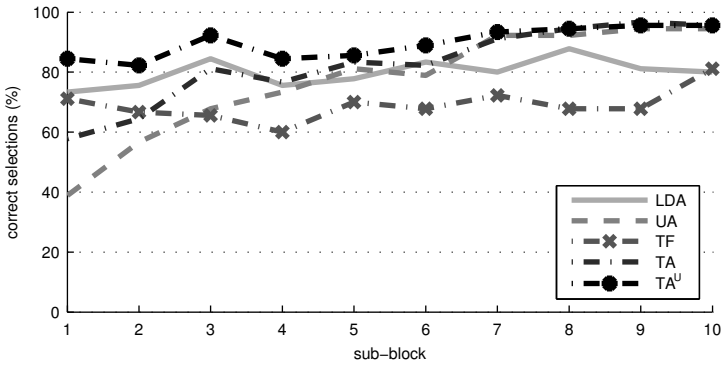


Figure 6.3: Performance comparison between a supervised LDA classifier, an unsupervised adaptive model UA, a fixed transfer model TF, an adaptive transfer model TA and the re-evaluation of the adaptive transfer model TA^U. The results are averaged averaged over 9 subjects and presented per sub-block of 10 trials. The nine subjects used in this analysis are those that needed at least 100 trials to spell the selected text, i.e. these are the more difficult subjects.

are offline results after adaptation to the entire test set. Surprisingly, the TF approach works remarkably well and UA is only slightly worse than LDA. TA produces the best online result and outperforms LDA. Data from nine harder subjects, who needed at least 100 trials to spell the requested text with the LDA method, was selected to study the temporal dynamics of all classifiers. Figure 6.3A gives a comparison of all methods and shows how their performance changes over blocks of 10 trials. While the LDA classifier and the TF were static, the UA and TA methods were adapted once per trial. The adaptation strategy leads to clear improvements for later trials. The classifier TA^U is the only classifier, for whom no simulated online performance was plotted. It had seen and learned from the complete amount of 100 trials, before re-analysing them. The small improvement even by the two static classifiers (LDA, TF) and by TA^U indicates that the subjects underwent a minor learning process during the experiment. An effect that had been reported by subjects themselves too. This fully corroborates the results obtained in the previous chapter.

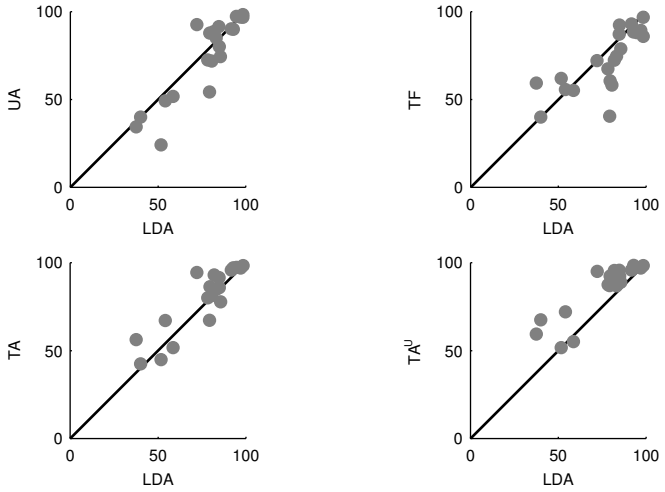


Figure 6.4: Subject-wise performance comparison on the selection accuracy between a supervised LDA classifier and either an unsupervised adaptive model UA, a fixed transfer model TF, an adaptive transfer model TA or the re-evaluation of the adaptive transfer model TA^U. Each dot represents a single subject.

True differential activation patterns

Attended target stimuli are known to elicit different ERP responses on average than un-attended non-target stimuli. We focus on the distribution of class-discriminative information, Figure 6.1 displays the differential activation patterns (mean target activity minus mean non-target activity) directly. Blue patches indicate areas on the scalp surface, which show a lower activity for targets than for non-targets and vice versa for red patches, while green colour indicates non-informative areas. The top row shows the activation pattern averaged over all 21 subjects. The second and third row depict patterns from two selected subjects, *faw* and *fcb*. Despite these large differences transfer learning is still possible between these subjects.

Recovered differential activation patterns

When we investigate the temporal evolution of the pattern associated with the transfer learning approach for subject *faw*, which is shown

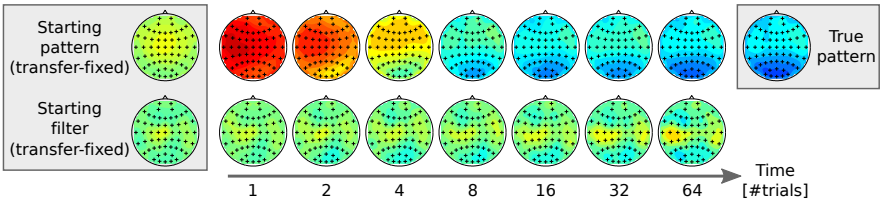


Figure 6.5: We present the evolution of the inferred patterns and filter for subject *faw* at 300 ms post-stimulus during the TA simulation. We start with the original pattern and filter from the general model. Subsequently, we show the patterns after the first 1, 2,4, 8, 16, 32, and 64 trials. The true activation pattern is given on the right. it is clear that after only 8 trials, the transfer learning based model is able to recover the correct neurophysiological activity.

in Figure 6.5, we see that the transfer learning model expects a positivity around 300 ms post stimulus and not the observed negativity. Over the course of the subsequent trials, during which the classifier processes subject-specific data and adapts to it, the inferred patterns change drastically. The changes to the classifier, i.e. the filter, on the other hand, are quite subtle. Nevertheless, after no more than 8 trials, the learning method has recovered most of the underlying structure in the true activation patterns. On top of that, the classifier was actually able to predict the desired stimulus right from the start.

Determining the dimensionality of the learning problem

BCI data is noisy, non-stationary, subject-specific and, with 550 ERP features per stimulus, very high-dimensional. As a result, one would assume that unsupervised learning is not feasible, even when the paradigm constraints are limiting the solution space. Furthermore, the high inter subject variability appears to be prohibitive for transfer learning. But remarkably, unsupervised learning is able to outperform a subject-specific supervised trained classifier and transfer learning is possible between a pair of subjects even when they exhibit significantly different ERP responses. To investigate why unsupervised learning is possible, we will first investigate where the difficulty in ERP decoding lies. This will be followed by an analysis of the

hidden similarities between subjects that explain the success of the transfer learning approach.

To find out why unsupervised learning is possible we begin by estimating the task relevant dimensionality via the RDE algorithm proposed in (Braun et al., 2008; Montavon et al., 2013). On average, the RDE estimate is 126, and thus significantly lower than the 550 dimensional feature space. However, inspection of the kPCA coefficients for subject *fcb* in Figure 6.6(middle plot) shows that the kPCA components decay slowly. This indicates a very high noise level in the dataset. Consequently, the RDE might over-estimate the actual relevant dimensionality. Luckily, we can reduce the noise by exploiting the ERP paradigm's constraints, just like in the unsupervised model. Within one iteration of 6 stimuli, only one can result in a target ERP response. Consequently, we can average classifier outputs per stimulus within a trial. Note that this does not require label information and that this is done explicitly in some form by all machine learning approaches to ERP spelling, including the unsupervised model. After having reduced the dataset by performing per-stimulus averaging within each trial, we have re-estimated the RDE. Inspection of the kPCA coefficients for subject *fcb* in Figure 6.6 (middle plot) reveals that the dimensionality as well as the noise level are significantly reduced. In Figure 6.6 (right plot), we compare the subject-specific RDE estimates with and without constraints. We obtain an average RDE of 54 using the constraints, which represents a reduction by a factor of 10 with respect to the original 550 dimensional feature space. Now we can relate this to the mean number of trials required by the UA classifier to achieve a performance similar to LDA. In Figure 6.3 we see that UA requires about 30-40 trials (6 per trial), which equals to 180-240 averaged feature vectors. This number is roughly 5 times the estimated dimensionality of the data.

Understanding transfer learning

Next, we argue that the task relevant subspace computed for a specific subject retains class-discriminative information for the other subjects, even when there is little or no similarity between their ERP responses. Additionally, we hypothesise that the direction within this subspace

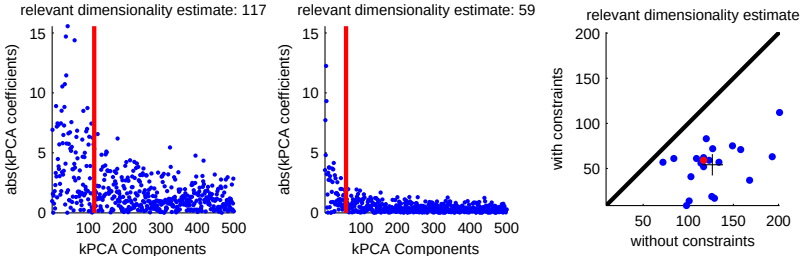


Figure 6.6: Comparison of kPCA coefficients (sorted by decreasing eigenvalue) and RDE estimates (red) for subject *fcb*. In the left plot, we did not perform the constraint based averaging. RDE indicates a rather noisy dataset. However, in the middle plot, we used the constraints to average the data per stimuli within each trial. The faster decaying coefficients indicate that the noise level is much reduced. As a result the estimate of the relevant dimensionality is much lower. Finally, in the rightmost plot, wee compare the RDE estimates for all subjects, with and without averaging. A red dot marks subject *fcb*, and a black cross marks the average.

in which the class discriminative information lies is also preserved across subjects. To empirically verify this hypothesis, we will re-use the subject-specific discriminative subspaces that we have obtained in the previous experiment. Using these task relevant subspaces, we will verify whether there is class discriminative information retained for other subjects and whether the class-discriminative manifold in this subspace can be shared across subjects.

To obtain the relevant subspaces, we compute the kPCA directions $\boldsymbol{\nu}_m$ and the kPCA coefficients z_m on the dataset of the *from-subject*. This allows us to estimate the RDE value d. Note that projecting the data to this discriminative subspace corresponds to Figure 6.2 C. For the *from-subject*, the task-discriminative information in this subspace is situated along the direction defined by the corresponding kPCA coefficients. Transfer learning between the *from-subject* and a novel subject, the so-called *to-subject*, is possible when kPCA coefficients of the *from-subject* can be used to label the projected data of the

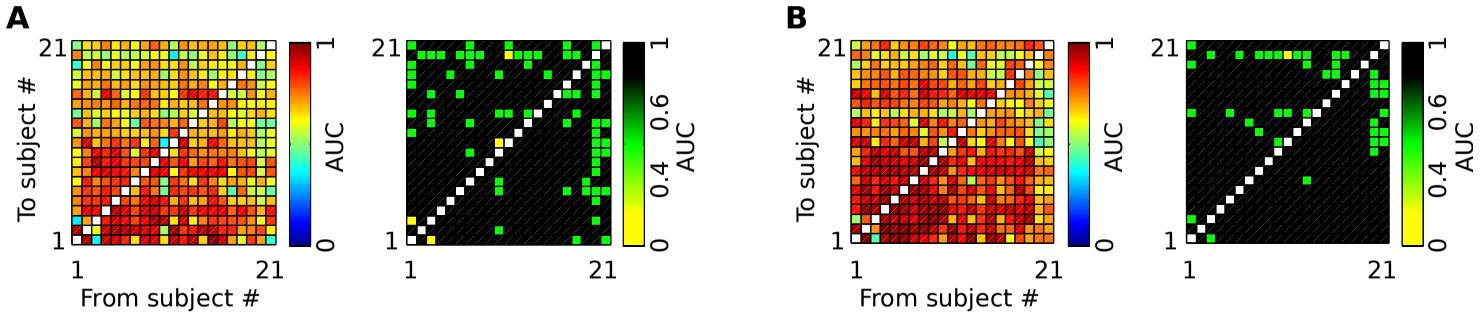


Figure 6.7: Subject to subject transfer learning results. Note that the subjects are sorted in order of decreasing online performance in the original study. To obtain these results, we computed a classifier using the data of the *from-subject* and applied it to the *to-subject*. Performance is measured in AUC. Furthermore, per panel, both plots give the same AUC values, where the first plot gives a fine-grained representation and the second plot give a coarse grained view. Panel A: For each subject, we computed the relevant dimensions and a classifier within these dimensions (as discussed in the toy example Figure 6.2. Panel B: Here, we trained an unsupervised classifier per subject and applied it to all the other subjects. It is clear that both approaches result in successful transfer learning between nearly all possible subject pairs. This shows explicitly that the discriminative subspaces can be shared between the different subjects.

to-subject:

$$\hat{y}\left(\hat{\boldsymbol{x}}\right) = \sum_{m=1}^{d} z_m f\left(\hat{\boldsymbol{x}}\right)_m ,$$

or to put it in other words, when the subjects are similar in the discriminative subspace.

The results from this experiment are presented in Figure 6.7A. We plot the AUC values for all possible combinations of *from-* and *to-subjects*. Interestingly, this approach results in a successful transfer (AUC > 0.5) in 95.7% of the cases. We get an AUC of at least 0.6 in 52.6 % and of 0.8 in 28.3 % of the transfer combinations. This gives a clear indication that the discriminative subspaces can effectively be shared between (nearly) all possible subject pairs, even when their differential activation patterns are very distinct. An example of such a case are the subjects *faw* and *fcb*. Their dissimilar activation patterns are shown in Figure 6.1). In spite of the clear mismatch between these activation patterns, transfer learning between the subjects is quite successful and results in an AUC of at least 0.76 in either direction. Similar to our initial toy example, we have shown that even though the recorded features are diverse, task relevant information can effectively be shared across subjects.

Furthermore, as we have discussed previously, the classifiers used for this experiment $\hat{y}\left(\hat{\boldsymbol{x}}\right) = \sum_{m=1}^{d} z_m f\left(\hat{\boldsymbol{x}}\right)_m$ can be reduced to linear classifiers working on the original feature space. These linear classifiers are similar to the ones used in the TF and TA models. Consequently, this experiment provides an increased understanding of the underlying mechanisms of transfer learning in this high dimensional and noisy setting. Furthermore, an analogous experiment where we share the unsupervised trained subject-specific classifiers corroborates our previous findings. Figure 6.7B shows that the individual subject-specific classifiers from the UA method can be successfully transferred to the almost all other subjects.

Furthermore, the fact that for nearly all *to-subjects* class-relevant information is retained in the relevant subspace of the *from-subject* and that the information itself lies along the same direction within this subspace is surprising. This contrasts with recent findings on motor imagery based BCI, where it is shown that finding a common

informative subspace across subjects is not feasible, while finding a common noise subspace is (Samek et al., 2013).

6.5 Conclusion

In this chapter, we have given empirical evidence which supports that subject-specific, task relevant subspaces retain much discriminative information for other subjects and that the direction in which the discriminative information lies is similar across subjects, even when their ERP responses are substantially different. This provides an answer as to why the (unsupervised) transfer learning approach is successful, because classifying a feature vector in the task-relevant subspace can be reduced to linear classification in the original feature space.

The concept of transfer learning can be applied to a wide array of modalities and applications. The key message is that high dimensional datasets with a large amount of inter-subject variability might be more similar than one would expect after analysing the data. The high dimensional feature space potentially encompasses hidden structure that is shared across subjects. Hence, by uncovering this hidden structure, we can further our understanding of the observations at hand. Furthermore, utilising transfer learning allows us to build improved and more data efficient models.

7 Conclusions and future perspectives

Since the late eighties, the BCI community has put a lot of effort into improving the decoding speed and accuracy of ERP based BCI. On top of that, the community has invested many resources in reducing the need for labelled data. Even though significant gains have been made, true online zero training ERP-BCI, which implies that the BCI can be used with high accuracy without a user specific calibration recording, was not yet possible. In this thesis, I have documented the development and evaluation of a novel machine learning framework for ERP based BCI decoding. This framework supports the integration of transfer learning, language models, dynamic stopping and a reliable unsupervised learning algorithm. By combining these different techniques, we have made true zero-training for ERP-BCI a reality.

7.1 Research conclusions

7.1.1 A unified model for ERP-BCI

Machine learning based decoders for ERP-BCI have become increasingly more convoluted. Originally, the brain signals were decoded by a simple linear classifier. Nowadays, a basic classifier is combined with language models to increase the reliability, and with dynamic stopping strategies to speed up the decision making process. Unfortunately, in most models, the basic classifier, the language model and the dynamic stopping strategy are isolated subsystems. To cou-

ple these subsystems, the use of heuristics is required. Furthermore, these heuristics add extra tunable parameters to the model. These additional parameters have to be optimised using labelled data.

To simplify both the model and the optimisation process, we have proposed a unified probabilistic model that integrates language models and the classifier in a coherent manner. As an added bonus, the probabilistic output of this classifier gives way to a natural dynamic stopping strategy based on the likelihood of the predicted symbol. Additionally, we have shown that the parameters of this model can be optimised without complicated cross-validation procedures. The end result is that we have built a state of the art decoder, which can be optimised easily on a limited labelled dataset.

However, this is not the main contribution of the proposed model, it has to be seen as a basic building block that can be extended to build an even more powerful model. Our two main extensions are the unsupervised learning rule and the incorporation of transfer learning.

7.1.2 Unsupervised learning is a valid training strategy

Up to now, calibration of an ERP-BCI always required labelled data, which is recorded during a tedious calibration session. To eliminate the need for this time-consuming process, we have proposed a novel unsupervised learning algorithm for ERP-BCI. This algorithm is able to learn how to decode the brain signals without seeing a single labelled datapoint. The key to this unsupervised learning method is the exploitation of constraints posed by the ERP paradigm. In an ERP paradigm, each trial has a single target stimulus. This target stimulus must elicit a target ERP response. The other stimuli must evoke a non-target response. This limitation significantly reduces the number of possible labellings of the data, which simplifies the decoding problem drastically. As a result, an Expectation Maximisation based optimisation of the classifier is possible.

We have evaluated the proposed unsupervised model extensively, both offline and online. Our offline evaluation comprises 25 subjects from three datasets with visual ERP data. This evaluation has shown that when enough unlabelled data is available, the unsuper-

vised method can build a model that is as accurate as state of the art supervised classifiers. However, while this approach does not require any labelled data point, it does need a sizeable amount of unlabelled data before it is able to build this highly accurate model.

In our online simulations, where at the beginning of each experiment the model is initialised randomly, this dependence on a substantial amount of data is noticeable in the form of a warm-up period. During this warm-up, the classifier is not reliable and makes decoding mistakes. However, this inherently adaptive method can correct these mistakes when the model has improved. The model quality is increased throughout the experiment as more and more unlabelled data becomes available.

The biggest test for a novel decoding approach for BCI is a true online experiment. To put the novel algorithm to the test, we have conducted an online study using a challenging auditory ERP paradigm: AMUSE. Ten subjects took part in this online BCI experiment. This demanding experiment demonstrated that it is truly possible to build a reliable unsupervised decoder in the time normally used for calibration. Furthermore, in contrast to a typical calibration recording, the unsupervised approach allows the user to effectively communicate during the entire session by using the revised predictions after the warm-up period.

A great challenge for BCI decoders is the non-stationarity of EEG data, which can be caused by external influences such as fatigue. Our experimental results indicate that the unsupervised adaptive approach is robust to changes in the EEG features. The result is that while the performance of a supervised classifier deteriorates by the end of a longer session, the unsupervised model is still able to present the user with highly accurate predictions.

7.1.3 Zero-training is a reality for ERP based BCI

To build a true zero-training BCI, it is not sufficient to merely eliminate the need for labelled data. A true zero-training BCI should not have a warm-up period. Hence, to eliminate the warm-up effect, we have extended the unified model to incorporate inter-subject transfer learning. By allowing the model to share decoding knowledge between

users, we can build a general subject-unspecific model of moderate accuracy. But, by combining this transfer learning methodology with unsupervised adaptation, we can transform the general model quickly to a subject-specific model of high quality without labelled data. Furthermore, this transfer learning based model can be further improved with dynamic stopping and language models.

This transfer learning method has been evaluated offline on a dataset comprising 22 subject that participated in a visual ERP-BCI experiment. On this dataset, our results show that unsupervised transfer learning can compete directly with a state of the art supervised decoder. However, the major advantage is that, when combined with dynamic stopping, the transfer learning model allows a user to spell 29 symbols in the time normally spent on calibration. This vast improvement in efficiency is invaluable for patient applications.

To verify the applicability of transfer learning in an online setting, we have tested the unsupervised adaptive transfer model on the AMUSE paradigm. For this evaluation, 9 subjects from the original unsupervised study returned and we invited 11 novel subjects too. The results from this experiment indicate that true zero-training BCI is indeed possible. Obviously, the next step is to bring this promising approach to the end users, where it can substantially improve the usability of BCI.

7.1.4 A better understanding of unsupervised transfer learning in ERP based BCI

While transfer learning has shown great promise, it is not an intuitive approach to BCI decoding. BCI data exhibits a significant amount of inter-subject variability. Moreover, the BCI community has a tradition of using subject-specific supervised classifiers. This makes it all the more surprising that transfer learning can actually work. To provide a better understanding of the fundamental concept behind transfer learning we have investigated this in more detail.

Our analysis has shown that, while the data from different subjects appears to be fundamentally different, this difference is actually superficial. The features commonly used in ERP decoding can be incredibly high dimensional, e.g. we have used feature vectors of over

640 dimension. However, the relevant information content is contained in a much smaller subspace. Furthermore, within this informative subspace, the subjects are actually rather similar and decoding knowledge can be shared across users. Finally, we have shown that projecting the data into the informative subspace and classification in this subspace can be reduced to a single linear operation on the original feature space. Naturally, this result gives additional support for the proposed transfer learning model.

7.2 Future directions

The unsupervised transfer learning method developed in this thesis is tailored to ERP based BCI. Nevertheless, the principles and concepts, such as constraint based unsupervised learning and transfer learning, can be applied in a much wider context. Before discussing the opportunities outside the field of BCI, I would like to discuss how our finding can shape future BCI research.

7.2.1 The synergy between man and machine

To build a true zero-training BCI, a paradigm with a shallow learning curve is required on top of a reliable machine learning decoder that works instantly. We have shown that when the users are instructed properly, the unsupervised transfer learning model does result in a true zero-training BCI. This was possible thanks to the combination of a paradigm that provided the machine learning model with a big chunk of side information to simplify the decoding process. It is the paradigm's structure that makes unsupervised (transfer) learning possible.

However, the BCI community has always developed novel paradigms independently from the machine learning decoders. Only after the paradigm is finalised, can a machine learning researcher attempt to improve the decoding performance or to reduce the need for labelled data. Given our promising results on unsupervised decoding, we argue that by taking the interaction between the user, the machine learning algorithm and the paradigm into account, we will be able to develop a

new generation of highly efficient true zero-training BCIs. Obviously, we must remain focussed on building a user-friendly system as well.

7.2.2 The need for adaptation

One of the key assumptions in (supervised) machine learning is that the data that is used to train the model has the same underlying distribution as the test data. This assumption is often violated, especially in applications where the machine learning method has to interact directly with its environment.

In BCI, too, non-stationarity, e.g. due to fatigue, can reduce the performance of a static classifier. An adaptive classifier can detect these changes and incorporate them into the model. Therefore, we argue that the future of BCI decoding lies in the use of adaptive models that will make frequent re-calibration no longer required. This will not only increase the accuracy but can increase the efficiency as well.

Furthermore, this is not the only opportunity for advanced machine learning methods to help the field. When learning is involved, there is co-adaptation between the user and the machine learning algorithm. Hence, it is extremely interesting to investigate the mechanisms underlying this learning process during the interaction between man and machine. We believe that an analysis of the adaptation in the machine learning models, can provide additional insights in the processes that govern human learning.

Finally, we believe that constraint-based learning will become a key concept in building highly adaptive and accurate machine learning models for these difficult problems.

7.2.3 Constraint-based learning

Big data is one of the current buzzwords in data analysis. But, we should not forget that labelling big datasets will eventually become infeasible. Moreover, for some problems labelling small datasets is already expensive, e.g. datasets for epileptic seizure detection have to be labelled by medical professionals.

Unsupervised algorithms do not require these labelled datasets .

Unfortunately, they are currently used for finding structure in the data, but not to solve a problem directly. Hence, after the data is processed by a common unsupervised algorithm, it has to be interpreted by a human or evaluated by a supervised model. As a result, labelled data is still required.

It has been shown in the context of natural language processing and in this thesis that constraints can enable us to learn very efficiently. Therefore, a major opportunity in increasing the capabilities of data processing models lies in the investigation and development of novel constraint-based learning algorithms that can cope with the endless stream of data that is generated today.

This investigation will have to take place on the theoretical level to find out how these constraints provide us with information and what the best approach is to encode them in a machine learning model. A very interesting, but ambitious, path is to investigate whether the automatic discovery of constraints and applicable models is possible. To facilitate this, we will potentially have to rely on a combination of transfer and unsupervised learning.

7.2.4 Combining transfer learning and unsupervised learning

Learning form scratch without labelled data is a herculean task. Hence, we believe that to process large unlabelled datasets, (multi-modal) transfer learning will become inevitable. Transfer learning allows us to share knowledge between related tasks. By relying on this "experience", the learning process is simplified significantly. However, even the state of the art zero-shot and one-shot learning models still require an ample amount of labelled data to train the model. To alleviate this, I strongly believe in the combination of (multi-modal) transfer learning and constraint based unsupervised learning to build a novel generation of models that can solve tasks directly with a limited amount of prior information. Such an approach will allow us to harness the ever increasing flood of data that is being recorded.

A

Unsupervised (transfer) learning update equations

This part of the appendix considers the derivation of the update equations of our unsupervised learning algorithm. We will present the derivation only in the more general case of transfer learning. The basic unsupervised update equations can be obtained by setting the mean of the prior on $\boldsymbol{w}$ equal to zero.

The update equations are derived using the EM algorithm, where $\boldsymbol{w}$ and β are the parameters, the attended stimuli $\boldsymbol{c}$ are latent. The Additionally, α can be optimised by direct maximum likelihood, when we assume that $\boldsymbol{w}$ is observed.

The transfer model, without language models and where we dropped the subject specific identifier s, is defined as follows.

$$\begin{aligned} p\left(\boldsymbol{\mu}_{\boldsymbol{w}}\right) &= \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{w}}|0, \alpha_p^{-1} I\right) \\ \alpha_p &= 0 \\ p\left(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}\right) &= \mathcal{N}\left(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \alpha^{-1} I\right) \\ p\left(c_t\right) &= \frac{1}{C} \\ p\left(\boldsymbol{x}_{t,i}|c_t, \boldsymbol{w}, \beta\right) &= \mathcal{N}\left(\boldsymbol{x}_{t,i}^T \boldsymbol{w}_s | y_{t,i}(c_t), \beta^{-1}\right) \end{aligned}$$

Consequently, we have to optimise the following expected data log likelihood with respect to $\hat{\boldsymbol{w}}, \hat{\beta}$ using the EM algorithm, where $\boldsymbol{w}, \beta$ denote our current parameter estimates.

$$\sum_{\boldsymbol{c}} p\left(\boldsymbol{c}|X, \boldsymbol{w}, \beta\right) \log p\left(X|\boldsymbol{c}, \hat{\boldsymbol{w}}, \hat{\beta}\right) p\left(\boldsymbol{c}\right) + \log p\left(\hat{\boldsymbol{w}}|\boldsymbol{\mu}_{\boldsymbol{w}}, \alpha I\right)$$

Because $\hat{\beta}, \hat{\boldsymbol{w}}$ do not depend on the prior distribution on $\boldsymbol{c}$, we can drop that term and further simplify the equation that has to be optimised to:

$$\sum_t \sum_{c_t} p\left(c_t | X, \boldsymbol{w}, \beta\right) \log p\left(X_t | c_t, \hat{\boldsymbol{w}}, \hat{\beta}\right) + \log p\left(\hat{\boldsymbol{w}} | \boldsymbol{\mu}_{\boldsymbol{w}}, \alpha I\right)$$

To optimise this equation, we have to compute the derivative or gradient with respect to the parameter we are interested in, setting the resulting equation equal to zero and solving for the desired parameter. We will deal with the parameters one by one. Furthermore, please note that we have written the conditioning on all the data, not on the data from a single trial. As a result, the derivation remains valid when we utilise language models.

Updating $\boldsymbol{w}$

The gradient with respect to $\hat{\boldsymbol{w}}$ can be written as follows. Where we assume that $\hat{\alpha} = \alpha$ and $\hat{\beta} = \beta$ are set equal to our last estimates.

$$-\sum_t \sum_{c_t} p\left(c_t | X, \boldsymbol{w}, \beta\right) \beta \left(X_t X_t^T \hat{\boldsymbol{w}} - X_t \boldsymbol{y}\left(c_t\right)\right) - \alpha I \left(\hat{\boldsymbol{w}} - \mu_{\boldsymbol{w}}\right)$$

By using the fact that $X_t X_t^T$ does not depend on c_t, we can pull this part outside of the sum.

$$-\beta \left(X X^T + \frac{\alpha}{\beta} I\right) \hat{\boldsymbol{w}} + \sum_t \sum_{c_t} p\left(c_t | X, \boldsymbol{w}, \beta\right) \beta X_t \boldsymbol{y}\left(c_t\right) + \alpha I \mu_{\boldsymbol{w}}$$

Where it becomes clear that the update equation will be closely related to regularised regression. Setting this equal to 0 and this for $\hat{\boldsymbol{w}}$ yields the following update equation.

$$\hat{\boldsymbol{w}} = \left(X X^T + \frac{\alpha}{\beta} I\right)^{-1} \sum_t \sum_{c_t} p\left(c_t | X, \boldsymbol{w}, \beta\right) \left(X_t \boldsymbol{y}\left(c_t\right) + \frac{\alpha}{\beta} I \mu_{\boldsymbol{w}}\right)$$

This can be written slightly more compact, but less straightforward to implement, by combining the sums.

$$\hat{\boldsymbol{w}} = \left(XX^T + \frac{\alpha}{\beta}I\right)^{-1} \sum_{\boldsymbol{c}} p\left(\boldsymbol{c}|X, \boldsymbol{w}, \beta\right) \left(X\boldsymbol{y}\left(\boldsymbol{c}\right) + \frac{\alpha}{\beta}I\mu_{\boldsymbol{w}}\right)$$

This is a sum of regression classifiers, weighted by the probability that the target labels are correct given the previous estimate of $\boldsymbol{w}$ and regularised towards $\boldsymbol{\mu}_{\boldsymbol{w}}$.

Updating β

To update β, we have to take the derivative with respect to $\hat{\beta}$ in

$$\sum_t \sum_{c_t} p\left(c_t|X, \boldsymbol{w}, \beta\right) \log p\left(X_t|c_t, \hat{\boldsymbol{w}}, \hat{\beta}\right)$$

Since these depend on the individual stimuli i, it makes sense to rewrite this to:

$$\sum_t \sum_{c_t} p\left(c_t|X, \boldsymbol{w}, \beta\right) \sum_i \log p\left(\boldsymbol{x}_{t,i}|c_t, \hat{\boldsymbol{w}}, \hat{\beta}\right)$$

The derivative of this with respect to $\hat{\beta}$ is

$$\sum_t \sum_{c_t} p\left(c_t|X, \boldsymbol{w}, \beta\right) \frac{1}{2} \sum_i \left(\frac{1}{\hat{\beta}} - \left(\boldsymbol{x}_{t,i} - y\left(c_n\right)\right)^2\right)$$

From which we find the optimum, which we have written as the variance, as the expected mean squared error between are target projections and the true projection.

$$\hat{\beta}^{-1} = \left\langle \sum_{c_{s,t}} p\left(c_t|X, \boldsymbol{w}, \beta\right) \left(\boldsymbol{x}_{t,i}^T \boldsymbol{w} - y_{t,i}(c_t)\right)^2 \right\rangle_{t,i}$$

Updating α

The update for α is a straightforward optimisation of the likelihood when $\boldsymbol{w}$ is observed and $\boldsymbol{\mu}_w$ is known. We have to derive $\log p(\boldsymbol{w}|\alpha)$, set it equal to 0 and solve for α.

$$\begin{aligned} 0 &= \frac{\partial}{\partial \alpha} \log \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \alpha) \\ &= \frac{\partial}{\partial \alpha} \left(\log \frac{1}{|\alpha^{-1} I|^{\frac{1}{2}}} - \frac{\alpha}{2} (\boldsymbol{w} - \boldsymbol{\mu}_w)^T (\boldsymbol{w} - \boldsymbol{\mu}_w) \right) \\ &= \frac{D}{2} \frac{1}{\alpha} - \frac{1}{2} (\boldsymbol{w} - \boldsymbol{\mu}_w)^T (\boldsymbol{w} - \boldsymbol{\mu}_w) \\ \alpha^{-1} &= \frac{D}{(\boldsymbol{w} - \boldsymbol{\mu}_w)^T (\boldsymbol{w} - \boldsymbol{\mu}_w)} \end{aligned}$$

In this final equation, D is the dimensionality of the weight vector $\boldsymbol{w}$.

B

Constructing the transfer learning toy example

To construct the toy example in Figure: 6.2 in Chapter 6, we used 29 source channels that are all independent and follow a normal distribution. The first two channels have a class dependent mean, the remaining 27 have mean equal to 0. We write these source channels as a function of the measured features $\boldsymbol{x}$ to keep notation identical to the discussion in Chapter 6.

$$f(\boldsymbol{x}) = [f(\boldsymbol{x})_1, \ldots, f(\boldsymbol{x})_{29}]^T.$$

The measured features are generated from the source channels by using a subject specific linear transform Λ. Hence, for subject 1 the recorded features can be computed as follows:

$$\boldsymbol{x} = \Lambda_1 f(\boldsymbol{x}).$$

To project back into the original source space, we have to compute the backwards model, which is equal to the inverse of the forwards model:

$$\Delta_1 = \Lambda_1^{-1}.$$

As a consequence, the mapping $f(\boldsymbol{x})$ for subject 1 equals.

$$f(\boldsymbol{x}) = \Delta_1 \boldsymbol{x}.$$

To discriminate between target and non-target trials, we only require the first two source dimension $f(\boldsymbol{x})_1$ and $f(\boldsymbol{x})_2$ since the remaining

channels contain only noise. The corresponding linear transformations are $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2$. These are the first two rows of Δ_1. Because we assume that the two informative source channels are shared between the users, the first two rows of Δ_2 are identical to those of Δ_1. Therefore, we generated the backward model Δ_2 for the second subject by fixing the first to rows to $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2$. The remaining rows are initialised randomly. Given the backward model of the second subject, we can compute its corresponding forward model:

$$\Lambda_2 = \Delta_2^{-1}.$$

To generate the data, we generate random vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ for both subjects and multiply them by their respective forward models. The result is the data we have shown in Figure 6.2. Of course, we constructed the toy example such that it adheres to our transfer learning assumption. It was conceived to illustrate the transfer learning hypothesis and allowed us to demonstrate that a lot of inter subject and inter trial variability can be present even though the informative source signals are located within the same subspace. We do not claim that the generative model of the toy example is a perfect model for how the informative brain signals propagate. It is, however, a starting point to investigate what the underlying shared mechanisms are and how we can exploit them in future work to build even better transfer learning models.

C Datasets

In this part of the appendix, we will present specific details on the datasets used in this work.

BCI Competition II

Paradigm: 6 by 6 matrix speller recorded using BCI2000

Subjects: 1

EEG recording: 64 Channels at 240 Hz

Trainset: 42 trials, 15 iterations

Testset: 31 trials, 15 iterations

Inter Stimulus Interval: 75 ms

Stimulus Duration: 100 ms

Pause between trials: 5 s

Reference: Blankertz et al. (2004)

BCI Competition III

Paradigm: 6 by 6 matrix speller recorded using BCI2000

Subjects: 2

EEG recording: 64 channels at 240 Hz

Trainset: 85 trials, 15 iterations

Testset: 100 trials, 15 iterations

Inter Stimulus Interval: 75 ms

Stimulus Duration: 100 ms

Pause between trials: 5 s

Reference: Blankertz et al. (2006b)

Akimpech

Paradigm: 6 by 6 matrix speller recorded using BCI2000

Subjects: 22

EEG recording: g.USB amp, 10 channels at 256 Hz

Trainset: 16 trials, 15 iterations

Testset: 17-29 trials (22.18 on average), 15 iterations

Inter Stimulus Interval: 125 ms

Stimulus Duration: 62.5 ms

Pause between trials: 4 seconds

Reference: Yanez-Suarez et al. (2012)

Original Amuse study

Paradigm: AMUSE (auditory ERP)

Subjects: 21

EEG recording: 56 channels, 1 kHz

Trainset: 16 trials, 15 iterations

Testset: 64-165 trials, 15 iterations

Inter Stimulus Interval: 135 ms

Stimulus Duration: 40 ms

Pause between trials: unknown

Reference: Schreuder et al. (2011b)

Bibliography

Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. *Advances in Neural Information Processing Systems*, 19:1.

Acqualagna, L. and Blankertz, B. (2013). Gaze-independent BCI-spelling using rapid visual serial presentation (RSVP). *Clinical Neurophysiology*, 124(5):901–908.

Ang, K. K., Chin, Z. Y., Wang, C., Guan, C., and Zhang, H. (2012). Filter bank common spatial pattern algorithm on bci competition iv datasets 2a and 2b. *Frontiers in Neuroscience*, 6.

Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

Berger, H. (1929). Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570.

Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., Perelmouter, J., Taub, E., and Flor, H. (1999). A spelling device for the paralysed. *Nature*, 398(6725):297–298.

Bird, S. (2006). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition.

Bishop, W., Chestek, C. C., Gilja, V., Nuyujukian, P., Foster, J. D., Ryu, S. I., Shenoy, K. V., and Yu, B. M. (2014). Self-recalibrating classifiers for intracortical brain–computer interfaces. *Journal of Neural Engineering*, 11(2):026001.

Blankertz, B., Dornhege, G., Krauledat, M., Müller, K.-R., and Curio, G. (2007). The non-invasive berlin brain–computer interface: fast acquisition of effective performance in untrained subjects. *Neuroimage*, 37(2):539–550.

Blankertz, B., Dornhege, G., Krauledat, M., Schröder, M., Williamson, J., Murray-Smith, R., and Müller, K.-R. (2006a). The berlin brain-computer interface presents the novel mental typewriter hex-o-spell. In *"Proceedings of the 3rd International Brain-Computer Interface Workshop and Training Course 2006"*.

Blankertz, B., Lemm, S., Treder, M., Haufe, S., and Müller, K.-R. (2011). Single-trial analysis and classification of ERP components, a tutorial. *Neuroimage*, 56(2):814 – 825.

Blankertz, B., Müller, K.-R., Curio, G., Vaughan, T., Schalk, G., Wolpaw, J., Schlögl, A., Neuper, C., Pfurtscheller, G., Hinterberger, T., Schröder, M., and Birbaumer, N. (2004). The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials. *Biomedical Engineering, IEEE Transactions on*, 51(6):1044 –1051.

Blankertz, B., Müller, K.-R., Krusienski, D., Schalk, G., Wolpaw, J., Schlögl, A., Pfurtscheller, G., Millan, J., Schröder, M., and Birbaumer, N. (2006b). The BCI competition III: validating alternative approaches to actual BCI problems. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 14(2):153 –159.

Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., and Muller, K.-R. (2008). Optimizing spatial filters for robust eeg single-trial analysis. *Signal Processing Magazine, IEEE*, 25(1):41–56.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bonnet, L., Lotte, F., and Lécuyer, A. (2013). Two brains, one game: Design and evaluation of a multiuser bci video game based on motor imagery. *Computational Intelligence and AI in Games, IEEE Transactions on*, 5(2):185–198.

Braun, M. L., Buhmann, J., and Müller, K.-R. (2008). On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research*, 9:1875–1908.

Brouwer, A.-M., van Erp, J., Heylen, D., Jensen, O., and Poel, M. (2013). Effortless passive bcis for healthy users. In *Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion*, pages 615–622. Springer.

Brouwer, A.-M. and Van Erp, J. B. (2010). A tactile p300 brain-computer interface. *Frontiers in Neuroscience*, 4.

Cecotti, H. and Gräser, A. (2010). Convolutional neural networks for P300 detection with application to brain-computer interfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 99.

Chang, M.-W., Ratinov, L., and Roth, D. (2007). Guiding semi-supervision with constraint-driven learning. *Urbana*, 51:61801.

Chen, S. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Cheng, M., Gao, X., Gao, S., and Xu, D. (2002). Design and implementation of a brain-computer interface with high transfer rates. *Biomedical Engineering, IEEE Transactions on*, 49(10):1181–1186.

Cincotti, F. (2010). Tactile, visual, and bimodal p300s: Could bimodal p300s boost bci performance? *SRX Neuroscience*, 2010.

Coates, A. and Ng, A. Y. (2012). Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pages 561–580. Springer.

Colwell, K., Throckmorton, C., Collins, L., and Morton, K. (2013). Transfer learning for accelerated p300 speller training. In d. R. Millán, J., Gao, S., Müller-Putz, G. R., Wolpaw, J. R., and Huggins, J. E., editors, *Proceedings of the Fifth International Brain-Computer Interface Meeting 2013*, Graz. Verlag der Technischen Universität Graz.

Dähne, S., Höhne, J., and Tangermann, M. (2011). Adaptive classification improves control performance in ERP-based BCIs. In *Proceedings of the 5th International BCI Conference*, pages 92–95, Graz.

Dal Seno, B., Matteucci, M., and Mainardi, L. T. (2010). The utility metric: a novel method to assess the overall performance of discrete brain–computer interfaces. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 18(1):20–28.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38.

Devlaminck, D., Wyns, B., Grosse-Wentrup, M., Otte, G., and Santens, P. (2011). Multisubject learning for common spatial patterns in motor-imagery bci. *Computational intelligence and neuroscience*, 2011:8.

Dieleman, S., Brakel, P., and Schrauwen, B. (2011). Audio-based music classification with a pretrained convolutional network. In *12th International Society for Music Information Retrieval Conference (ISMIR-2011)*, pages 669–674. University of Miami.

Dornhege, G., del R. Millan, J., Hinterberger, T., McFarland, D. J., and Müller, K.-R. (2007). *Towards Brain-Computer Interfacing*. MIT Press.

Farquhar, J. and Hill, J. N. (2013). Interactions between preprocessing and classification methods for event-related-potential classification. *Neuroinformatics*, pages 1–18.

Farwell, L. A. and Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510 – 523.

Fazli, S., Danóczy, M., Schelldorfer, J., and Müller, K.-R. (2011). L1-penalized Linear Mixed-Effects Models for high dimensional data with application to BCI. *Neuroimage*, 56(4):2100 – 2108.

Fazli, S., Popescu, F., Danóczy, M., Blankertz, B., Müller, K.-R., and Grozea, C. (2009). Subject-independent mental state classification in single trials. *Neural Networks*, 22(9):1305–1312.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., et al. (2013). Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2121–2129.

Gopal, S., Yang, Y., Bai, B., and Niculescu-Mizil, A. (2012). Bayesian models for large-scale hierarchical classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2420–2428.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Haufe, S., Meinecke, F., Görgen, K., Dähne, S., Haynes, J.-D., Blankertz, B., and Bießmann, F. (2014). On the interpretation of weight vectors of linear models in multivariate neuroimaging. *Neuroimage*, 87:96–110.

Herweg, A., Kaufmann, T., and Kübler, A. (2013). Using generic models to improve tactile ERP-BCI performance of low aptitude users. In d. R. Millán, J., Gao, S., Müller-Putz, G. R., Wolpaw, J. R., and Huggins, J. E., editors, *Proceedings of the Fifth International Brain-Computer Interface Meeting 2013*. Verlag der Technischen Universität Graz.

Hill, N. J., Lal, T. N., Bierig, K., Birbaumer, N., and Schölkopf, B. (2004). An auditory paradigm for brain-computer interfaces. In *Advances in Neural Information Processing Systems (NIPS)*.

Hochberg, L. R., Bacher, D., Jarosiewicz, B., Masse, N. Y., Simeral, J. D., Vogel, J., Haddadin, S., Liu, J., Cash, S. S., van der Smagt, P., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375.

Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., and Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hoffmann, U., Vesin, J.-M., Ebrahimi, T., and Diserens, K. (2008). An efficient P300-based brain–computer interface for disabled subjects. *Journal of Neuroscience Methods*, 167(1):115 – 125.

Höhne, J., Krenzlin, K., Dähne, S., and Tangermann, M. (2012). Natural stimuli improve auditory BCIs with respect to ergonomics and performance. *Journal of Neural Engineering*, 9(4):045003.

Höhne, J., Schreuder, M., Blankertz, B., and Tangermann, M. (2010). Two-dimensional auditory p300 speller with predictive text system. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 4185–4188. IEEE.

Höhne, J., Schreuder, M., Blankertz, B., and Tangermann, M. (2011). A novel 9-class auditory ERP paradigm driving a predictive text entry system. *Frontiers in Neuroscience*, 5(99).

Jin, J., Sellers, E., Zhang, Y., Daly, I., Wang, X., and Cichocki, A. (2012). Whether generic model works for rapid ERP-based BCI calibration. *Journal of Neuroscience Methods*.

Johnson Jr, R. and Donchin, E. (1978). On how p300 amplitude varies with the utility of the eliciting stimuli. *Electroencephalography and Clinical Neurophysiology*, 44(4):424–437.

Jolliffe, I. (2005). *Principal component analysis*. Wiley Online Library.

Kaufmann, T., Holz, E. M., and Kübler, A. (2013). Comparison of tactile, auditory, and visual modality for brain-computer interface use: a case study with a patient in the locked-in state. *Frontiers in Neuroscience*, 7.

Kaufmann, T., Schulz, S. M., Grünzinger, C., and Kübler, A. (2011). Flashing characters with famous faces improves ERP-based brain-computer interface performance. *Journal of Neural Engineering*, 8(5):056016.

Kaufmann, T., Vöker, S., Gunesch, L., and Kübler, A. (2012). Spelling is just a click away – a user-centered brain-computer interface including auto-calibration and predictive text entry. *Frontiers in Neuroscience*, 6(72).

Kemp, C., Perfors, A., and Tenenbaum, J. (2007). Learning overhypotheses with hierarchical bayesian models. *Developmental Science*, 10(3):307–321.

Kindermans, P.-J., Tangermann, M., Müller, K.-R., and Schrauwen, B. (2014). Integrating dynamic stopping, transfer learning and language models in an adaptive zero-training erp speller. *Journal of Neural Engineering*.

Kindermans, P.-J., Verschore, H., and Schrauwen, B. (2013). A unified probabilistic approach to improve spelling in an event-related potential based brain-computer interface. *Biomedical Engineering, IEEE Transactions on*, 60(10):2696–2705.

Kindermans, P.-J., Verschore, H., Verstraeten, D., and Schrauwen, B. (2012a). A P300 BCI for the masses: Prior information enables instant unsupervised spelling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 719–727.

Kindermans, P.-J., Verstraeten, D., Buteneers, P., and Schrauwen, B. (2011). How do you like your P300 speller : adaptive, accurate and simple? In *Proceedings of th 5th international Brain-Computer Interface Conference*.

Kindermans, P.-J., Verstraeten, D., and Schrauwen, B. (2012b). A Bayesian model for exploiting application constraints to enable unsupervised training of a P300-based BCI. *PLoS ONE*, 7(4):e33758.

Kleih, S., Nijboer, F., Halder, S., and Kübler, A. (2010). Motivation modulates the p300 amplitude during brain–computer interface use. *Clinical Neurophysiology*, 121(7):1023–1031.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Koles, Z. J., Lazar, M. S., and Zhou, S. Z. (1990). Spatial patterns underlying population differences in the background eeg. *Brain Topography*, 2(4):275–284.

Krauledat, M., Tangermann, M., Blankertz, B., and Müller, K.-R. (2008). Towards zero training for brain-computer interfacing. *PLoS ONE*, 3(8):e2967.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 1, page 4.

Kubanek, J., Miller, K., Ojemann, J., Wolpaw, J., and Schalk, G. (2009). Decoding flexion of individual fingers using electrocorticographic signals in humans. *Journal of Neural Engineering*, 6(6):066001.

LaFleur, K., Cassady, K., Doud, A., Shades, K., Rogin, E., and He, B. (2013). Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface. *Journal of Neural Engineering*, 10(4):046003.

Lal, T., Schröder, M., Hinterberger, T., Weston, J., Bogdan, M., Birbaumer, N., and Schölkopf, B. (2004). Support vector channel selection in BCI. *Biomedical Engineering, IEEE Transactions on*, 51(6):1003–1010.

Lécuyer, A., Lotte, F., Reilly, R. B., Leeb, R., Hirose, M., Slater, M., et al. (2008). Brain-computer interfaces, virtual reality, and videogames. *IEEE Computer*, 41(10):66–72.

Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88:365–411.

Lemm, S., Blankertz, B., Curio, G., and Müller, K.-R. (2005). Spatio-spectral filters for improving the classification of single trial eeg. *Biomedical Engineering, IEEE Transactions on*, 52(9):1541–1548.

Lenhardt, A., Kaper, M., and Ritter, H. (2008). An adaptive P300-based online brain-computer interface. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 16(2):121–130.

Li, K., Raju, V. N., Sankar, R., Arbel, Y., and Donchin, E. (2011). Advances and challenges in signal analysis for single trial p300-bci. In *Proceedings of the 6th International Conference on Foundations of Augmented Cognition: Directing the Future of Adaptive Systems*, FAC'11, pages 87–94, Berlin, Heidelberg. Springer-Verlag.

Li, Y., Guan, C., Li, H., and Chin, Z. (2008). A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system. *Pattern Recognition Letters*, 29(9):1285 – 1294.

Liang, P., Jordan, M. I., and Klein, D. (2009). Learning from measurements in exponential families. In *Proceedings of the 26th annual international conference on machine learning*, pages 641–648. ACM.

Liu, T., Goldberg, L., Gao, S., and Hong, B. (2010). An online brain–computer interface using non-flashing visual evoked potentials. *Journal of Neural Engineering*, 7(3):036003.

Lu, S., Guan, C., and Zhang, H. (2009). Unsupervised brain computer interface based on intersubject information and online adaptation. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 17(2):135 –145.

Luck, S. J. (2005). *An introduction to the event-related potential technique*. MIT press Cambridge, MA:.

Mak, J., Arbel, Y., Minett, J., McCane, L., Yuksel, B., Ryan, D., Thompson, D., Bianchi, L., and Erdogmus, D. (2011). Optimizing the p300-based brain–computer interface: current status, limitations and future directions. *Journal of Neural Engineering*, 8(2):025003.

Mann, G. S. and McCallum, A. (2008). Generalized expectation criteria for semi-supervised learning of conditional random fields.

Montavon, G., Braun, M. L., Krueger, T., and Müller, K.-R. (2013). Analyzing local structure in kernel-based learning: Explanation, complexity and reliability assessment. *Signal Processing Magazine, IEEE*, 30(4):62–74.

Müller, K.-R., Krauledat, M., Dornhege, G., Curio, G., and Blankertz, B. (2004). Machine learning techniques for brain-computer interfaces. *Biomed Tech*, 49(1):11–22.

Müller, K.-R., Tangermann, M., Dornhege, G., Krauledat, M., Curio, G., and Blankertz, B. (2008). Machine learning for real-time single-trial EEG-analysis: From brain-computer interfacing to mental state monitoring. *Journal of Neuroscience Methods*, 167(1):82–90.

Müller-Putz, G. R., Scherer, R., Brauneis, C., and Pfurtscheller, G. (2005). Steady-state visual evoked potential (ssvep)-based communication: impact of harmonic frequency components. *Journal of Neural Engineering*, 2(4):123.

Nijboer, F., Sellers, E., Mellinger, J., Jordan, M., Matuz, T., Furdea, A., Halder, S., Mochty, U., Krusienski, D., Vaughan, T., Wolpaw, J., Birbaumer, N., and Kübler, A. (2008). A P300-based brain-computer interface for people with amyotrophic lateral sclerosis. *Clinical Neurophysiology*, 119(8):1909 – 1916.

Nishimoto, S., Vu, A. T., Naselaris, T., Benjamini, Y., Yu, B., and Gallant, J. L. (2011). Reconstructing visual experiences from brain

activity evoked by natural movies. *Current Biology*, 21(19):1641–1646.

Nunez, P. L., Srinivasan, R., Westdorp, A. F., Wijesinghe, R. S., Tucker, D. M., Silberstein, R. B., and Cadusch, P. J. (1997). Eeg coherency: I: statistics, reference electrode, volume conduction, laplacians, cortical imaging, and interpretation at multiple scales. *Electroencephalography and Clinical Neurophysiology*, 103(5):499–515.

Palatucci, M., Pomerleau, D., Hinton, G. E., and Mitchell, T. M. (2009). Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 3, pages 5–2.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Panicker, R. C., Puthusserypady, S., and Ying, S. (2010). Adaptation in P300 brain-computer interfaces: A two-classifier cotraining approach. *Biomedical Engineering, IEEE Transactions on*, 57(12):2927 –2935.

Pasley, B. N., David, S. V., Mesgarani, N., Flinker, A., Shamma, S. A., Crone, N. E., Knight, R. T., and Chang, E. F. (2012). Reconstructing speech from human auditory cortex. *PLoS biology*, 10(1):e1001251.

Pfurtscheller, G., Muller-Putz, G. R., Scherer, R., and Neuper, C. (2008). Rehabilitation with brain-computer interface systems. *Computer*, 41(10):58–65.

Pfurtscheller, G., Neuper, C., Flotzinger, D., and Pregenzer, M. (1997). Eeg-based discrimination between imagination of right and left hand movement. *Electroencephalography and Clinical Neurophysiology*, 103(6):642–651.

Porbadnigk, A. K., Treder, M. S., Blankertz, B., Antons, J.-N., Schleicher, R., Möller, S., Curio, G., and Müller, K.-R. (2013). Single-trial analysis of the neural correlates of speech quality perception. *Journal of Neural Engineering*, 10(5):056003.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). *Dataset Shift in Machine Learning*. MIT Press.

Rakotomamonjy, A. and Guigue, V. (2008). BCI competition III: Dataset II- ensemble of SVMs for BCI P300 speller. *Biomedical Engineering, IEEE Transactions on*, 55(3):1147 –1154.

Ranzato, M., Huang, F. J., Boureau, Y.-L., and Lecun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Riccio, A., Mattia, D., Simione, L., Olivetti, M., and Cincotti, F. (2012). Eye-gaze independent eeg-based brain–computer interfaces for communication. *Journal of Neural Engineering*, 9(4):045001.

Rivet, B., Souloumiac, A., Attina, V., and Gibert, G. (2009). xdawn algorithm to enhance evoked potentials: application to brain–computer interface. *Biomedical Engineering, IEEE Transactions on*, 56(8):2035–2043.

Ryan, D. B., Frye, G. E., Townsend, G., Berry, D. R., Mesa, S., Gates, N., and Sellers, E. W. (2010). Predictive spelling with a P300-based brain–computer interface: Increasing the rate of communication. *Intl. Journal of Human–Computer Interaction*, 27(1):69–84.

Salakhutdinov, R., Tenenbaum, J. B., and Torralba, A. (2011). Learning to learn with compound hd models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2061–2069.

Samek, W., Kawanabe, M., and Müller, K.-R. (2014). Divergence-based framework for common spatial patterns algorithms. *Biomedical Engineering, IEEE Reviews in*, 7:50–72.

Samek, W., Meinecke, F. C., and Müller, K.-R. (2013). Transferring subspaces between subjects in brain-computer interfacing. *Biomedical Engineering, IEEE Transactions on*, 60(8):2289–2298.

Samek, W., Vidaurre, C., Müller, K.-R., and Kawanabe, M. (2012). Stationary common spatial patterns for brain–computer interfacing. *Journal of Neural Engineering*, 9(2):026013.

Santhanam, G., Ryu, S. I., Byron, M. Y., Afshar, A., and Shenoy, K. V. (2006). A high-performance brain–computer interface. *nature*, 442(7099):195–198.

Schalk, G., Mcfarland, D. J., Hinterberger, T., Birbaumer, N., and Wolpaw, J. R. (2004). BCI2000: A general-purpose brain-computer interface (BCI) system. *Biomedical Engineering, IEEE Transactions on*, 51:2004.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

Schreuder, M., Blankertz, B., and Tangermann, M. (2010). A new auditory multi-class brain-computer interface paradigm: Spatial hearing as an informative cue. *PLoS ONE*, 5(4):e9813.

Schreuder, M., Höhne, J., Blankertz, B., Haufe, S., Dickhaus, T., and Tangermann, M. (2013). Optimizing event-related potential based brain-computer interfaces: a systematic evaluation of dynamic stopping methods. *Journal of Neural Engineering*, 10(3):36025.

Schreuder, M., Höhne, J., Matthias, T., Blankertz, B., and Tangermann, M. (2011a). Performance optimization of ERP-based BCIs using dynamic stopping. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 4580–4583. IEEE.

Schreuder, M., Rost, T., and Tangermann, M. (2011b). Listen, you are writing! speeding up online spelling with a dynamic auditory BCI. *Frontiers in Neuroscience*, 5.

Shenoy, P., Krauledat, M., Blankertz, B., Rao, R. P., and Müller, K.-R. (2006). Towards adaptive classification for bci. *Journal of Neural Engineering*, 3(1):R13.

Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems (NIPS)*, pages 935–943.

Speier, W., Arnold, C., Lu, J., Taira, R. K., and Pouratian, N. (2012). Natural language processing with dynamic classification improves P300 speller accuracy and bit rate. *Journal of Neural Engineering*, 9(1):016004.

Squires, K. C., Donchin, E., Herning, R. I., and McCarthy, G. (1977). On the influence of task relevance and stimulus probability on event-related-potential components. *Electroencephalography and Clinical Neurophysiology*, 42(1):1–14.

Sugiyama, M. and Kawanabe, M. (2012). *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. The MIT Press.

Sugiyama, M., Krauledat, M., and Müller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005.

Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. In *International Conference on Machine Learning (ICML)*.

Tangermann, M. and Höhne, J. (2011). User-centered brain-computer interface design by optimizing auditory and visual stimuli. *Frontiers in Human Neuroscience*, (224).

Tangermann, M., Höhne, J., Stecher, H., and Schreuder, M. (2012). No surprise — fixed sequence event-related potentials for brain-computer interfaces. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 2501–2504. IEEE.

Tangermann, M., Krauledat, M., Grzeska, K., and Sagebaum, M. (2008). Playing pinball with non-invasive bci.

Tangermann, M., Schreuder, M., Dähne, S., Höhne, J., Regler, S., Ramsay, A., Quek, M., Williamson, J., and Murray-Smith, R. (2011). Optimized stimulation events for a visual ERP BCI. *International Journal of Bioelectromagnetism*, 13(3):119–120.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Kluwer Academic Publishers.

Thurlings, M. E., Brouwer, A. M., Van Erp, J. B., Blankertz, B., and Werkhoven, P. J. (2012). Does bimodal stimulus presentation increase ERP components usable in BCIs? *Journal of Neural Engineering*, 9(4):045005.

Toh, K.-A. (2008). Deterministic neural classification. *Neural Computation*, 20(6):1565 – 1595.

Tomioka, R. and Müller, K. R. (2010). A regularized discriminative framework for EEG analysis with application to brain-computer interface. *Neuroimage*, 49:415–432.

Townsend, G., LaPallo, B. K., Boulay, C. B., Krusienski, D. J., Frye, G. E., Hauser, C. K., Schwartz, N. E., Vaughan, T. M., Wolpaw, J. R., and Sellers, E. W. (2010). A novel P300-based brain-computer interface stimulus presentation paradigm: moving beyond rows and columns. *Clinical Neurophysiology*, 121(7):1109.

Treder, M. S. and Blankertz, B. (2010). (C)overt attention and visual speller design in an ERP-based brain-computer interface. *Behavioral and Brain Functions*, 6:28.

Treder, M. S., Schmidt, N. M., and Blankertz, B. (2011). Gaze-independent brain–computer interfaces based on covert attention and feature attention. *Journal of Neural Engineering*, 8(6):066003.

van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2643–2651.

Vaughan, T., McFarland, D., Schalk, G., Sarnacki, W., Krusienski, D., Sellers, E., and Wolpaw, J. (2006). The Wadsworth BCI research and development program: at home with BCI. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 14(2):229 –233.

Verhoeven, T. (2013). Brain-computer interfaces with machine learning: an improved paradigm for the p300 speller. Master's thesis.

Verschore, H., Kindermans, P.-J., Verstraeten, D., and Schrauwen, B. (2012). Dynamic stopping improves the speed and accuracy of a P300 speller. In *Artificial Neural Networks and Machine Learning–ICANN 2012*, pages 661–668. Springer Berlin/Heidelberg.

Vidal, J. J. (1973). Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2(1):157–180.

Vidaurre, C., Kawanabe, M., von Bünau, P., Blankertz, B., and Müller, K.-R. (2011a). Toward unsupervised adaptation of lda for brain-computer interfaces. *Biomedical Engineering, IEEE Transactions on*, 58(3):587 –597.

Vidaurre, C., Sannelli, C., Müller, K.-R., and Blankertz, B. (2011b). Co-adaptive calibration to improve BCI efficiency. *Journal of Neural Engineering*, 8(2):025009.

von Bünau, P., Meinecke, F. C., Király, F. J., and Müller, K.-R. (2009). Finding stationary subspaces in multivariate time series. *Physical review letters*, 103(21):214101.

Wang, Z., Schalk, G., and Ji, Q. (2011). Anatomically constrained decoding of finger flexion from electrocorticographic signals. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 2070–2078.

Wolpaw, J., Birbaumer, N., Heetderks, W., McFarland, D., Peckham, P., Schalk, G., Donchin, E., Quatrano, L., Robinson, C., and Vaughan, T. (2000). Brain-computer interface technology: a review of the first international meeting. *Rehabilitation Engineering, IEEE Transactions on*, 8(2):164 –173.

Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., and Vaughan, T. M. (2002). Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767 – 791.

Wolpaw, J. R., McFarland, D. J., Neat, G. W., and Forneris, C. A. (1991). An eeg-based brain-computer interface for cursor control. *Electroencephalography and Clinical Neurophysiology*, 78(3):252–259.

Xu, P., Yang, P., Lei, X., and Yao, D. (2011). An enhanced probabilistic LDA for multi-class brain computer interface. *PLoS ONE*, 6(1):e14634.

Yanez-Suarez, O., Bougrain, L., Saavedra, C., Bojorges-Valdez, E., and Gentiletti, G. G. (2012). P300-speller public-domain database.

Yuan, P., Gao, X., Allison, B., Wang, Y., Bin, G., and Gao, S. (2013). A study of the existing problems of estimating the information transfer rate in online brain–computer interfaces. *Journal of Neural Engineering*, 10(2):026014.