UNIVERSITY OF AGDER

Stochastic Learning-Based Estimation Methods for Pattern Recognition and Its Application to Topic Detection and Tracking

by
Aleksander Mølsæther Stensby

Master Thesis in
Information and Communication Technology

University of Agder

Grimstad, May 26, 2008

Abstract

Every Pattern Recognition (PR) problem involves a training and a testing phase. In the training phase, the system is presented with samples, using which the distribution (also called the class-conditional distribution), of the features, is estimated. Traditional PR systems assume that the class-conditional distributions are stationary, and thus that they do not change with time. Recently Oommen and his co-authors have presented a strategy by which the parameters of a binomial/-multinomial distribution can be estimated when the distribution is non-stationary.

In this thesis, we propose a selection of performance indexes that take into account crucial characteristics of non-stationary environments. Furthermore, we use the proposed indexes to perform a more extensive empirical evaluation of the presented strategy, and compare it with traditional estimation algorithms operating in non-stationary environments. The purpose is to bring forward the unique strengths/weaknesses of the competing approaches.

This thesis will consider the design and implementation of PR-systems dealing with such non-stationary environments. In particular, we shall concentrate on the application domain that deals with language classification in multilingual Word of Mouth discussions. Unlike traditional PR systems, one novel feature of our method is that the training is achieved by learning the N-gram characteristics of every language. The testing, however, invokes the SLWE because the sample documents being classified contain parts written in different languages, interspersed with each other, without the user knowing when one language stops, and the second language starts. Our empirical testing demonstrates that our proposed method is capable of classifying multilingual documents with high overall accuracy. We show that our method scales well with regard to the dimensionality of the feature space, and that it is resistant to textual errors in the testing data. Finally, and more importantly, the classifier performs extremely well when classifying segments of moderate size (15-20 words), with a reported overall classifier accuracy of 0.989, and adequately for shorter segments (10 words per segment), yielding an accuracy of 0.9596.

Thus, we believe that our results provide additional insight into the performance of the SLWE and the MLE when operating in non-stationary environments. Furthermore, it is our opinion that our proposed technique for language classification will be of benefit in applications dealing with Pattern Recognition in multilingual text documents.

Preface

This master thesis is submittet in partial fulfillment of the requirements for the degree Master of Science in Information and Communication Technology at the University of Agder, Faculty of Engineering and Science. The project is supported by Integrasco A/S, who has provided data material and insight for the various simulations performed in this study. This work was carried out under the supervision of professor John B. Oommen at Carleton University, Canada and co-supervisor associate professor Ole-Christoffer Granmo at the University of Agder, Norway.

First of all, I wish to thank my supervisor and dear friend, John B. Oommen to whom I owe so much. I thank him for his excellent tutoring, follow up and support throughout the project period. Without his help, I would not have finished this thesis. Believing in me and showing great commitment helped me keep the spirit up and finish in time. His expertise in the field of learning automata and pattern recognition has led to countless hours of discussion and I have learned a lot. I also wish to thank professor Ole-Christoffer Granmo for his assistance and support throughout the project period. Giving valuable feedback on my writing and fruitful discussions on the proposed solution has lifted the quality of this thesis. My colleagues at Integrasco A/S have stood by me, and I would like to address a special thank you to my boss Jan Hansen and my co-worker Jaran Nilsen. Last but not least I would like to thank my fellow student and friend Ole-Alexander Moy for countless discussions and late evenings at the University.

Grimstad, May 2008.
Aleksander M. Stensby

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we introduce the problem domain of estimation in non-stationary environments. In particular, we shall concentrate on PR-systems where the class-conditional distributions of the features are non-stationary. In Section 1.1 we present the background and motivation of our thesis and in Section 1.2 we present our thesis definition. Section 1.3 presents our claims in support of the thesis, and in Section 1.4 we discuss the contribution of our thesis. Section 1.5 briefly describes our intended audience before we outline the remainder of this thesis in Section 1.6.

1.1 Background and Motivation

Every Pattern Recognition (PR) problem essentially involves two issues, namely the training and the classification of the patterns. In the training phase, the class-conditional distribution of the features is estimated, based on given training samples. Generally speaking, traditional PR systems assume that the class-conditional distributions are stationary, and thus that they do not change with time.

Statistical problems involve estimation as one of the fundamental issues. Since the problem involves random variables, the training/classification decisions are in some way dependent on the system obtaining reliable estimates on the parameters that characterize the underlying random variable. These estimates are computed from the observations of the random variable itself.

1.1.1 Estimation As a Tool

The theory of estimation has been studied for hundreds of years [3, 5, 26]. It traces its origins back to astronomers and their efforts to understand and predict planet and asteroid motion. Generally, estimation methods fall into various categories, including the Maximum Likelihood Estimates (MLE) and the Bayesian family of estimates [12, Ch.3], [61, pp.28–33]. The principle of Maximum Likelihood Estimation was pioneered by Fisher in the 1920s, and has a wide range of applications. It is an indispensable tool for many statistical modeling techniques, and in particular, in non-linear modeling with non-normal data. Although the MLEs and Bayesian estimates are known for having both good computational and statistical properties, they work with the premise that the parameters being estimated does not change with time, i.e., the distribution is assumed to be *stationary*. Moreover, it is desirable that the estimate converges to the true underlying parameter with probability 1, as the number of samples increases.

Consider, however the scenario when the parameter being estimated changes with time. Thus, for example let us suppose that the parameter of a binomial distribution is "switched" periodically. Such a scenario demonstrates the behavior of what we call a *non-stationary* environment. The goal of an estimator scheme in this case would thus be to estimate the parameter, and to be able to adapt to any changes occurring in the environment, e.g., the algorithm must be able to detect the changes and estimate the new parameter after a "switch" has occurred in the environment. If one use strong estimators (i.e., estimators that converge w.p. 1), it is unlikely that the learned parameter (or phenomenon) will change rapidly from that which it has converged to and thus resulting in poor estimates.

1.1.2 Weak Estimators and Non-Stationary Environments

Recently Oommen and his co-authors have presented a strategy by which the parameters of a binomial/multinomial distribution can be estimated when the distribution is non-stationary [46]. The method is referred to as the *Stochastic Learning Weak Estimator* (SLWE), and is a novel estimation method based on the principles of stochastic learning. The convergence of the estimate is *weak*, i.e., with regard to the first and second moments. Analogous to stochastic learning, the SLWE possess a user-defined *learning coefficient*, λ, and both the estimate variance and the speed decrease with this learning coefficient.

The traditional strategy to deal with non-stationary environments has been one of using a

sliding window [23]. The problem with this is that if the size of the window is too small, the corresponding estimates tend to be poor. In the case when choosing a too large window size, the estimates prior to the change of the parameter have too much influence on the new estimates. Moreover, the observations during the entire window width must be maintained and updated during the process of estimation.

1.1.3 Topic Detection and Tracking and Word of Mouth

There are numerous problems which we have recently encountered, where strong estimators pose a real-life concern. One such scenario occurs in pattern recognition problems related to Topic Detection and Tracking (TDT) in online discussions, where the content of the discussions represent the opinions of users from all over the world. This kind of information has high value for market-oriented or consumer-focused companies.

The ever expanding world wide web has become an important part of our everyday life. An overwhelming amount of information presents new challenges for search technology and the associated organizational tools such as those which involve large database systems. It is fair to say that the Internet has become a new communication channel for people from all over the world. The ability to express individual opinions and take advantage of a person's right to free speech is provided to the users through blogs, discussion boards and other online communities. Social networking is one of these rapidly growing phenomena. In addition to being a meeting place for users where they exchange experiences and opinions, it is also a source of knowledge for every single user. People exchange knowledge, and go online seeking other opinions, rendering the credibility of the individual to be almost higher than those claimed by commercials or newspaper advertisements.

Typically, the users discuss products, exchange experiences, and ask for advice before buying new products. If a consumer encounters a problem with his product, it is possible for him to go online and ask for help without having to go back to the store where the product was purchased. The information that people used to share with their friends on campus or over a cup of coffee is now being broadcasted online, and thousands of people are listening to the personal opinions of others. The phenomenon of consumers providing information to other consumers is often referred to as Word of Mouth (WoM), and according to a marketing forecast, published by PQ Media in 2007, advertisers spend $\$1$ billion on word-of-mouth marketing [50]. Moreover, their forecast predicts that spending will reach $\$3.7$ billion in 2011.

The nature of these discussions, consisting of multiple opinions, different topics and different languages, presents us with a problem of finding training and classification strategies when the class-conditional distribution is non-stationary.

Companies dealing with WoM analysis collect vast amount of data from discussion boards on the Internet. To aid the work of trend analysis, these companies are constantly seeking to find new and intelligent methods for organizing the data, classifying it and detecting important events when they occur. Due to the vast amount of available information, it is of high importance to automate these classification tasks.

1.1.4 Language Classification in Multilingual Documents

Traditional text classification involves tasks such as classifying news articles into categories such as *politics*, *sport* or *business*. This is often referred to as topic classification. Another text classification task is the classification of sentiment or opinions such as classifying movie reviews into *positive* or *negative* reviews. A crucial problem that has received little attention in the literature is that of classifying documents containing several languages, or so-called multilingual documents. The task of language classification has been widely studied, but most of the approaches focus on classifying documents written in a single language, often referred to as monolingual documents. Cavnar & Trenkle approached the task of language classification in monolingual documents in [6], by using N-Gram analysis. The basic idea of their approach is to identify so-called N-grams whose occurrences in a document gives strong evidence for or against identification of a text as belonging to a particular language. Their method proved to be a reliable and simple way to categorize documents for a wide range of classification tasks. Another frequently used approach to language classification is the dictionary approach, where a dictionary is kept for each possible language, and a look up on every word in the sample document is done to see in which dictionary it occurs in. Then, the dictionary which contains the most words from the document indicates which language the document was written in. A simplified version of this approach is to use a lists of words that commonly appear in the languages of interest [21], instead of the full dictionary. Such non-linguistically motivated features generally perform well for documents of moderate length, but their performance is significantly decreased when the length of the sample text gets shorter.

Other approaches to language classification, using linguistic factors that differ among languages are also found in the literature. One such approach is based on the use of *morphological* features presented in [10]. Morphemes are the smallest components of a language that carry se-

mantic value. The problem with this approach is that construction of a morphological lexicon for a given language requires a large amount of work by trained experts. There are ways of generating simplified morpheme "language models", and one such algorithm that has demonstrated high accuracy for various languages, has been developed by Creutz [11]. Creutz uses an Hidden Markov Model to model morpheme sequences, without assuming prior knowledge of the morphemes themselves, nor of their individual functional categories.

Common for all of these approaches to language classification is that they deal with monolingual documents. In this thesis we are interested in classification tasks that involve non-stationary data such as multilingual documents, which contain multiple languages within each document. Moreover, we do not know the boundaries of the different language segments in the document. Ozbek et. al. presented an approach in [47], where they make use of Creutz algorithm. Their approach demonstrated good results for the Turkish language, but for the English language, the results were discouraging, with worst performance as low as 40% accuracy. Ludovik & Zacharski propose an algorithm for classifying multilingual documents that is based on mixed-order n-grams, Markov chains, maximum likelihood and dynamic programming in [34]. To deal with the multilingual nature of the documents, Ludovik *et. al.* use a dynamic programming algorithm based on a Markov model of multilingual documents. This approach is used to segment the documents into monolingual parts, and each segment is classified individually. Language classification in multilingual documents using a word-window approach has been studied in [35] by Mandl *et. al.*

In contrast to previous solutions, we will in this thesis consider the design and implementation of a PR-system dealing with language classification in multilingual documents where the classification is done without any prior segmentation of the sample document, using the SLWE proposed by Oommen *et. al.* In particular, our goal is to design a solution that handles short segments that are subject to noise, such as individual postings found in online WoM discussions.

1.2 Thesis Definition

We formulate the thesis definition in the following fashion:

> *We will investigate the new family of weak estimators, proposed by Oommen and his co-authors in [46] and formally prove the convergence properties of their approach, operating in non-stationary environments. Moreover, we will empirically*

evaluate the performance and scalability of their method in comparison to the traditional sliding window approach of the maximum likelihood estimates, through simulations on synthetic data.

We will also consider the design and implementation of PR-systems where the class-conditional distribution of the features are non-stationary. In particular we will concentrate on the application domain that deals with language classification in multilingual Word of Mouth discussions. We will explore the use of the proposed weak estimators in order to solve this problem. Furthermore, we will evaluate the performance of the resulting solution on real-life multilingual documents.

1.3 Claims

In this thesis we demonstrate that it is sub-optimal to work with strong estimators when the data is non-stationary. Moreover, we will empirically demonstrate the superiority of the new family of weak estimators over the traditional estimation methods operating in such application domains, through simulations on synthetic data.

We will also investigate the SLWE and demonstrate its power when applied to PR-tasks where the class-conditional distribution of the features exhibits a non-stationary behavior. In particular, we will propose a novel solution to language classification in multilingual documents.

We claim that the SLWE demonstrates its power over other approaches found in the literature, because of its ability to maintain and update a running estimate at each time instant, based on the value of the current sample, i.e., without maintaining a sliding window or performing any prior segmentation of the sample documents being classified.

1.4 Contribution

In this thesis we evaluate the SLWE in comparison to the traditional sliding-window variant of the MLE through extensive laboratory experiments on synthetic data to demonstrate the power of the SLWE operating in non-stationary environments. We develop a test bench for simulations on both binomial and multinomial distributions to assess the efficiency and robustness of the estimator algorithms with regard to their ability to adapt to changes in the environment and with regard to their scalability when the feature space grows large. We believe that our results provide addi-

tional insight into the performance of the SLWE and the MLE when operating in non-stationary environments.

Traditionally, TDT related tasks focus on news articles or documents of a certain length. The main difference between classification of news articles or journal papers and WoM discussions, is that discussions generally contain the opinions of several different authors. Considering a discussion where several authors write parts of it means that we have a document with continuous content changes. Not only may the subject or topic of each segment in the discussion change, but the discussion can have multiple sentiments and it can contain multiple languages as well.

Another important aspect of text classification of such WoM discussions is that the postings often are composed on the fly by the different users, without any form of spell checking. Thus, when performing text classification on such data, one must tolerate the presence of different kinds of textual errors, such as spelling and grammatical errors. Abbreviations and Internet "slang" may also be present.

This thesis considers these challenges and focus on the design of a PR-system that is able to deal with this in a low computational manner, without prior knowledge about the boundaries of the different segments in the document. The state-of-the-art on classification of multilingual documents are either computationally expensive with regard to morphological features or prior segmentation of the documents. In addition, most of these approaches are dealing with documents containing a substantial number of words. Our approach differs from these approaches in the way that the classification is done without prior segmentation, and by using low-computational, non-linguistic, discriminatory features. The proposed method is able to deal with short segments and are resistant to the presence of grammatical or spelling errors.

While the paper by Oommen *et. al.* is theoretical and only solves the general problem of SLWE for non-stationary environments, this thesis provides a new potential application of the scheme. We introduce a novel approach to language classification in multilingual documents that utilizes the principles of the SLWE to update the probabilities of the input samples, combined with mixed order N-grams as the discriminatory features.

Furthermore, it is our opinion that our proposed technique for language classification will be of benefit in applications dealing with Pattern Recognition in multilingual text documents.

Thus, our contribution can be summed up in the following sentences: We evaluate the new family of weak estimators operating in non-stationary environments and compare this approach to traditional estimation methods through extensive laboratory experiments. We present a language

classifier in which the SLWE estimates can be used in conjunction with non-linguistic features to classify documents, where the data is characterized by a non-stationary distribution, and no prior segmentation of the document is done. We demonstrate this fact by utilizing it in the context of multilingual WoM discussions that are subject to noise and where the individual segments may consist of a limited amount of words.

Furthermore, we also believe that our approach can be adapted to similar classification tasks, such as topic classification and sentiment classification in documents containing more than one topic or opinions of several authors.

1.5 Target Audience

The target audience of this thesis is anyone interested in PR-systems, estimation techniques, language classification or in general applications of PR-systems to Word of Mouth analysis. The thesis is written in a language that requires some insight in the field of estimation and pattern recognition, but most of these technical details are explained in a way that should make it understandable for anyone with sufficient background in computer programming. Also, since estimation theory is closely related to probability theory, the reader should have fundamental knowledge about probability theory, since this is not explained in this thesis.

1.6 Report Outline

The rest of this thesis is organized as follows: Chapter 2 gives a brief overview of the field of Topic Detection and Tracking (TDT), which is closely related to Pattern Recognition (PR) and Information Retrieval (IR) systems. The chapter gives an introduction to some popular techniques related to TDT, and in addition it also details the problem domain associated with online discussion boards and so-called Word of Mouth (WoM) analysis. Chapter 3 gives a detailed description of estimation theory and some of the traditional estimation techniques. Chapter 4 formally presents the Stochastic Learning Based Weak Estimator (SLWE). Here we analyze and prove theorems formerly presented by Oommen *et.al.* In Chapter 5 we present simulation results from simulations on synthetic data, using both the SLWE and the sliding window variant of the MLE (MLEW), to assess the efficiency and robustness of the SLWE in comparison to the traditional MLEW. In Chapter 6 we present a novel approach to language classification in multilingual documents, using

the SLWE and mixed order N-gram language models. A prototype implementation is presented and results from extensive experiments are presented to demonstrate that the SLWE is capable of classifying documents of different length with high overall accuracy. Chapter 7 concludes this thesis. It also outlines some problems that arise and may be subject to further research.

Chapter 2

Topic Detection and Tracking

This chapter gives a brief overview of the field of Topic Detection and Tracking (TDT), and various techniques related to it. The chapter also details the problem domain associated with online discussion boards and so-called Word of Mouth (WoM) analysis.

2.1 Word of Mouth and Online Discussion Boards

The ever expanding world wide web has become an important part of our everyday life. An overwhelming amount of information presents new challenges for search technology and the associated organizational tools such as those which involve large database systems. It is fair to say that the Internet has become a new communication channel for people from all over the world. The ability to express individual opinions and take advantage of a person's right to free speech is provided to the users through blogs, discussion boards and other online communities. User-generated content such as online videos, podcasts and blogposts are more popular than ever.

Social networking is one of these rapidly growing phenomena. Online social networks like *MySpace*, *Facebook* and *Orkut* are prime examples of this. In September, 2007, *MySpace* could celebrate having over 200 million user accounts on their site. In the same vein, *Facebook* has over 34 million active members worldwide, and *Orkut* had over 67 million members as of August 2007. In addition to being a meeting place for users where they exchange experiences and opinions, it is also a source of knowledge for every single user. People exchange knowledge, and go online seeking other opinions, rendering the credibility of the individual to be almost higher than those claimed by commercials or newspaper advertisements.

Besides the large social network sites, there are also many smaller ones. Arguably, they are almost as significant, since they involve a large number of discussion boards, newsgroups and blogs. Online discussion boards tend to focus on specific topics or technologies. One example of this could be a discussion board which focuses on cars, while a different board may be focusing on cell phones or computers. Discussion boards that have regular users often develop into a small social community. Figure 2.1 shows a screenshot from a forum called *boards.ie*. In this example, the users are discussing a problem with a hard drive. Typically, the users discuss products, exchange experiences, and ask for advice before buying new products. If a consumer encounters a problem with his product, it is possible for him to go online and ask for help without having to go back to the store where the product was purchased. These discussion boards, where people from all over the world are discussing *en masse*, have become a new marketing channel over the last couple of years. The information that people used to share with their friends on campus or over a cup of coffee is now being broadcasted online, and thousands of people are listening to the personal opinions of others.

Figure 2.1: Users discussing on the forum *boards.ie*

2.1.1 Word of Mouth

The phenomenon of consumers providing information to other consumers is often referred to as Word of Mouth (WoM). According to the WoM Marketing Association, [64] the latter is

> *"the art and science of building active, mutually beneficial consumer-to-consumer and consumer-to-marketer communications."*

Consumer generated product reviews and opinions is an emerging marketing phenomenon. Product reviews, consumer opinions and end-user experiences play an increasingly important role in the decisions made by consumers before a purchase takes place. According to a study made by *eMarketer* in June 2007 [15],

> *"More and more, consumers are relying on advice from friends, family and even strangers to make purchase decisions, select physicians, choose travel destinations and pick politicians to vote for."*

The study states that 64 million US adults regularly share advice on products or services, and over 25 million of them wield their influence online. Statistics provided by the *Pew Internet American Life Project* in 2001 show that 84% of Internet users (which is equivelent to about 90 million Americans) have participated in online discussion groups [20].

2.1.2 Impact and Value

According to *Integrasco A/S*, a trend analysis company based in Norway, the truly market-oriented or customer-focused companies learn from their customers by listening to what they say when it counts the most — i.e. when they share their experience and knowledge with other customers.

Jan Hansen, CEO of *Integrasco*, asserts:

> *"Word of Mouth on the net offers a unique opportunity for companies that seeks a substantial ROI from their efforts toward Customer Understanding"*,

By measuring the buzz or volume of discussion about certain topics, and observing how marketing strategies affect this buzz, companies can learn from their own mistakes, and improve their products in the future. The monitoring and analysis of this buzz is often called trend analysis or buzz analysis.

2.1.3 Integrasco A/S and Trend Analysis

As companies are starting to acknowledge the value and usefulness of WoM as an instrument for both consumers and marketers, several tools and services known as "Buzz Monitoring" have risen in Public Relations societies over the last couple of years. *Integrasco A/S* is one of the

companies offering services for this type for tracking and analyzing online WoM. As part of their trend analysis, they collect vast amounts of data from discussion boards on the Internet. The data is stored in their database system and processed in relation to Buzz-measurement, early detection, product comparisons, and from these, written reports on the current market situation is submitted to their costumers. To aid the work of trend analysis, *Integrasco* is constantly seeking to find new and intelligent methods for organizing data, classifying it and detecting important events when they occur.

Methods useful to such trend analysis may, for instance, be capable of tracking change in positive/negative opinions, also known as sentiment analysis. Also, the question of observing how users change topics as the discussion evolves is also fascinating, because it indicates what is generating buzz at the present moment, and to what extent this effects the online communities.

2.2 Topic Detection and Tracking

Since the large amount of electronically available information on the Internet is ever increasing, search engines, newsreaders and corporate indexing systems are being developed to cope with this. Structuring and clustering data and presenting it to the user in a compact and comprehensible fashion is desirable, but not always feasible. This is the main objective of Information Retrieval (IR) systems. IR is the task of finding documents that are relevant to a user-specified query, or to the user's need for information. Well known examples of IR systems are the available search engines on the Internet, such as *Google* or *Yahoo*.

A busy lifestyle characterizes the average "Joe" of today's society, thus rendering fast information access mandatory. People want concise and exact information "in the twinkling of an eye". According to *Internet World Stats* [22], in June 2007, 17.8 % of the world's population accessed the Internet on a regular basis, and according to their statistics, there has been a growth in usage by an astonishing rate of 225.0% from 2000 to 2007.

This enormous amount of information, and the fact that it is not humanly feasible to navigate the Internet without the appropriate tools, has led to a new line of research called Topic Detection and Tracking (TDT) [44]. The TDT research was initiated and supported by the U.S. Government in 1996. There have also been several research projects related to TDT. The *Pilot Research Project* described that it would be desirable for an intelligent system to automatically detect significant events from a large corpus of news articles, alert the onset of novel events as they happen, and

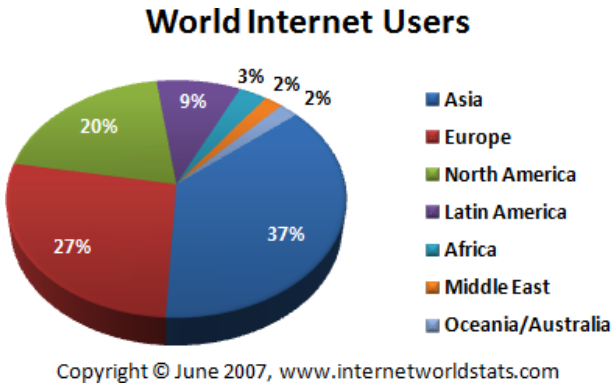

Figure 2.2: World Internet Users, June 2007

track events based on user-given sample stories. In the context of the TDT project, "an event identifies something happening in a certain place at a certain time". Topics, on the other hand, covers several similar events. Thus the item *"Cuban soldiers hijacks airplane"* would refer to a specific event in the topic titled *"airplane hijacking"*.

Although the TDT project, as it was initiated by the U.S. Government, has been terminated, the concepts are still being researched, and are still highly relevant. Despite the fact that the initial TDT project was intended to be used against news articles, the same theory and problems apply to other collections of articles, such as blogposts or web discussions. Finding similar discussions on the same topic (or which concern the same event) quickly, effectively and without having to manually search through extensive discussions, would be one such task.

From a classification perspective, event detection can be seen to be an unsupervised learning task. TDT distinguishes between two ways of doing this, namely through *retrospective detection* or *on-line detection*. Retrospective detection is based on a preset of chronologically-ordered documents or stories, while on-line detection considers the onset of new events from data obtained in a real-time manner.

Event tracking on the other hand is a supervised learning task, aiming to automatically label the incoming documents or stories based only on a small number of previously identified past stories that define each event.

Traditional TDT research focus on the classification of news articles or similar documents containing a significant amount of word. The documents being classified are also assumed to contain a single topic or subject. The application of TDT to WoM present us with a new area of research since these documents consist of shorter segments, and may indeed consist of multiple

subjects and sentiments. Moreover, they may consist of segments that are written in different languages. We will investigate this problem domain further in Ch. 6.

2.3 Relevancy, Ranking and Similarity Measurement

Detecting textual similarities is an important building block in document collection structuring (e.g. clustering), or in IR. The art relies on the computation of indices quantifying textual similarities, and on measuring the distance between a given query and documents, or the similarity between multiple documents.

To detect the relevance of a document to a specific user's query or quest for information, is a highly pertinent problem. Ranking documents is also a task that can be done to prune a large collection of documents before presenting them to the user. To perform such actions, the system needs a metric of measurement that can be used to calculate the similarity or the difference between documents. Furthermore, to be able to apply good measures, the documents must also be represented in a suitable model or structure. One of the most commonly used models in IR systems is the Vector Space Model explained presently.

2.3.1 The Vector Space Model

The Vector Space Model (VSM, also called the vector model) was first presented by Salton *et. al.* [54] in 1975, and used as part of the SMART[1] Information Retrieval System developed at Cornell University. The model involves an algebraic system for document representation, where, in the processing of the text, the model uses *vectors* of *identifiers*, where each identifier is normally a *term* or a token. For the purpose of the representation of documents, the VSM would be a list of vectors for all the terms that occur in the document. Since a document can be viewed as a long string, each term in the string is given a correlating value, called a *weight*. Each vector consists of the identifier and its weight. If a certain term exists in the document, the weight associated with the term is a non-zero value, commonly a real number in the interval $[0, 1]$. The number of terms represented in the VSM is determined by the vocabulary of the entire document collection or corpus.

Although the VSM is a powerful tool in document representation, it has certain limitations.

[1] SMART is an abbreviation for Saltons Magic Automatic Retriever of Text.

The obvious weakness is that it requires vast computational resources. Also, when adding new terms to the term space, each vector has to be recalculated. Another limitation is that "long" documents are not represented optimally with regard to their similarity values as they lead to problems related to small scalar products, and large dimensionalities. Furthermore, the model is sensitive to semantic content, e.g. documents with similar context but different term vocabularies will not be associated. This is often referred to as a *false negative match*. Another important limitation that is worth mentioning is that search terms must match the terms found in the document precisely, because substrings might result in what is called a *false positive match*. Last, but not least, this model does not preserve the order in which the terms occur in the document. Despite these limitations, the model is useful, and can be improved in several ways as explained in Section 2.4.

Salton also presented a theory of "term importance" in automatic text analysis in [55]. There, he stated that the terms which have value to a document are those that highlight differences or contrasts among the documents in the corpus. He noted that

> *"A single term can decrease the document similarity among document pairs if its frequency in a large fraction of the corpus is highly variable or uneven."*

A very simple term weighting scheme is called the Term Count Model. Here, the weight of each term is simply given by counting the number of occurrences of the term. This is also called the term frequency and has the form:

$$w_i = tf_i, \tag{2.1}$$

where tf_i is the frequency of the term i.

The problem with this scheme is that it is inadequate when it concerns the repetition of terms, and that it actually favors large documents over shorter documents. Large documents obtain a higher score merely because they are longer, and not because they are more relevant.

The Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme achieves what Salton described in his term importance theory by associating a weight with every token in the document based on both local information from individual documents and global information from the entire corpus of documents. The scheme assumes that the importance of a term is proportional to the number of document that the term appears in. The TF-IDF scheme models both the importance of the term with respect to the document, and with respect to the corpus as a whole [53][55], and is explained in more detail in the next section.

2.3.2 The TF-IDF Scheme

The Term Frequency-Inverse Document Frequency (TF-IDF) scheme weights a term based on how many times it is represented in a document, and this weight is simultaneously negatively biased based on the number of documents it is found in [56]. Such a weighting philosophy can be seen to have the effect that it correctly predicts that very common terms, occurring in a large number of documents in the corpus, are not good discriminators of relevancy, which is what Salton required in his theory of term importance.

The TF-IDF scheme is described in the following equation[2]:

$$w(t,d) = \frac{tf(t,d) \times \log_2(N/n_t)}{\left\|\vec{d}\right\|}, \text{ where,} \tag{2.2}$$

$$\left\|\vec{d}\right\| = \sqrt{\sum_t (tf(t,d) \times \log_2(N/n_t))^2}. \tag{2.3}$$

In the above, $w(t,d)$ is the weight given to term t in document d, $tf(t,d)$ is the term frequency of term t in document d, and $\log_2(N/n_t)$ is the inverse document frequency (IDF). Also, N represents the number of documents in the corpus, while n_t represents the number of documents containing the term t[3], and d is the length of the entire term vector. Observe that the denominator is used to normalize the document weight vector.

Adaptive TF-IDF

The static TF-IDF weighting scheme presented above becomes inefficient when the system has documents that are continously arriving, e.g. systems used for *online* detection. Incremental or adaptive IDF can be efficiently used for document retrieval after a sufficient number of "past" documents have been processed [65]. The initial IDF values are calculated using a retrospective corpus of documents, and these IDF values are then updated incrementally using the following equation:

$$IDF(t,p) = \log_2(N(p)/n_t(p)), \tag{2.4}$$

[2]Note that in [65], a slightly different variant of the TF-IDF scheme is used, namely, the standard "ltc" version.
[3]This is also called the document frequency or DT.

where p is the current point in time, $N(p)$ is the number of accumulated documents up to that point, and $n_t(p)$ is the document frequency for a term t at time p.

The adaptive variant of the TF-IDF scheme is both pertinent and highly suitable for systems which involve the processing of online data (e.g., a stream of incoming documents).

2.3.3 Similarity Functions

The Euclidean and Mahalanobis distances are commonly used as dissimilarity functions for numeric data. Analogously, the Normalized dot product, or Cosine similarity is also used as a similarity function for high-dimensional data [12]. In IR and TDT, the user wants to measure the similarity between *documents*, and thus we seek for methods that are capable of leading to such measures. When it comes to strings and string comparisons, metrics such as the Hamming distance and the Levenshtein distance are commonly used. These metrics measure the dissimilarity (or difference) between two strings with respect to their individual characters. The Levenshtein distance measures the edit distance between the strings, as described in Section 2.4.4.

Techniques used for calculating string similarity can be classified into two groups: Sequence-based functions and vector-space-based functions. The Hamming and Levenshtein distances fall into the first category of functions. As opposed to these, Vector-space-based functions are different because they view strings as collections of elements. An important motivation for using vector-space-based functions is that the corresponding sequence-based functions become computationally expensive for longer strings. Also, they do not perform well when it comes to accuracy. In such cases, the computational complexity on large strings such at news articles, discussion threads or blog articles is not feasible, since the complexity is quadratic in the size of the string — for example, if one is to use an edit distance as a metric of measurement.

The above-mentioned problems are avoided through the use of the vector-space-based functions, in which the collection of tokens are treated without giving consideration to the order in which they occur in the document [53].

Jaccard Similarity

A vector-space-based function which attempts to resolve the above issues is the Jaccard similarity. If strings i and j are represented by sets of tokens I and J respectively, the Jaccard similarity is

described by the following equation:

$$Jaq\_Sim(i,j) = \frac{|I \cap J|}{|I \cup J|}. \tag{2.5}$$

The problem with this measure is that it does not make use of the term weight scheme presented in Section 2.3.2, because, as argued earlier, it is desirable to take these TF-IDF values into account when measuring the similarity between two documents. As mentioned, the tokens that occur frequently in the document should have a higher contribution to the similarity than those that occur infrequently. Moreover, tokens that occur few times throughout the whole corpus should also be given more value in the similarity measure.

Cosine Similarity

An alternative measure, and by far the most commonly used similarity function for this purpose, is the cosine similarity function or the normalized dot product measure. This function compares the deviation of angles between each document, or rather the vector representing the documents. For practical reasons, one often calculates the cosine of the angle instead of the angle itself. If the cosine value is zero, it means that the two documents in question are orthogonal to each other, or, in other words, that they do not match at all. Similarity functions can also be applied when comparing a document to a user query, where the query is presented in the same vector form as the document itself.

The TF-IDF cosine similarity between two strings is defined as the cosine of the angle between their vector representations, and has the form:

$$TF - IDF\_Sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\left\|\vec{i}\right\| \times \left\|\vec{j}\right\|}. \tag{2.6}$$

The TF-IDF cosine similarity function is computationally efficient due to the high sparsity of most of the vectors involved, and by using an appropriate inverted data structure for an efficient representation mechanism. Indeed, the TF-IDF cosine similarity is considered to be a reasonable off-the-shelf metric for long strings and text documents.

2.4 The Term Normalization Process

A process that is usually accomplished when setting up IR systems is the so-called "Term Normalization" process. In order to improve the term-vector model, researchers have attempted various steps. Among the most used steps are those which (a) Eliminate Stop Words and very common terms, (b) Stem terms to their root form, (c) Compute sub-vectors in long documents, and (d) Avoid retrieving documents which fall below a predefined similarity threshold. This section describes the tokenization, Stop Word removal and stemming processes. We also dedicate a paragraph to spell checking as a tool for improving term weights.

2.4.1 Tokenization

In [2], tokenization is described in the following fashion:

> *"Tokenization is typically performed by treating each individual word of certain minimum length as a separate token usually excluding a fixed set of functional Stop Words and optionally stemming tokens to their roots."*

A tokenizer is a method that separates tokens according to a predefined pattern. The simplest form of a tokenizer separates the terms where there are occurrences of white space. More advanced tokenizers make use of filters, and can, for instance, remove special characters, non-alpha-numerical characters, url-patterns and other predefined patterns. It is also common to transform all characters to be in lowercase before processing the input through the tokenizer to prevent differentiation between upper and lower-cased terms.

After the string has been separated into individual tokens, the next step of the normalization process is often to *stem* the word, as explained below.

2.4.2 Stemming

Stemming algorithms attempt to reduce a word to a common root form. The process of stemming mainly involves removing the more common morphological and inflexional endings from words. This is seen for example, with "lazy" being reduced to "lazi", in which case the word "laziness" would also be reduced to "lazi", thus permitting searches for either one of the words to locate

documents containing the other. Other examples are the plural form of all words. Observe that if the process of stemming is not invoked, the words “car” and “cars” would be interpreted as two completely different terms. Algorithmic stemmers continue to have great utility in IR, and are commonly used in search technology to improve the search result.

2.4.3 Stop Words

Another action that can be done to improve the quality of measures is to remove common words with little information content from the corpus. These common words are often called “Stop Words”, and include pronouns and articles including “the”, “to”, “be”, “a”, etc. While Stop Words take up space in the index, they do not improve the scoring of results in IR systems.

After removing the Stop Words, the document will only be represented by the so-called “content bearing” words. Salton estimates that 40-50% of the total number of words in a document can be removed with the help of a stop list [53].

The downside of removing high frequency words or Stop Words is that the indexing method becomes language dependent.

2.4.4 Spell Checking

Although, the removal of stop words is a fairly trivial procedure in the term normalization process, it is very powerful. A more advanced step to further improve the corpus is to run a proximity searching algorithm to remove any misspelled words. This can sometimes be a complex task, and can also impose unintended errors on the corpus if the algorithms make false assumptions.

A well known scheme for measuring the differences between strings is the Levenshtein distance. Classical (Levenshtein) edit distance between two strings is defined as the minimum cost of the edit operations (deletions, insertions, and substitutions of elements) required to transform one string into another [32]. The Levenshtein distance can be considered to be a generalization of the Hamming distance, which is used for strings of the same length, and, which only considers substitution edits.

Chapter 3

Traditional Estimation

The theory of estimation has been studied for hundreds of years [3, 5, 26]. It traces its origins back to astronomers and their efforts to understand and predict planet and asteroid motion. Statistical problems involve estimation as one of the fundamental issues. By observing random variables, reliable estimates of the parameters that characterize these variables can be computed.

In this chapter we give an introduction to estimation theory and some of its applications. We describe and evaluate two families of traditional estimation methods, namely Maximum Likelihood Estimation (MLE) and the Bayesian family of estimators. These estimators are evaluated through testing on both discrete and continuous distributions to give an indication of their strengths and weaknesses.

3.1 Estimation Theory

Estimation theory is a branch of statistics that deals with estimating the values of parameters based on empirical data. We assume that these pieces of empirical data, or observations, contain details of an information-bearing quantity. We also make the assumption that the desired information is embedded in noisy data. It is unequivocal that the empirical data at hand contains both noise and uncertainty — without uncertainty, there would be no need for estimation. The aim of this estimation process is to extract as much valuable information from the data as possible.

The Webster's Dictionary [1] defines estimation in the following manner:

> *"To form an opinion of, as to amount,, number, etc., from imperfect data, comparison, or experience; to calculate roughly."*

In the STEPS Statistics Glossary [13], the task of estimation is explained as follows:

> *"Estimation is the process by which sample data are used to indicate the value of an unknown quantity in a population. An estimate is an indication of the value of an unknown quantity based on observed data."*

The field of estimation began to mature along with the development of computers. Estimation is a tool, widely used in research problems such as signal processing, telecommunications, network intrusion detection systems, opinion polls, and, in general, in the interpretation of scientific experiments. For a detailed historical survey, we refer the reader to [57].

In the following sections we will elaborate further on the theory of estimators with regard to their different properties, and how these properties can be used for measuring the accuracy and efficiency of estimators.

3.1.1 The Properties of Estimators

The classical problem of parameter estimation can be approached in several different ways. In the parametric form, it arises when we are able to simplify the problem of estimating an unknown function $p(\boldsymbol{x})$ to the problem of estimating a set of the parameters instead — which, in turn, assumes the knowledge of the functional form of $p(\boldsymbol{x})$.

Norvig *et. al.* [45] define the task of parameter learning in the following fashion:

> *"A parameter learning task involves finding the numerical parameters for a probability model whose structure is fixed."*

The parameter estimation problem is to determine from a set of n observations, or samples, represented by the set $\mathcal{X}$, the values of parameters denoted by the vector $\boldsymbol{\theta}$.

The samples, which are the elements of $\mathcal{X}$, are individually all occurrences of the random variable $\boldsymbol{x}$. They are assumed to be obtained by selecting a state of nature ω with probability

$P(\omega)$, and then by independently selecting each of them according to the probability law $p(\boldsymbol{x}|\omega)$. When the samples are drawn this way, we say that they are independent and identically distributed (*i.i.d.*) random variables.

When we assume that $p(\boldsymbol{x}|\omega)$ has a known parametric form, we imply that it is determined uniquely by the value of a parameter vector, $\boldsymbol{\theta}$. To express the dependence on $\boldsymbol{\theta}$ explicitly, we write $p(\boldsymbol{x}|\omega)$ as $p(\boldsymbol{x}|\omega, \boldsymbol{\theta})$. This function is referred to as the probability density function (PDF) of $\boldsymbol{x}$. The PDF specifies the probability of the random variable taking on a specific value, $\boldsymbol{x} = x$, given the parameter $\boldsymbol{\theta}$.

If the individual observations are statistically independent of each other (i.i.d.), then according to the theory of probability, we can express the PDF for the data $\boldsymbol{x}$ as a product of the PDFs for each individual observation. Using this we can obtain the likelihood of $\boldsymbol{\theta}$, which is explained further in Section 3.3.1.

Our intention is to determine $\boldsymbol{\theta}$ and specify the estimate by a quantity, $\hat{\boldsymbol{\theta}}$. $\hat{\boldsymbol{\theta}}$ is itself a random vector because of the problem's probabilistic nature, which possesses its own statistical properties. It is desirable that $E[\hat{\boldsymbol{\theta}}]$ converges to the true unknown value, $\boldsymbol{\theta}$.

Variance

The variance of an estimator can be used to indicate how far, on average, the estimations are from the expected value of the estimate. It is expressed by the following equation:

$$Var(\hat{\boldsymbol{\theta}}) = E[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2], \tag{3.1}$$

where $E[\hat{\boldsymbol{\theta}}]$ is the expected value of the estimator.

Error and Standard Deviation

The estimation error, ϵ, is given by the difference between the estimate and the actual parameter value, as described in the following equation:

$$\epsilon = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}. \tag{3.2}$$

The estimation error is used to define the concept of *optimality*, which will be described further in Section 3.1.4.

The standard deviation of an estimator $\hat{\boldsymbol{\theta}}$, or an estimate of the standard deviation of $\hat{\boldsymbol{\theta}}$, is called the standard error of $\hat{\boldsymbol{\theta}}$. The standard deviation is simply the square root of the variance, as given by Eq. (3.1).

The mean-squared error (MSE) is the expected value of the square of the estimation error, and describes the amount by which the estimator differs from the quantity to be estimated. The MSE is defined as:

$$MSE(\hat{\boldsymbol{\theta}}) = E[||(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})||^2] = E[\epsilon^2], \tag{3.3}$$

where ϵ is the estimation error described in Eq. (3.2), which, expressed vectorially is also given by:

$$MSE(\hat{\boldsymbol{\theta}}) = E[\epsilon^T \epsilon]. \tag{3.4}$$

3.1.2 Bias

An estimate is said to be *unbiased* if the expected value of the estimate equals the true value of the parameter, implying that

$$E[\hat{\boldsymbol{\theta}}|\boldsymbol{\theta}] = \boldsymbol{\theta}. \tag{3.5}$$

Otherwise, the estimate is said to be *Biased*.

The bias is usually considered to be additive, and is then given by the following equation:

$$Bias(\hat{\boldsymbol{\theta}}) = E[\hat{\boldsymbol{\theta}}|\boldsymbol{\theta}] - \boldsymbol{\theta}. \tag{3.6}$$

When the estimate is biased, the bias is usually dependent on the number of observations n. We say that an estimate is *asymptotically unbiased* if the bias converges to zero when the number of observations is large (i.e. when $n \to \infty$), as described in Eq. (3.7).

$$\lim_{n\to\infty} Bias(\hat{\boldsymbol{\theta}}) = 0. \tag{3.7}$$

Eq. (3.6) leads to the following equation expressing the Mean Squared Error as a relation between the variance and the bias;

$$MSE(\hat{\boldsymbol{\theta}}) = Bias^2(\hat{\boldsymbol{\theta}}) + Var(\hat{\boldsymbol{\theta}}). \tag{3.8}$$

3.1.3 Consistency

An estimate is characterized to be *consistent* if the mean-squared error converges to zero as the number of observations increases indefinitely, as per

$$\lim_{n\to\infty} MSE(\hat{\boldsymbol{\theta}}) = 0. \tag{3.9}$$

From Eq. (3.7), we see that consistency implies that the estimate must be at least asymptotically unbiased. An estimate that is consistent is able to asymptotically recover the true parameter value that generated the data.

3.1.4 Efficiency

The quality of an estimator is generally measured by its mean squared error (MSE). To find the minimum expected variance of an unbiased estimator, we introduce the Cramér-Rao Bound (CRB), also known as the *Cramér-Rao inequality*. An efficient estimate is said to have an MSE equal to the CRB. For an efficient estimate, the *Cramér-Rao Lower Bound* (CRLB) is the greatest lower bound of the CRB. The estimate is optimum with regards to its MSE, implying that no other estimate has a smaller MSE. Also, in many cases, even though we may not have an *efficient* estimator, the CRLB still acts as a lower bound, possessing a value that is smaller than what the estimator is capable of achieving. We will not delve into the details of the CRB, but it is important to be aware of this bound, and that it exists. More information on the CRB is found elsewhere [9].

To compare two estimates, $\hat{\boldsymbol{\theta}}_\mathbf{1}$ and $\hat{\boldsymbol{\theta}}_\mathbf{2}$, we can describe the *relative efficiency* by

$$\frac{MSE(\hat{\boldsymbol{\theta}}_\mathbf{1})}{MSE(\hat{\boldsymbol{\theta}}_\mathbf{2})}, \tag{3.10}$$

where $MSE(\hat{\boldsymbol{\theta}}_\mathbf{1})$ and $MSE(\hat{\boldsymbol{\theta}}_\mathbf{2})$ are the mean-squared errors of the individual estimates. We say that an estimator that exhibits efficiency, asymptotically achieves the lowest possible variance of the parameter estimates.

3.2 Distributions

In probability and statistics, each population is identified by a corresponding PDF. Each PDF has a unique associated value of the model's parameter. Formally, we say that a model is defined by the collection of all its probability distributions, represented by the model's parameters. In this section we explain the difference between *discrete* and *continuous* distribution models. The probability distribution of a real-valued random variable $\boldsymbol{x}$, is completely characterized by its *cumulative distribution function*, which is a function specifying the probability that the random variable $\boldsymbol{x}$ is less than or equal to v, for every value v.

3.2.1 Discrete Models

A discrete probability distribution is best explained through the simple example of flipping a coin. A single coin flip of a fair coin can take values *heads* or *tails* with a probability of exactly 0.5 each. Such a distribution is called a *discrete* distribution because there are a countable number of discrete outcomes with positive probabilities. In such a case, the set of all values that the random variable can assume with non-zero probability is finite or countably infinite, where the sum of the associated probabilities converges to unity.

Let $\boldsymbol{x}$ denote a discrete random variable that can assume any one of a finite number of k different values in the set $\{v_1, \ldots, v_k\}$. Then, p_i denotes the probability of $\boldsymbol{x}$ assuming the value v_i, with

$$p_i = Pr[\boldsymbol{x} = v_i], \quad i = 1, \ldots, k. \tag{3.11}$$

The probability of all possible values must sum to unity with each $p_i \geq 0$. For discrete random variables, it is common to express the set of probabilities in terms of the *Probability Mass Function* $P(x)$, which must satisfy the following conditions:

$$P(x) \geq 0, \quad and \quad \sum_{\boldsymbol{x}} P(x) = 1. \tag{3.12}$$

One such discrete distribution is the *binomial distribution*, whose functional form is:

$$Pr(\boldsymbol{x} = i) = \binom{k}{i} a^i (1-a)^{k-i}, \tag{3.13}$$

where a is the parameter of the distribution.

An example of this distribution, and its probability mass function, is displayed in Figure 3.1.

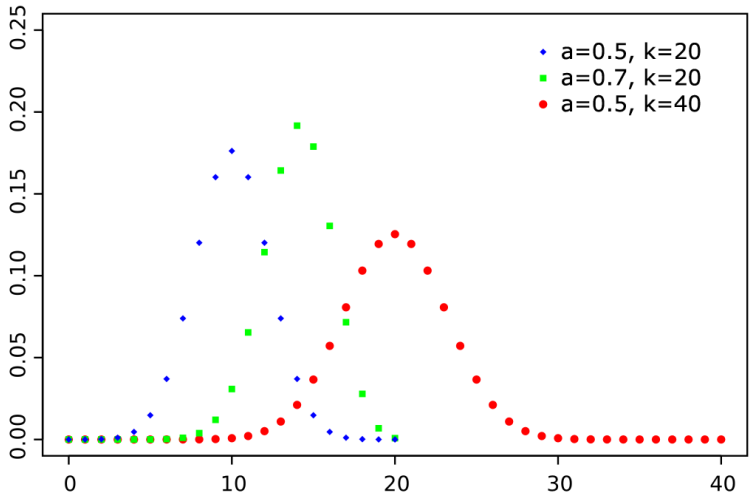

Figure 3.1: The PMF of the Binomial Distribution.

3.2.2 Continuous Models

A probability distribution is called *continuous* if its cumulative distribution function is continuous, which means that it belongs to a random variable $\boldsymbol{x}$ for which $Pr[\boldsymbol{x} = v] = 0$ for all $v \in \mathbb{R}$. In this case, rather than speaking about $\boldsymbol{x}$ assuming specific values, we talk about $\boldsymbol{x}$ falling within some interval (a, b). Moreover, instead of having a probability mass function $P(x)$, we have a *probability density function* $p(x)$, expressed in Eq. (3.14), where

$$Pr[x \in (a, b)] = \int_a^b p(x)dx. \tag{3.14}$$

It follows that this function must satisfy the following two criteria:

$$p(x) \geq 0 \quad \text{and} \quad \int_{-\infty}^{\infty} p(x)dx = 1. \tag{3.15}$$

The same definition is valid for the multivariate situation when we are dealing with continuous random vectors $\boldsymbol{X}$ instead of variables. If the components of $\boldsymbol{X}$ are statistically independent, then the joint probability density function becomes the product of all the individual densities, as in Eq.

(3.16).

$$p(\boldsymbol{X}) = \prod_{i=1}^{d} p_{\boldsymbol{x}_i}(x_i) \tag{3.16}$$

An example of a continuous distribution is the normal or Gaussian distribution, whose functional form is:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right], \tag{3.17}$$

where μ is the mean of the random variable, and σ^2 is the variance.

This distribution, and its probability density function, is displayed in Figure 3.2.

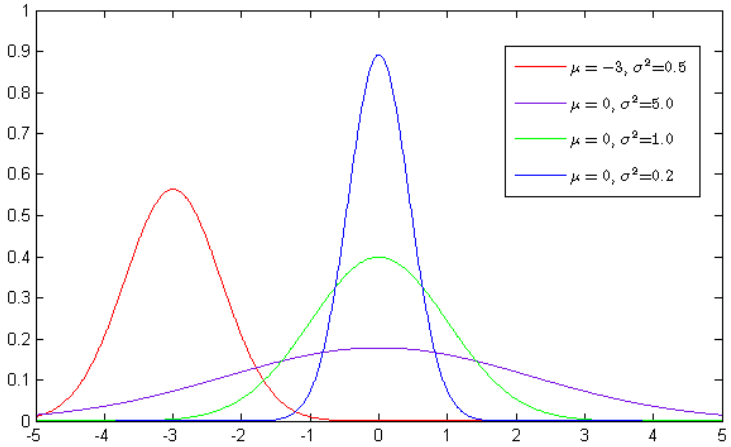

Figure 3.2: The Normal distribution PDF, plotted for different values of μ and σ^2.

3.3 Maximum Likelihood Estimation (MLE)

The goal of data analysis and parameter estimation is, generally, to identify the population that is most likely to have generated the samples at hand. The MLE has become a standard approach to parameter estimation and inference in statistics, mainly due to its many attractive attributes, described in Duda *et. al.* [12]. One of these attributes is that MLE methods nearly always have good convergence properties as the number of training samples increase. Asymptotically, for most distributions, the MLE can be said to be optimal with regard to *sufficiency*, *consistency*, *efficiency* and *parameterization invariance*. By sufficiency, we mean that complete information about the

parameter of interest is contained in its MLE estimator. By being asymptotically efficient, we imply that the estimator achieves the CRLB as $n \to \infty$. Also, another attractive attribute is that the MLE often leads to an estimate that is simpler in form than the one obtained by an alternative estimation method such as the Bayesian method described later in this chapter.

The principle of Maximum Likelihood Estimation was pioneered by Fisher in the 1920s, and has a wide range of applications. It is an indispensable tool for many statistical modeling techniques, and in particular, in non-linear modeling with non-normal data.

Before we describe the technique, it is important to note that ML estimates need not exist nor be unique. For simplicity, we now show how ML estimates can be computed when we know that they exist and that they are unique.

If each sample in $\mathcal{X}$ is drawn from an *i.i.d.* random variable obeying the PDF $p(\boldsymbol{x}|\omega, \boldsymbol{\theta})$, our problem is to use the information provided by the observations to obtain good estimates for the unknown parameter vector, $\boldsymbol{\theta}$. The problem can be formulated as follows[12]: From a set of training samples $\mathcal{X}$, drawn independently from the probability density $p(\boldsymbol{x}|\boldsymbol{\theta})$, estimate the unknown parameter vector, $\boldsymbol{\theta}$.

With n samples in $\mathcal{X}$, namely, $\{x_1, \ldots, x_n\}$, we then consider the following function, known as the likelihood function:

$$p(\mathcal{X}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(x_k|\boldsymbol{\theta}). \tag{3.18}$$

Viewed as a function of $\boldsymbol{\theta}$, $p(\mathcal{X}|\boldsymbol{\theta})$ is called the likelihood of $\boldsymbol{\theta}$ with respect to the set of samples, $\mathcal{X}$. Furthermore, the *maximum likelihood estimate* of $\boldsymbol{\theta}$ is defined as the value $\hat{\boldsymbol{\theta}}$ that maximizes $p(\mathcal{X}|\boldsymbol{\theta})$.

Since we will be dealing with binary variables, we will exemplify the MLE using a series of Bernoulli trials[1] (e.g. tossing a coin repeatedly). To achieve this, we consider the case where we have only a single parameter, in which the probability for head (or success in our Bernoulli trial) is given by θ. Then, for a series of n Bernoulli trials, the probability density function is given by the following equation:

$$P(x|n, \theta) = \frac{n!}{x!(n-x)!}\, \theta^x (1-\theta)^{n-x}, \tag{3.19}$$

where $x = 0, 1, \ldots, n$ and $0 \leq \theta \leq 1$. The PDF described in Eq. (3.19) is also known as the

[1] A Bernoulli trial is an experiment whose outcome is random and can be either of two possible outcomes, "success" and "failure".

general PDF of the *binomial distribution* for arbitrary values θ and n.

3.3.1 Likelihood Function

The task of maximum likelihood estimation boils down to the following: Find the PDF, among all the possible PDFs of the model, that is most likely to have produced the observed data. The likelihood function is defined by "reversing" the parameter and data vector in the probability density function $p(\boldsymbol{x}|\boldsymbol{\theta})$, as explained below.

Indeed, this function is, in fact, the same likelihood function described for the general case in Eq. (3.18). $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x})$ represents the likelihood of the parameter $\boldsymbol{\theta}$, having caused the data $\mathcal{X}$. An alternative notation, is shown in Eq. (3.20).

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{\theta}), \tag{3.20}$$

whence the MLE of $\boldsymbol{\theta}$ is inferred.

Proceeding with this argument for the binomial distribution and its PDF given in Eq. (3.19), we get the following likelihood function:

$$\mathcal{L}(\theta|n, x) = P(x|n, \theta) = \frac{n!}{x!(n-x)!}\,\theta^x(1-\theta)^{n-x}. \tag{3.21}$$

The significant difference here, is that the likelihood function is defined on the parameter scale, as opposed to the PDF that is a function of the data, which when given the parameters, is defined on the data scale. This is explained through the following example with $n = 10$ and $\theta = 0.7$. In this case, we get the following PDF:

$$P(x|n = 10, \theta = 0.7) = \frac{10!}{x!(10-x)!}(0.7)^x(0.3)^{10-x}, \tag{3.22}$$

and the following likelihood function for a given observed value, where, for example $x = 7$:

$$\mathcal{L}(\theta|n = 10, x = 7) = p(x = 7|n = 10, \theta) = \frac{10!}{7!3!}\,\theta^7(1-\theta)^3. \tag{3.23}$$

The binomial distribution for this example is displayed in Figure 3.3.

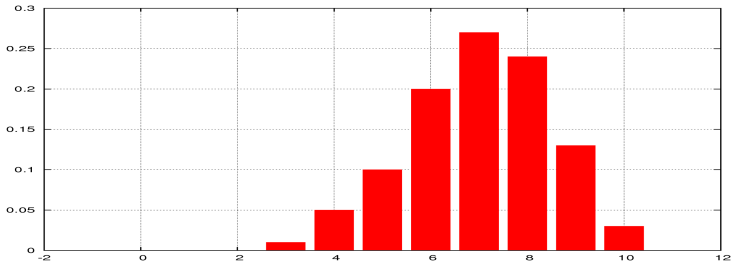

Figure 3.3: The Binomial probability distribution for $n = 10$ and $\theta = 0.7$.

This example exemplifies the case when we have only a single parameter, θ, (beside n, which is assumed to be known). If we deal with models with more than one parameter (for example, k parameters), the likelihood function will take the shape of a k-dimensional surface instead of a simple curve. *Myung* used the following explanation in [40]:

> *"In general, for a model with k parameters, the likelihood function $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x})$ takes the shape of a k-dim geometrical "surface" sitting above a k-dim hyperplane spanned by the parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$."*

3.3.2 Likelihood Equation

By using the observed data and the likelihood function of a model given the data, we can make statistical inference about the probability distribution that underlies the data. In the MLE case, this means that we must find the value of the parameter vector $\boldsymbol{\theta}$ that maximizes the likelihood function $\mathcal{L}(\boldsymbol{\theta}|\mathcal{X})$. Formally, the MLE is defined in the following manner:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathcal{X}). \tag{3.24}$$

For our example (Eq. (3.22) and (3.23)) we get the following MLE estimate $\hat{\theta} = 0.7$, as the corresponding maximized likelihood value $\mathcal{L}(\theta = 0.7|n = 10, x = 7) = 0.267$. This implies that $\theta = 0.7$ yields the highest likelihood, or is most likely, to have generated our observed data value of $x = 7$.

For computational convenience, it is usually easier to work with the logarithm of the likelihood rather than the likelihood itself. The estimate that maximizes the log-likelihood also maximizes the likelihood, since the two of them (the log-likelihood and the likelihood) are monotonically

related to each other. If we assume that the likelihood function is a differentiable function of $\boldsymbol{\theta}$, then the estimate $\hat{\boldsymbol{\theta}}$ can be found by the standard methods of differential calculus. We define this log-likelihood function, $l(\boldsymbol{\theta})$, for the general case in the following manner[2]:

$$l(\boldsymbol{\theta}) \equiv \ln p(\mathcal{X}|\boldsymbol{\theta}), \tag{3.25}$$

resulting in the following MLE definition

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \ln p(\mathcal{X}|\boldsymbol{\theta}). \tag{3.26}$$

Here, the dependence on the data set $\mathcal{X}$ is implicit. Expanding the PDF, we can now express it as a sum, rather than a product of all individual PDFs, since we are using the log-likelihood. This gives us the following log-likelihood function:

$$l(\boldsymbol{\theta}) = \sum_{k=1}^{n} \ln p(x_k|\boldsymbol{\theta}). \tag{3.27}$$

It can be shown that the MLE for $\boldsymbol{\theta}$ in Bernoulli trials is, in fact, the same as the one obtained by the method of moments (i.e., the estimate obtained by counting the relative frequency of successes), which can be expressed in the following form:

$$\hat{\boldsymbol{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x_i}. \tag{3.28}$$

It is important to bear in mind that our solution $\hat{\boldsymbol{\theta}}$ is an estimate, and that it is only in the limit of an infinitely large number of samples that we can expect the estimate to equal the true value of the generating function.

3.3.3 MLE Bias

For most distributions, the MLE is asymptotically unbiased. As we defined earlier, this means that the bias tends to zero as the number of samples increases indefinitely. In most PR problems with large training sets, we would like to use asymptotically unbiased estimators.

[2]The base of the logarithm can be chosen for convenience, but generally, in most analytical problems, base e is most appropriate.

3.3.4 Sliding Window

Traditional PR systems assume that the class-conditional distributions are stationary, and thus that they do not change with time. In this Thesis we also consider the case when the distribution is non-stationary. When dealing with non-stationary environments, the MLE performs poorly since it make use of all samples when estimating the parameters. The traditional strategy to deal with this has been one of using a *sliding window* [23]. The crucial part of the sliding window approach is the window-width parameter. If the width is too "small", the estimates are necessarily poor, and on the other hand, if the width is "large", estimates prior to the change in the environment will influence the new estimates significantly.

We will compare the MLE and the windowed variant of the MLE (MLEW) with regard to their performance in non-stationary environments in Ch. 5.

3.3.5 Model Error

If we have a reliable model for the underlying probability distributions, and their dependence on the parameter vector $\boldsymbol{\theta}$, the ML estimator can give excellent results. But if our assumption for the underlying model is inappropriate, our estimation will give us a sub-optimal solution. As an example, consider the case of assuming a Gaussian distribution as the model, but when instead, the actual signal is produced from a doubly-exponential distribution. This would result in values of θ that are not the best solution available. This points out the need for reliable information concerning the underlying models; If the assumed model is poor, we cannot be assured that the derived results from our estimator is the best.

3.4 Bayesian Estimation

The Bayesian approach to estimation and statistical learning, makes use of hypotheses and training data. Related to estimation, the method uses the prior data as evidence to obtain a "prior" distribution (or description) of the domain, which is then used to obtain the current estimate. The hypotheses on the other hand, are probabilistic theories of how the domain works. This Section gives a brief overview of the Bayesian family of estimators.

The conceptual difference between ML methods and the Bayesian methods is that in the MLE

case, we view the true parameter sought for, $\boldsymbol{\theta}$, to be fixed, whereas in Bayesian estimation, we consider it to be a random variable. The training data is used to learn the *posterior probability density* of the random variable.

3.4.1 Bayesian Learning

As stated in [45], Bayesian learning simply calculates the probability of each hypothesis, given the data, and makes predictions based on the samples. The core of the Bayesian approach is the posterior probability $p(\boldsymbol{\theta}|\mathcal{X})$. The name Bayesian estimation, or Bayesian learning, comes from the central role that Bayes formula has in this approach, named after Thomas Bayes. Bayes rule is:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}, \tag{3.29}$$

where A and B are events in their corresponding event spaces.

In Bayesian PR, we apply this rule to compute the posterior probabilities, $P(\omega_i|x)$, based on the *prior probabilities*, $P(\omega_i)$, and the class-conditional probability density function, $p(x|\omega_i)$. The problem is that these values are generally unknown. The approach we use is to compute $P(\omega_i|x)$ using all the information we have available. We seek this information in the set of training data we have available, namely, $\mathcal{X}$. We emphasize the dependence on these samples by explicitly writing $P(\cdot)$ as $P(\omega_i|x, \mathcal{X})$. By using these probabilities, and Bayes formula, we can derive the Bayes classifier.

$$P(\omega_i|x, \mathcal{X}) = \frac{p(x|\omega_i, \mathcal{X})P(\omega_i|\mathcal{X})}{\sum_{j=1}^{c} p(x|\omega_j, \mathcal{X})P(\omega_j|\mathcal{X})}, \tag{3.30}$$

when we can separate the training samples by class into c subsets $\mathcal{X}_1, \ldots, \mathcal{X}_c$, with the samples in $\mathcal{X}_i$ belonging to ω_i.

It is explained in detail in [45, Ch.20] that

> *"the true hypothesis eventually dominates the Bayesian predication. For any fixed prior that does not rule out the true hypothesis, the posterior probability of any false hypothesis will eventually vanish, simply because the probability of generating "uncharacteristic" data indefinitely is vanishingly small."*

Duda *et. al.* describe the information that may be provided in the training set $\mathcal{X}$ in the following way:

> *"Part of this information might be prior knowledge, such as knowledge of the functional forms for unknown densities and ranges for the values of unknown parameters."*

From Eq. (3.30), we see that the information provided by the samples can be used to calculate both the class-conditional densities and the prior probabilities. We make the assumption that the true values of the prior probabilities are known (or obtainable by a trivial calculation), and therefore write $P(\omega_i|\mathcal{X})$ as $P(\omega_i)$. As stated earlier for the case of maximum likelihood, we assume that the samples contained in $\mathcal{X}$ are *i.i.d.* Using this, and our assumption of known prior probabilities, we can simplify Bayes formula and express it in the following manner:

$$P(\omega_i|x,\mathcal{X}) = \frac{p(x|\omega_i,\mathcal{X}_i)P(\omega_i)}{\sum_{j=1}^{c} p(x|\omega_j,\mathcal{X}_j)P(\omega_j)}. \tag{3.31}$$

Moreover, because of each sample is assumed to be *i.i.d.*, we now have c separate problems of the following form: From a set of observations or samples $\mathcal{X}$ drawn independently according to the fixed, but unknown probability distribution $p(x)$, determine $p(x|\mathcal{X})$.

3.4.2 Bayesian Parameter Estimation

As before, we assume that the unknown probability density $p(x)$ has a known parametric form. We are interested in the unknown value of the parameter vector $\boldsymbol{\theta}$. To express the dependence of $p(x)$ on $\boldsymbol{\theta}$, we say that $p(x|\boldsymbol{\theta})$ is completely known. The prior information (hypothesis) about $\boldsymbol{\theta}$, is kept in the known prior density $p(\boldsymbol{\theta})$. Through the observations $\mathcal{X}$, this is converted into a *posterior* density $p(\boldsymbol{\theta}|\mathcal{X})$. The idea here is that this density, will form a sharp peak about the true value of the parameter vector $\boldsymbol{\theta}$. The learning problem is now posed as a problem of estimating the parameter vector $\boldsymbol{\theta}$.

The goal is to compute $p(x|\mathcal{X})$[3]. This is done by integration and is explained in Eq. (3.32).

$$p(x|\mathcal{X}) = \int p(x,\boldsymbol{\theta}|\mathcal{X})d\boldsymbol{\theta}. \tag{3.32}$$

[3]Here x is the observed sample which has to be classified.

The integration here spans the entire parameter space. Probability theory allows us to write $p(x, \boldsymbol{\theta}|\mathcal{X})$ as the product of $p(x|\boldsymbol{\theta}, \mathcal{X})p(\theta|\mathcal{X})$. Furthermore, since the selection of x and that of the samples in $\mathcal{X}$ is done independently, the first factor is simply $p(x|\boldsymbol{\theta})$. When we know the parameter vector, the distribution of x is known completely. We now rewrite the integration of the joint density in the following way:

$$p(x|\mathcal{X}) = \int p(x, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{X})d\boldsymbol{\theta}. \tag{3.33}$$

If $p(x|\mathcal{X})$ peaks very sharply about some value $\hat{\boldsymbol{\theta}}$, we obtain $p(x|\mathcal{X}) \approx p(x|\hat{\boldsymbol{\theta}})$. This is the result that we would obtain by substituting the estimate $\hat{\boldsymbol{\theta}}$ for the true parameter value. The integral can then be computed numerically by using for example Monte-Carlo simulation. The result we obtain here is based on the assumption that $p(x|\boldsymbol{\theta})$ is smooth.

3.4.3 General Solution

The posterior density is given by Bayes formula

$$p(\boldsymbol{\theta}|\mathcal{X}) = \frac{p(\mathcal{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}, \tag{3.34}$$

where $p(\mathcal{X}|\boldsymbol{\theta})$ is given through the independence assumption defined as

$$p(\mathcal{X}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(x_k|\boldsymbol{\theta}). \tag{3.35}$$

This is our general solution to the posed estimation problem. These equations also show the relation to the MLE solution described in Section 3.3.

If $p(\mathcal{X}|\boldsymbol{\theta})$ peaks sharply at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$, and the prior density is not zero at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$, and does not change much in its surrounding neighborhood, then $p(\boldsymbol{\theta}|\mathcal{X})$ will also peak at the same point, thus proving our statement from earlier that $p(x|\mathcal{X})$ approximates $p(x|\hat{\boldsymbol{\theta}})$. This would also be the result from using the MLE as if it was the true value.

A special case occurs when the prior probability distributions and the posterior distributions have the same functional form. This is called the concept of conjugate priors and was introduced by Raiffa & Schlafier in [51]. Using conjugate prior makes the calculation of the posterior simple

and makes the estimation process intuitive. It is especially useful for sequential estimation, where the posterior of the current iteration is used as the prior of the next iteration. For further detail on carrying out the computations, we refer the reader to [12, Ch.3].

In the next chapter we investigate a learning based approach to estimation, referred to as the Stochastic Learning Weak Estimator (SLWE).

Chapter 4

Learning-Based Estimators

In the previous chapter we investigated two well-known methods of estimation, namely the Maximum Likelihood and the Bayesian estimates. They are known for having both good computational and statistical properties, but work with the premise that the parameters being estimated does not change with time, i.e., the distribution is assumed to be *stationary*. Generally, we assume that there is an underlying parameter vector $\boldsymbol{\theta}$, which do not change with time, and as the number of samples increases, we want the estimate $\hat{\boldsymbol{\theta}}$ to converge to $\boldsymbol{\theta}$ with probability 1, or in the sense of its mean square error, as described in Section 3.1.4.

We will now investigate the scenario where the parameters being estimated do, indeed, change with time. Consider, for example, the case when the parameter of a binomial distribution is "switched" periodically. Such a scenario demonstrates the behavior of what we call a *non-stationary* environment. The goal of an estimator scheme in this case would thus be to estimate the parameter, and to be able to adapt to any changes occurring in the environment, e.g., the algorithm must be able to detect the changes and estimate the new parameter after a "switch" has occurred in the environment. To illustrate this, we consider the following example: We have a binomial distribution in which the true parameter is initialized to $\theta = 0.7$, meaning that our Bernoulli trials will give the output 1 w.p. 0.7 (and 2 w.p. 0.3). After 60 time steps, the environment changes this parameter to $\theta = 0.2$. Now, the Bernoulli trials result in the output 1 w.p. 0.2. The task for our estimator is to track this change as fast as possible, and to make a correct estimate both before and after the change takes place.

With this as our motivation, we will now investigate a learning-based approach to estimation. The method is referred to as the *Stochastic Learning Weak Estimator* (SLWE), and is a novel

estimation method presented by Oommen and his co-authors in [46]. The estimate is based on the principles of stochastic learning, and the convergence of the estimate is *weak*, i.e., with regard to the first and second moments. Analogous to stochastic learning, the SLWE possess a user-defined *learning coefficient*, λ. Both the estimate variance and the speed decrease with this learning coefficient.

Oommen *et. al.* point out one significant property of the SLWE:

> *"(...) our new estimators can be shown to converge to the true value fairly quickly, and just as quickly "unlearn" what they have learned so as to adapt to the new "switched" environment."*

In Section 3.3.4, we introduced the concept of a sliding window to overcome the challenges faced in non-stationary environments for the traditional estimation methods. The problem with this is that if the width of the window is too small, the corresponding estimates tend to be poor. We also have a similar problem if we choose large window. In this case, the estimates prior to the change of the parameter have too much influence on the new estimates. Another downside with the sliding window approach is that the observations during the entire window width must be maintained and updated during the process of estimation. If we use a window size of 40 time steps, each of these observations must be kept, and when adding a new observation, the oldest observation must be deleted, and the most recent sample must take its place.

As stated in the previous quotation, this problem is avoided with the SLWE. The SLWE makes use of an updating scheme that is *multiplicative* and not *additive*. It should be noted that many other methods have been proposed as an alternative to the traditional sliding window approach in non-stationary environments. Oommen *et. al.* mention several such methods in [46], but also state that these methods are quite different from the SLWE approach. One of the methods mentioned is an approach that involves estimating the log-normal multivariate distribution in non-stationary environments, proposed by Ho *et. al.* in [19]. Another approach is the sequential change-point detection and estimation method, which is based on approximations from large samples and the null hypothesis. This was proposed by Gombay in [17]. In the interest of brevity, we do not delve into the details of these methods here. Oommen and his co-author observe that the fundamental difference between these methods and the SLWE approach, is that the updating scheme used by the SLWE is multiplicative and not additive.

4.1 Stochastic Learning-Based Weak Estimator (SLWE)

The SLWE maintains a running estimate that is updated at every time instant, based on the value of the current observation. As states in [46]:

> *"Based on the actual event that has occurred at any particular time instant, our estimation procedure multiplies the current estimate of the parameter with a multiplying constant, and thus this multiplication operation is random."*

The multiplicative updating rule used by the SLWE is similar to the action-probability updating rule used by reinforcement learning schemes as described by Narendra and Thathachar in [42, 43].

In this chapter we will examine the properties of the SLWE with regard to efficiency and accuracy. When we discuss efficiency, we here talk about the estimators rate of convergence to the real value, or in the mean-square sense. In the following sections we will show that the SLWE converges *weakly*. If the estimator is "*strong*" it means that it converges with probability equal to 1. We will show that it is sub-optimal to work with strong estimators when the data is truly non-stationary. The main motivation for this is that with strong estimators, it is unlikely that the learned parameter (or phenomenon) will change rapidly from that which it has converged to. We will prove mathematically and empirically that the SLWE converges weakly, and that the convergence is independent of the value of the learning coefficient, λ. Moreover, we will show that the *rate of convergence*, is determined fully by λ. With regard to estimator accuracy, we will show how the variance of the estimate depends on the same learning coefficient. In the following sections we will introduce the same theorems as Oommen *et. al.* presented in [46]. The theorems will be validated and proved for both binomial and multinomial distributions. The analysis that we will include involves determining the updating equations, and then taking the conditional expectation of the quantity the we analyze. By taking the expectation a second time, this condition disappears. This again will lead to a difference equation which will be explicitly solved and used in proving the established theorems.

Before we investigate weak estimators of binomial and multinomial distributions, we dedicate a section to Markov chains and the Markovian property.

4.2 Markov Chains And The Markovian Property

As we have stated, the SLWE exhibits asymptotic behavior, i.e., it converges weakly, in the limit (e.g. as time $n \to \infty$.) In this section we will describe systems modeled by so-called Markov chains, which also exhibit an asymptotic behavior. Closely related to the analysis of the SLWE's convergence properties, are also the properties of eigenvectors and eigenvalues. We introduce Markov chains and the Markovian property in this section so as to make the proofs later in this chapter easier to understand.

Markov chains was pioneered by the Russian mathematician Andrei Andreevich Markov in the early 1900's [36]. They are discrete-parameter stochastic processes that satisfy the Markovian property. Having the Markovian property means that, given the present state, future states are independent of the past states. In other words, the present state fully encapsulates all the information that influences any future states of the Markov process. We say that, given the current state, the future is conditionally independent of the past. Markov chains are discussed in detail in [38], where the reader can also find proofs and calculations.

We shall see later in this chapter that the analysis of the SLWE involves a stochastic matrix, $\mathbf{M}$, which exhibits a Markovian behavior. Since this is true, it is imperative that we look into the properties of Markov chains. This will help us later on in analyzing the properties of $\mathbf{M}$, and in particular, its eigenvalue and eigenvector properties.

A Markov chain is called an *ergodic* chain if it is possible to go from every state to every other state, but not necessarily in a single move. Such a chain is referred to as an *irreducible* Markov chain. A state in the chain is called *absorbing* if it is impossible to leave this state once it has been reached there. If there is a non-zero probability that we will never return back to a given state ϕ_i, after leaving it, then this state is said to be *transient*. If a state is not transient, it is called *recurrent* or *persistent*. If the expected return time to a given state is finite, then the state is said to be *positive recurrent*. A state is called *aperiodic* if it is not recurring at regular intervals, e.g. with a given period $k = 1$. Otherwise, if $k > 1$, the state is said to be periodic with period k.

A state ϕ_i is said to be ergodic if it is both *aperiodic* and *positive recurrent*. If all the states in the Markov chain are ergodic, then the chain is said to be ergodic.

A Markov chain is called a *regular* chain if some power of the transition matrix has only positive elements. In other words, for some n, it is possible to go from any state to any state in exactly n steps. It is clear from this definition that every regular chain is ergodic.

Figure 4.1 shows a simple three-state Markov chain, with the corresponding Markov matrix given below, in Eq. (4.1).

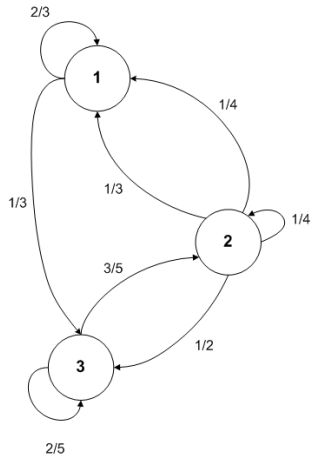

Figure 4.1: Example Markov chain

$$\mathbf{B} = \begin{bmatrix} 2/3 & 0 & 1/3 \\ 1/4 & 1/4 & 1/2 \\ 0 & 3/5 & 2/5 \end{bmatrix}. \tag{4.1}$$

The Markov matrix displays a fundamental property of Markov chains, namely that each row *sum* is unity, i.e., $\sum_{j=1}^{d} b_{ij} = 1$, where b_{ij} corresponds to the element in row i and column j. Each of these entries in the Markov matrix, correspond to a given *state transition probability* $b_{ij} = Pr[\phi(n+1) = \phi_j | \phi(n) = \phi_i]$.

We define $\pi_i(n)$ as the probability of being in state ϕ_i at time n. Then,

$$\mathbf{B}^T \boldsymbol{\pi}(n) = \boldsymbol{\pi}(n+1). \tag{4.2}$$

If we are in state $[1, 0, 0]^T$ at $\boldsymbol{\pi}(n)$, then

$$\boldsymbol{\pi}(n+1) = \begin{bmatrix} 2/3 & 0 & 1/3 \\ 1/4 & 1/4 & 1/2 \\ 0 & 3/5 & 2/5 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 0 \\ 1/3 \end{bmatrix}. \tag{4.3}$$

If $\boldsymbol{\pi}(n+1) = \mathbf{B}^T\boldsymbol{\pi}(n)$, and if $\boldsymbol{\phi}$ converges as $n \to \infty$, then $\boldsymbol{\pi}(\infty) = \mathbf{B}^T\boldsymbol{\pi}(\infty)$. This implies that $\boldsymbol{\pi}(\infty)$ is the *eigenvector* of *eigenvalue*[1] $\xi = 1$ of the transposed Markov matrix, $\mathbf{B}^T$.

To verify this, we need to investigate the following, important class of linear equations [28]:

$$\mathbf{B}^T\boldsymbol{y} = \xi\boldsymbol{y}, \tag{4.4}$$

the so-called eigenvector-eigenvalue equation, where $\mathbf{B}$ is the d-by-d matrix and ξ is a scalar eigenvalue associated with the eigenvector $\boldsymbol{y}$. The eigenvalue equation can be rewritten in the following fashion:

$$(\mathbf{B}^T - \xi\boldsymbol{I})\boldsymbol{y} = \mathbf{0}\,, \tag{4.5}$$

where, $\boldsymbol{I}$ is the identity matrix and $\mathbf{0}$ is the zero vector. We will see later that this equation is important in our analysis of the SLWE. The solution vector $\boldsymbol{y} = \boldsymbol{e_i}$ and the corresponding scalar $\xi = \xi_i$ are called the *eigenvector* and associated *eigenvalue*, respectively. If $\boldsymbol{B}$ is *real*, there are d (possibly non distinct) solution vectors $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_d}\}$, each with its respective eigenvalue $\{\xi_1, \xi_2, \ldots, \xi_d\}$.

The transition matrix for which we are finding the eigenvalue and its corresponding eigenvector must have been set up so that the columns sum to 1. This is why we deal with the transpose $\boldsymbol{B}^T$ instead of $\boldsymbol{B}$ itself, since we know that the rows of $\boldsymbol{B}$ sum to unity.

When $\xi = 1$, $\mathbf{B}^T\boldsymbol{\pi}^* = \boldsymbol{\pi}^*$, which leads us to the scenario where the eigenvalue $\equiv 1$ and the eigenvector $\equiv \boldsymbol{\pi}(\infty)$. $\boldsymbol{\pi}^*$ is the vector to which $\boldsymbol{\pi(n)}$ converges to as n grows. Thus, to solve the Markov chain, we know what to look for, namely for the eigenvector of $\xi = 1$. We also know that this is independent of the initial state. This gives us the result that every irreducible ergodic Markov chain has one eigenvalue whose magnitude is unity. Since $\boldsymbol{B}$ is real, it has d eigenvalues, and since the chain is irreducible, exactly one of these is unity. The remaining eigenvalues come either in complex conjugate pairs (on either side of the imaginary axis, i.e. as $a \pm \mathbf{i}b$), or as real values. This is in accordance to the *Perron-Frobenius* theorem which deals with positive stochastic matrices [16, 49].

From our example Markov chain, we can show that $\boldsymbol{\pi}(\infty) = [1/4, 1/3, 5/12]^T$, independent of $\boldsymbol{\pi}(0)$.

This vector, $\boldsymbol{\pi}(\infty)$, that the system converges to, is often called the *long run distribution*,

[1] Please note that we here use the notation ξ for the eigenvalues instead of the more commonly used λ notation. We do this because λ is used to represent the learning parameter of the SLWE. We will show that all non-unity eigenvalues of the stochastic matrix of the SLWE are in fact equal to this learning parameter.

the *invariant distribution* or the *stationary distribution* of the Markov chain. Stating that $\boldsymbol{\pi}(\infty)$ is *independent* of $\boldsymbol{\pi}(0)$ implies that there is no constraint on the starting distribution; the chain converges to the stationary distribution regardless of where it begins.

From our example vector, we further investigate the form of one of its elements, namely $\pi_3(n)$.

$$\pi_3(n) = \frac{5}{12} + k_2(0.4876)^n + k_3(-0.1709)^n, \tag{4.6}$$

where k_2 and k_3 are some scalars of the third components of the eigenvectors $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$, respectively. The scalar constants are determined by the initial state of the Markov chain, $\boldsymbol{\pi}(0)$. The scalar constant k_1, associated with the largest eigenvector $\boldsymbol{e}_1$, is unity, and is therefore left out of the equation, due to the nature of $\boldsymbol{\pi}$ being a probability vector where the elements sum to unity.

In this example, $\mathbf{B}$ has three eigenvalues; $\begin{cases} \xi_1 = 1 \\ \xi_2 = 0.4876 \\ \xi_3 = -0.1709 \end{cases}$

The general form of $\boldsymbol{\pi}(n)$ is given in the following equations:

$$\pi_1(n) = A_{11}\xi_1^n + c_2 A_{12}\xi_2^n + c_3 A_{13}\xi_3^n. \tag{4.7}$$

$$\pi_2(n) = A_{21}\xi_1^n + c_2 A_{22}\xi_2^n + c_3 A_{23}\xi_3^n. \tag{4.8}$$

$$\pi_3(n) = A_{31}\xi_1^n + c_2 A_{32}\xi_2^n + c_3 A_{33}\xi_3^n. \tag{4.9}$$

If we order the eigenvalues, so that $\xi_1 = 1$ and $\xi_2, \xi_3 < 1$, then, as $n \to \infty$,

$$\boldsymbol{\pi}(\infty) = \begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$$

Note that the A_{ij} values correspond to the components of the j'th eigenvector ($\boldsymbol{y} = \boldsymbol{e}_j$) for the j'th eigenvalue (ξ_j), and c_i is some constant scalar determined by the initial state, $\boldsymbol{\pi}(0)$, of the Markov chain. In other words, $\boldsymbol{\pi}(\infty)$ is the eigenvector for eigenvalue $\xi = 1$. To obtain the solution, we must simply calculate this eigenvector. For a detailed explanation of this calculation, we refer the reader to [28, Ch.4]. From this we observe that the rate of convergence in the asymptotic system does not depend on $\xi_1 = 1$, but on the other eigenvalues, and in particular on the second largest eigenvalue, ξ_2.

We will now investigate weak estimators of binomial distributions.

4.3 Weak Estimators of Binomial Distributions

To exemplify the strengths of the SLWE, we will use the binomial distribution, or Bernoulli distribution, introduced in Ch. 3. The binomial distribution is characterized by two parameters, namely, the *number* of Bernoulli trials, and the Bernoulli parameter, which characterize *each* Bernoulli trial. We assume that the number of observations is the number of trials. We seek to estimate the *Bernoulli* parameter for each trial.

Let $\boldsymbol{x}$ be a binomially distributed random variable, which takes on the value of either "1" or "2". We choose to use these values instead of the more common "0" or "1" notation, to make the theorems and proofs introduced here consistent with the theorems and proofs for the multinomial case, that will be introduced later in this chapter. It is assumed that the distribution of $\boldsymbol{x}$ is characterized by the parameter $\boldsymbol{\theta}$, where $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$. This means that

$$\begin{aligned} \boldsymbol{x} &= \text{'1' w.p. } \theta_1 \\ &= \text{'2' w.p. } \theta_2\,, \end{aligned}$$

where the probabilities, θ_1 and θ_2, sum to unity. With these probabilities, we let $x(n)$ denote the *concrete realization* of $\boldsymbol{x}$ at time n. We want to estimate $\boldsymbol{\theta}$, i.e., θ_1 and θ_2. The SLWE maintain a running estimate $P(n) = [p_1(n), p_2(n)]^T$ of $\boldsymbol{\theta}$, where p_i represent the estimate of θ_i at time n, for $i = 1, 2$. $p_1(n)$ is then updated as follows:

$$p_1(n+1) \leftarrow \lambda p_1(n) \qquad \text{if } x(n) = 2 \tag{4.10}$$

$$\leftarrow 1 - \lambda p_2(n) \quad \text{if } x(n) = 1\,, \tag{4.11}$$

where λ is the user-defined learning parameter, $0 < \lambda < 1$, and

$$p_2(n+1) \leftarrow 1 - p_1(n+1). \tag{4.12}$$

The above rule is analogous to the L_{RI} reinforcement learning scheme known from the field of learning automata [42].

Note that the running estimate, P, is the equivalent of $\hat{\boldsymbol{\theta}}$, introduced in Ch. 3, but, for simplicity, we use $P(n)$ or P in this Chapter. We omit the time index, n, whenever this does not lead to confusion, and thus, in such cases equivalently using P and $P(n)$. In the following Theorem, we will prove that the mean of P, obtained through Eqs. (4.10) and (4.11), converges exactly to $\boldsymbol{\theta}$ as the number of samples grows large. From Ch. 3 we know that such behavior defines an asymptotically unbiased estimator.

Theorem 1. Let $\boldsymbol{x}$ be a binomially distributed random variable, and $P(n)$ be the estimate of $\boldsymbol{\theta}$ at time n. Then, $\mathrm{E}\left[P(\infty)\right] = \boldsymbol{\theta}$.

Proof. The conditional expected value of $p_1(n+1)$ given $P(n)$ can be expressed in the following manner, based on the updating scheme given in Eqs. (4.10) and (4.11):

$$\begin{aligned} \mathrm{E}\left[p_1(n+1)|P(n)\right] &= \lambda\theta_2 p_1 + \theta_1 - \lambda\theta_1 + \lambda\theta_1 p_1 && (4.13)\\ &= \lambda p_1\theta_2 + \lambda p_1\theta_1 + (1-\lambda)\theta_1 && (4.14)\\ &= (1-\lambda)\theta_1 + \lambda p_1(\theta_1+\theta_2)\,. && (4.15) \end{aligned}$$

Since $\theta_2 + \theta_1 = 1$, we can simplify Eq. (4.15) as follows:

$$\mathrm{E}\left[p_1(n+1)|P(n)\right] = (1-\lambda)\theta_1 + \lambda p_1\,. \tag{4.16}$$

If we now take the expectation a second time, we eliminate the condition and can write Eq. (4.16) as

$$\mathrm{E}[p_1(n+1)] = (1-\lambda)\theta_1 + \lambda\mathrm{E}[p_1(n)]\,. \tag{4.17}$$

As $n \to \infty$, both $\mathrm{E}[p_1(n)]$ and $\mathrm{E}[p_1(n+1)]$ converge to $\mathrm{E}[p_1(\infty)]$. The reason for this is that the coefficient of the linear difference equation is λ, with $0 < \lambda < 1$, and thus making $\mathrm{E}[p_1(n)]$ converge to a limit. Then, by solving for $\mathrm{E}[p_1(\infty)]$ in Eq. (4.17), we get:

$$\mathrm{E}[p_1(\infty)](1-\lambda) = (1-\lambda)\theta_1\,, \tag{4.18}$$

which implies that $\mathrm{E}[p_1(\infty)] = \theta_1$ and $\mathrm{E}[p_2(\infty)] = \theta_2$, thus proving the theorem. □

We will now examine the mean of the limiting distribution of the vector $P(n)$ and its rate of convergence. It is important to bear in mind that $P(n)$ converges to a distribution, and not a single value. We are interested in the mean of this distribution. We will show that $P(n)$ converge exactly to the mean of $\boldsymbol{\theta}$. The rate of convergence is determined fully by the learning parameter, λ, but the mean of the limiting distribution is independent of λ. The results we will show are asymptotic, meaning that the estimates converge to the true value, only in the limit. The implications of this will be explained presently, and we will see that the SLWE is capable of converging with a satisfactorily low variance after a relatively small number of time steps. The results that follows also indicate how $\mathrm{E}[P(n+1)]$ and $\mathrm{E}[P(n)]$ are related to each other by means of a stochastic

matrix.

Theorem 2. If the components of $P(n+1)$ are obtained from the components of $P(n)$ according to Eqs. (4.10) and (4.11), the expectation of $P(n+1)$ is related to the expectation of $P(n)$ as $\mathrm{E}\left[P(n+1)\right] = \mathbf{M}^T \mathrm{E}\left[P(n)\right]$, where $\mathbf{M}$ is a stochastic matrix. Thus, the limiting value of the expectation of $P(.)$ converges to $\boldsymbol{\theta}$, and the rate of convergence of P to $\boldsymbol{\theta}$ is *fully determined* by λ.

Proof. Knowing that $p_1 + p_2 = 1$, and using this in Eq. (4.17), we can write

$$\mathrm{E}\left[p_1(n+1)|P\right] = (1-\lambda)\theta_1(p_1+p_2) + \lambda \mathrm{E}\left[p_1(n)\right] \tag{4.19}$$

$$\mathrm{E}\left[p_2(n+1)|P\right] = (1-\lambda)\theta_2(p_1+p_2) + \lambda \mathrm{E}\left[p_2(n)\right] \tag{4.20}$$

Simplifying the above equations by substitution and taking the expectations again, we get the following vectorial form for the expected value of $P(n+1)$:

$$\mathrm{E}\left[P(n+1)\right] = \mathbf{M}^T \mathrm{E}\left[P(n)\right] , \tag{4.21}$$

where the stochastic matrix, $\mathbf{M}$, is expressed as

$$\mathbf{M} = \begin{bmatrix} (1-\lambda)\theta_1 + \lambda & (1-\lambda)\theta_2 \\ (1-\lambda)\theta_1 & (1-\lambda)\theta_2 + \lambda \end{bmatrix} = (1-\lambda) \begin{bmatrix} \theta_1 & \theta_1 \\ \theta_2 & \theta_2 \end{bmatrix} + \lambda \mathbf{I} , \tag{4.22}$$

and $\mathbf{I}$ is the identity matrix. We see that Eq. (4.21) is analogous to the state occupation rule for a Markov chain, given previously in Eq. (4.2). Thus, $\mathrm{E}\left[P(n)\right]$ obeys an ergodic Markov chain whose *long run distribution* can be seen to converge to $\boldsymbol{\theta}$.

Since both $\mathrm{E}[P(n+1)]$ and $\mathrm{E}[P(n)]$ converge to $\mathrm{E}[P(\infty)]$, as $n \to \infty$, it follows that:

$$\mathrm{E}\left[P(\infty)\right] = \mathbf{M}^T \mathrm{E}\left[P(\infty)\right] . \tag{4.23}$$

To prove that

$$\mathrm{E}\left[P(\infty)\right] = \boldsymbol{\theta} , \tag{4.24}$$

we use the stochastic matrix from Eq. (4.22) and express $\mathrm{E}[p_1(\infty)]$ as

$$\begin{aligned} \mathrm{E}\left[p_1(\infty)\right] &= (1-\lambda)\theta_1 \left\{\mathrm{E}\left[p_1(\infty)\right] + \mathrm{E}\left[p_2(\infty)\right]\right\} + \lambda \mathrm{E}\left[p_1(\infty)\right] && (4.25)\\ &= (1-\lambda)\theta_1 + \lambda \mathrm{E}\left[p_1(\infty)\right] && (4.26)\\ \Rightarrow \mathrm{E}\left[p_1(\infty)\right](1-\lambda) &= \theta_1(1-\lambda)\,. && (4.27) \end{aligned}$$

This implies that $\mathrm{E}[p_1(\infty)] = \theta_1$. The same approach proves that $\mathrm{E}[p_2(\infty)] = \theta_2$, proving Eq. (4.24), which is the first part of the theorem. By observing that $(1-\lambda)$ is the common factor of $(\mathbf{M} - \lambda\mathbf{I})$, we see that the rate of convergence of P to $\boldsymbol{\theta}$ is *fully determined* by λ. The common factor is found in the simplified eigenvalue equation introduced previously in Eq. (4.5) and from the behavior of Markov chains, we know that the rate of convergence is determined by the eigenvalues of the stochastic matrix. More precisely, it is determined by the second largest eigenvalue, which in this case is λ. The theorem is thus proved. □

In the next theorem, we derive the asymptotic variance of the SLWE, with the knowledge we obtained in the previous analysis. It describes the rate of convergence in relation to the variance, and its dependence on λ. We will show that a small λ results in fast convergence and a large variance, and that a large value of λ will lead to slow convergence and a small variance. As we will see, the choice of this user-defined learning parameter, λ, boils down to a trade off between the speed and the corresponding accuracy.

Theorem 3. Let $\boldsymbol{x}$ be a binomially distributed random variable governed by the distribution $\boldsymbol{\theta}$, and $P(n)$ be the estimate of $\boldsymbol{\theta}$ at time n updated as per Eqs. (4.10) and (4.11). Then, the algebraic expression for the variance of $P(\infty)$ is fully determined by λ.

Proof. To derive the algebraic expression for the variance of $P(\infty)$, we will use the rule for calculating the variance introduced in Ch. 3. For this we need to derive $\mathrm{E}\left[p_1^2(\infty)\right]$. Using the notation from the previous theorem, and the updating rules given in Eqs. (4.10) and (4.11), we

express the square of p_1 at time $n+1$, as

$$\begin{aligned} p_1^2(n+1) &\leftarrow \lambda^2 p_1^2 && \text{w.p. } \theta_2 && (4.28)\\ &\leftarrow (1-\lambda p_2)^2 && \text{w.p. } \theta_1 && (4.29)\\ &= 1-2\lambda p_2+\lambda^2 p_2^2 && && (4.30)\\ &= 1-2\lambda(1-p_1)+\lambda^2(1-p_1)^2 && && (4.31)\\ &= 1-2\lambda+2\lambda p_1+\lambda^2(1-2p_1+p_1^2) && && (4.32)\\ &= 1-2\lambda+2\lambda p_1+\lambda^2-2\lambda^2 p_1+\lambda^2 p_1^2\,. && && (4.33) \end{aligned}$$

We can now write $\mathrm{E}\left[p_1^2(n+1)|P(n)\right]$ as follows, by using the updating rules given in Eqs. (4.28) and (4.33):

$$\begin{aligned} \mathrm{E}\left[p_1^2(n+1)|P(n)\right] &= \lambda^2 p_1^2\theta_2 \\ &\quad +(1-2\lambda+\lambda^2+2\lambda p_1-2\lambda^2 p_1+\lambda^2 p_1^2)\theta_1 && (4.34)\\ &= \lambda^2 p_1^2\theta_2+(1-2\lambda+\lambda^2)\theta_1 \\ &\quad +2\lambda(1-\lambda)p_1\theta_1+\lambda^2 p_1^2\theta_1 && (4.35)\\ &= \lambda^2 p_1^2+2\lambda(1-\lambda)p_1\theta_1+(1-\lambda)^2\theta_1\,. && (4.36) \end{aligned}$$

Again, this shows us that both $\mathrm{E}[p_1^2(n)]$ and $\mathrm{E}[p_1^2(n+1)]$ converge to $\mathrm{E}[p_1^2(\infty)]$ as $n\to\infty$. We can now simplify Eq. (4.36) by taking expectations a second time to get:

$$\begin{aligned} \mathrm{E}\left[p_1^2(\infty)\right] &= \lambda^2\mathrm{E}\left[p_1^2(\infty)\right] \\ &\quad +2\lambda(1-\lambda)\mathrm{E}\left[p_1(\infty)\right]\theta_1+(1-\lambda)^2\theta_1 && (4.37)\\ \mathrm{E}\left[p_1^2(\infty)\right](1-\lambda^2) &= 2\lambda(1-\lambda)\mathrm{E}\left[p_1(\infty)\right]\theta_1+(1-\lambda)^2\theta_1\,, && (4.38) \end{aligned}$$

which is the same as

$$\begin{aligned} \mathrm{E}\left[p_1^2(\infty)\right](1+\lambda)(1-\lambda) &= 2\lambda(1-\lambda)\mathrm{E}\left[p_1(\infty)\right]\theta_1+(1-\lambda)^2\theta_1 && (4.39)\\ \Rightarrow \mathrm{E}\left[p_1^2(\infty)\right](1+\lambda) &= 2\lambda\mathrm{E}\left[p_1(\infty)\right]\theta_1+(1-\lambda)\theta_1 && (4.40)\\ &= 2\lambda\theta_1^2+(1-\lambda)\theta_1\,. && (4.41) \end{aligned}$$

From Theorem 1, we know that $\mathrm{E}[p_1(\infty)]=\theta_1$, which allows us to write the last equality, given

in Eq. (4.41). Using this, we get

$$\mathrm{E}\left[p_1^2(\infty)\right] = \frac{2\lambda\theta_1^2 + (1-\lambda)\theta_1}{1+\lambda}\,. \tag{4.42}$$

Using the rule for calculating the variance, as given by Eq. (3.1) in Ch. 3, we express the variance of $p_1(\infty)$ as

$$\begin{aligned}
\mathrm{Var}[p_1(\infty)] &= \mathrm{E}[p_1^2(\infty)] - \mathrm{E}[p_1(\infty)]^2 && (4.43)\\
&= \frac{2\lambda\theta_1^2 + (1-\lambda)\theta_1}{1+\lambda} - \theta_1^2 && (4.44)\\
&= \frac{2\lambda\theta_1^2 + (1-\lambda)\theta_1 - (1+\lambda)\theta_1^2}{1+\lambda} && (4.45)\\
&= \frac{\lambda\theta_1^2 - \theta_1^2 + \theta_1 - \lambda\theta_1}{1+\lambda} && (4.46)\\
&= \frac{(\lambda-1)\theta_1^2 + (1-\lambda)\theta_1}{1+\lambda}\,. && (4.47)
\end{aligned}$$

The fact that $\theta_1 = 1 - \theta_2$, allows us to rewrite Eq. (4.47) as follows:

$$\begin{aligned}
\mathrm{Var}[p_1(\infty)] &= \frac{\theta_1((\lambda-1)\theta_1 + (1-\lambda))}{1+\lambda} && (4.48)\\
&= \frac{\theta_1((\lambda-1)(1-\theta_2) + (1-\lambda))}{1+\lambda} && (4.49)\\
&= \frac{\theta_1(\theta_2 - \lambda\theta_2)}{1+\lambda} && (4.50)\\
&= \frac{(1-\lambda)\theta_1\theta_2}{1+\lambda}\,. && (4.51)
\end{aligned}$$

As we now see, the algebraic expression for the variance of $P(\infty)$ is fully determined by λ, and the theorem is proved. □

The results we have derived here are asymptotic, i.e., they are only valid as $n \to \infty$. We observe from Eq. (4.51) that as $\lambda \to 1$, the variance of our estimate tends to zero. This implies mean square convergence, as defined earlier in Ch. 3. The maximum value of the estimator variance is attained when the learning parameter, λ is 0. On the contrary, its minimum value is obtained when $\lambda = 1$. The asymptotic nature of the SLWE may seem contradictory with regard to our initial goal presented in the introduction of this chapter, i.e., that we are working with a non-stationary environment. Realistically though, the convergence of the SLW estimate takes

place after a relatively small value of n. Our empirical results, that will be presented in the next chapter, demonstrates this. We have also empirically obtained good values of λ, which we found to lie in the interval $[0.9, 0.99]$. This is also in accordance with "suitable" values presented in the literature on stochastic learning [43]. Even for a "small" value of λ, such as 0.9, the results shows that sufficient convergence is achieved after only 50 iterations. By sufficient, we here obtain a variation from the asymptotic value of the order of 10^{-50}. Narendra and Thathachar discuss the asymptotic behavior of reinforcement learning schemes, such as the L_{R-I} scheme in [43], which is similar to the updating rules we use here. They demonstrate how the learning parameter, λ, determines the rate of convergence and that this occurs in a geometric manner. These results apply to the SLWE as well.

We can conclude that even if the environment switches its Bernoulli parameter after only 50 time steps, the SLWE will be able to track this change. We also see that there is no need to make use of a "sliding window", as discussed earlier. Our simulation results, which will be presented in the following chapter, demonstrate these properties of the SLWE, and compare it to the "sliding window" version of the ML estimates. We will now look at a generalization of the SLWE for multinomial distributions.

4.4 Weak Estimators of Multinomial Distributions

In this section, we will investigate the problem of estimating the parameters generated by a multinomial distribution. Such a distribution is characterized by two parameters, namely; the number of trials for our experiment, and, a probability vector which tells us the probability of a specific event from occurring. As opposed to the binomial case, where the problem was to estimate a single parameter, we here deal with a probability *vector* instead. This probability vector consist of a predefined number of possible events. Again, we assume that the number of observations is the number of trials. The problem at hand is thus to estimate the probability vector associated with the set of r possible outcomes. Instead of dealing with a random variable taking on two possible values, like we did in the binomial case, we now deal with a multinomially distributed random variable, $\boldsymbol{x}$, which takes on the values from the set $\{1, \ldots, r\}$. Again, we assume that $\boldsymbol{x}$ is governed by the distribution $\boldsymbol{\theta}$, which in the general case takes on the form $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_r]^T$. Here, $\boldsymbol{x} = i$ w.p. θ_i, and the probabilities sum to unity, $\sum_{i=1}^{r} \theta_i = 1$.

We let $x(n)$ represent the concrete realization of $\boldsymbol{x}$ at time n. The task at hand is to estimate $\boldsymbol{\theta}$, or θ_i for $i = 1, \ldots, r$. As in the binomial case, we maintain a running estimate $P(n) =$

$[p_1(n), \ldots, p_r(n)]^T$ of $\boldsymbol{\theta}$. This vector, $P(n)$ is analogous to the estimate in the binomial case, where $p_i(n)$ is the estimate of θ_i at time n, for $i = 1, \ldots, r$. Again, we omit the time reference, n, in $P(n)$, whenever this does not lead to confusion. Generalizing the updating scheme used in the binomial case, we get the following, simple, updating scheme for the multinomial case:

$$p_i(n+1) \leftarrow p_i + (1-\lambda)\sum_{j \neq i} p_j \quad \text{when } x(n) = i \tag{4.52}$$

$$\leftarrow \lambda p_i \quad \text{when } x(n) \neq i. \tag{4.53}$$

The rule is similar for other values of $p_j(n)$, where $j \neq i$. The updating rule used above is analogous to that of the updating scheme used in multi-action L_{RI} learning automata [43].

In the following theorem we will show the dependence of $\mathrm{E}[P(n+1)]$ on $\mathrm{E}[P(n)]$, as we did for the binomial distribution, and how this dependence is related to a stochastic matrix. We will also see how this matrix possess the same eigenvector and eigenvalue properties as the ergodic Markov matrix, introduced previously in Section 4.2. As in the binomial case, we will show that this leads to two results. The first involves the limiting solution of the expected solution vector of the Markov chain, and the second involves the rate of convergence of the chain. We will see that in a worst case scenario, both of these will only be dependent on the learning parameter, λ. We will see that the rate of convergence is only determined by a linear function of the learning parameter, λ, which is the same function as the eigenvector function introduced in Eq. (4.4). These results are analogous to the results shown for the binomial case in Section 4.3.

Theorem 4. Let $\boldsymbol{x}$ be a multinomially distributed random variable governed by the distribution $\boldsymbol{\theta}$, and let $P(n)$ be the estimate of $\boldsymbol{\theta}$ at time n, obtained by the updating rules given in Eqs. (4.52) and (4.53). Then, $\mathrm{E}\left[P(\infty)\right] = \boldsymbol{\theta}$.

Proof. To prove the theorem, we can first rewrite the updating rules given in Eqs. (4.52) and (4.53), as follows:

$$p_i(n+1) \leftarrow p_i + (1-\lambda)(1-p_i) \quad \text{w.p. } \theta_i \tag{4.54}$$

$$\leftarrow \lambda p_i \quad \text{w.p. } \sum_{j \neq i} \theta_j\,, \tag{4.55}$$

since we know that the probabilities sum to unity. This lets us express the expected value of

$p_i(n+1)$ based on the running estimate, P, at time n, as

$$\begin{aligned} \mathrm{E}\left[p_i(n+1)|P(n)\right] &= p_i\theta_i + (1-\lambda-p_i+\lambda p_i)\theta_i + \lambda p_i(1-\theta_i) && (4.56)\\ &= p_i\theta_i + \theta_i - \lambda\theta_i - p_i\theta_i + \lambda p_i\theta_i + \lambda p_i - \lambda p_i\theta_i && (4.57)\\ &= (1-\lambda)\theta_i + \lambda p_i\,. && (4.58) \end{aligned}$$

As we did in the binomial case, we take the expectations again, giving us the following

$$\mathrm{E}[p_i(n+1)] = (1-\lambda)\theta_i + \lambda\mathrm{E}[p_i(n)]\,. \tag{4.59}$$

Now, as $n \to \infty$, we see that both the expected value of p_i at time n and $n+1$, converges to $\mathrm{E}[p_i(\infty)]$. In the binomial case, we observed that the expected value of p_i converged to a limit because the common multiplying factor of the linear difference equation was λ, which we know is both positive and strictly less than 1. The same observation made for the binomial case is valid for the multinomial case as well. We can now write

$$\begin{aligned} \mathrm{E}[p_i(\infty)](1-\lambda) &= (1-\lambda)\theta_i && (4.60)\\ \Rightarrow \mathrm{E}[p_i(\infty)] &= \theta_i\,. && (4.61) \end{aligned}$$

Since the equation above is valid for all components p_i of P, the theorem is proved. □

With the above theorem proved, we can now derive the explicit dependence of $\mathrm{E}[P(n+1)]$ on $\mathrm{E}[P(n)]$ and the consequences.

Theorem 5. Let the parameter $\boldsymbol{\theta}$ of the multinomial distribution be estimated at time n by $P(n)$, obtained by (4.52) and (4.53). Then, the expectation of $P(n+1)$ is related to the expectation of $P(n)$ as $\mathrm{E}\left[P(n+1)\right] = \mathbf{M}^T\mathrm{E}\left[P(n)\right]$, in which every off-diagonal term of the stochastic matrix, $\mathbf{M}$, has the *same* multiplicative factor, $(1-\lambda)$. Furthermore, the final solution of this *vector* difference equation is independent of λ.

Proof. The expectation of $p_1(n+1)$, given P at time n, can be expressed from Eq. (4.58) in the following fashion:

$$\mathrm{E}\left[p_1(n+1)|P\right] = (1-\lambda)\theta_1\sum_{j=1}^{r} p_j + \lambda p_1\,. \tag{4.62}$$

This can be generalized in the same fashion for all other conditional probabilities simply by replacing the explicit reference to p_1 by p_i, for all $i = 1,\ldots,r$, as expressed in the following

equation:

$$\mathrm{E}\left[p_i(n+1)|P\right] = (1-\lambda)\theta_i \sum_{j=1}^{r} p_j + \lambda p_i \,. \tag{4.63}$$

If we organize all the terms in Eq. (4.63) vectorially and express them by means of a stochastic matrix, $\mathbf{M}$, we can show that $\mathrm{E}[P(n+1)]$ can be expressed as a dependency on the expected value of $P(n)$ as $\mathrm{E}[P(n+1)] = \mathbf{M}^T\mathrm{E}[P(n)]$. The stochastic matrix, $\mathbf{M}$, is then as given in Eq. (4.64) below.

$$\mathbf{M} = \begin{bmatrix} (1-\lambda)\theta_1 + \lambda & (1-\lambda)\theta_2 & \cdots & (1-\lambda)\theta_r \\ (1-\lambda)\theta_1 & (1-\lambda)\theta_2 + \lambda & \cdots & (1-\lambda)\theta_r \\ \vdots & \vdots & \ddots & \vdots \\ (1-\lambda)\theta_1 & (1-\lambda)\theta_2 & \cdots & (1-\lambda)\theta_r + \lambda \end{bmatrix} . \tag{4.64}$$

To obtain the solution for $\mathrm{E}[P(n)]$ as $n \to \infty$, we solve the following vectorial difference equation:

$$\mathrm{E}\left[P(\infty)\right] = \mathbf{M}^T\mathrm{E}\left[P(\infty)\right] \,. \tag{4.65}$$

We know that this solution exists because of the nature of the stochastic matrix $\mathbf{M}$, which has one eigenvalue which is unity, and all the other eigenvalues are strictly less than unity.

The theorem states that every off-diagonal term of the stochastic matrix, $\mathbf{M}$, has the *same* multiplicative factor, $(1-\lambda)$, which we clearly see in Eq. (4.64). This also counts for the diagonal of the matrix, but, we observe that the diagonal also contains the addition of λ. Using this, we can express Eq. (4.64) in the following fashion:

$$\mathbf{M} = (1-\lambda)\left[\boldsymbol{\theta}|\boldsymbol{\theta}|\cdots|\boldsymbol{\theta}\right]^T + \lambda\mathbf{I} \,, \tag{4.66}$$

where $\boldsymbol{I}$ is the identity matrix.

Using this new notation for $\mathbf{M}$ and inserting this in the difference equation introduced previously in Eq. (4.65), we get the following:

$$\mathrm{E}\left[P(\infty)\right] = (1-\lambda)\left[\boldsymbol{\theta}|\boldsymbol{\theta}|\cdots|\boldsymbol{\theta}\right]^T\mathrm{E}\left[P(\infty)\right] + \lambda\mathbf{I}\mathrm{E}\left[P(\infty)\right] \,. \tag{4.67}$$

Through simplification, this leads to the following result:

$$\mathrm{E}\left[P(\infty)\right] = \boldsymbol{\theta} \,, \tag{4.68}$$

which can be verified for all $i = 1, \ldots, r$, as follows:

$$\begin{aligned} \mathrm{E}\left[p_i(\infty)\right] &= (1-\lambda)\theta_i \sum_{j=1}^{r} \mathrm{E}\left[p_j(\infty)\right] + \lambda \mathrm{E}\left[p_i(\infty)\right] && (4.69)\\ &= (1-\lambda)\theta_i + \lambda \mathrm{E}\left[p_i(\infty)\right] && (4.70)\\ \Rightarrow \mathrm{E}\left[p_i(\infty)\right](1-\lambda) &= (1-\lambda)\theta_i\,, && (4.71) \end{aligned}$$

implying that $\mathrm{E}[p_i(\infty)] = \theta_i$, since we know that $\sum_{j=1}^{r} \mathrm{E}\left[p_j(\infty)\right] = 1$. The result follows. □

From the analysis given above we will now investigate the convergence and eigenvalue properties of the stochastic matrix, $\boldsymbol{M}$, and its dependence on the learning parameter, λ.

Theorem 6. Let the parameter $\boldsymbol{\theta}$ of the multinomial distribution be estimated at time n by $P(n)$ obtained by Eqs. (4.52) and (4.53). Then, all the non-unity eigenvalues of M are exactly λ, and thus the rate of convergence of P is *fully* determined by λ.

Proof. To analyze the rate of convergence of the SLWE, we first find the eigenvalues of the stochastic matrix, $\boldsymbol{M}$, namely $\xi_1, \xi_2, \ldots, \xi_r$. We use the vector difference equation posed in Eq. (4.65) as a starting point for our analysis. According to the rule of eigendecomposition of a matrix, we are able to express $\boldsymbol{M}$ as follows [14]:

$$\mathrm{M} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1}\,, \tag{4.72}$$

where $\boldsymbol{\Phi}$ is the square $n \times n$ matrix containing all the eigenvectors of $\boldsymbol{M}$, as described in the following equation:

$$\boldsymbol{\Phi} = \begin{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \middle| \begin{bmatrix} -\frac{\theta_2}{\theta_1} \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \middle| \begin{bmatrix} -\frac{\theta_3}{\theta_1} \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \middle| \cdots \middle| \begin{bmatrix} -\frac{\theta_r}{\theta_1} \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \end{bmatrix}, \tag{4.73}$$

The eigendecomposition is useful for understanding the solution of a system of linear ordinary differential equations or linear difference equations. Since $\boldsymbol{\Lambda}$ is diagonal, raising it to power t to yield $\boldsymbol{\Lambda}^t$, just involves raising each element on the diagonal to the power t.

From Eq. (4.72), we observe that $\mathbf{\Lambda}$ consists of all the eigenvalues of $\boldsymbol{M}$, as outlined in the following equation:

$$\mathbf{\Lambda} = \text{diag}(1, \lambda, \lambda, \ldots, \lambda) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix}. \tag{4.74}$$

It turns out that due to the unique properties of $\mathbf{\Phi}$, the following is true:

$$\mathbf{\Phi}^{-1} - \mathbf{I} = \begin{bmatrix} \theta_1 - 1 & \theta_2 & \cdots & \theta_r \\ -\theta_1 & -\theta_2 & \cdots & -\theta_r \\ \vdots & \vdots & \ddots & \vdots \\ -\theta_1 & -\theta_2 & \cdots & -\theta_r \end{bmatrix}. \tag{4.75}$$

Note that in Section 4.2, we stated that every Markov chain has one eigenvalue of magnitude unity. The remaining eigenvalues come either in complex conjugate pairs or as real values. By ordering the eigenvalues, the following is true: $\xi_1 = 1 > \xi_2 \geq \xi_3 \geq \ldots \geq \xi_r$. For our stochastic matrix, $\boldsymbol{M}$, we have one eigenvalue with value $\xi_1 = 1$, and all other eigenvalues are exactly equal to the learning parameter, λ, i.e., $\xi_i = \lambda$ for $i = 2, \ldots, r$, as seen in the previous equations.

The solution of the eigenvalue problem posed in Eq. (4.72), can be easily verified for $i = 1$ as follows:

$$\mathbf{M} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} (1-\lambda)(\theta_1 + \ldots + \theta_r) + \lambda \\ \vdots \\ (1-\lambda)(\theta_1 + \ldots + \theta_r) + \lambda \end{bmatrix} = 1 \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{4.76}$$

The same assertions are proved for $i = 2, \ldots, r$ in the following manner:

$$\mathbf{M}\begin{bmatrix} -\frac{\theta_i}{\theta_1} \\ 0 \\ \vdots \\ 0 \\ 1_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{[(1-\lambda)\theta_1+\lambda]\theta_i}{\theta_1} + (1-\lambda)\theta_i \\ 0 \\ \vdots \\ 0 \\ -\frac{(1-\lambda)\theta_1\theta_i}{\theta_1} + (1-\lambda)\theta_i + \lambda \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \lambda \begin{bmatrix} -\frac{\theta_i}{\theta_1} \\ 0 \\ \vdots \\ 0 \\ 1_i \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{4.77}$$

where 1_i represents the presence of unity at position i.

As demonstrated for the Markov matrix in Section (4.2), the rate of convergence of the stochastic matrix is determined by the second largest eigenvalue, which, for the SLWE, is λ. The first eigenvalue is equal to unity and the learning parameter λ is an eigenvalue of multiplicity $r - 1$. This proves the theorem. □

For the multinomial distribution, the SLWE shows the same results as we previously demonstrated for the binomial case. As $\lambda \to 1$, the variance of our estimate tends to zero, demonstrating that the estimate converge in the mean-square sense. This also implies that the SLWE is at least asymptotically unbiased. Again, we see that the maximum value of the variance is achieved when $\lambda = 0$, and the minimum value is attained as $\lambda = 1$. From Ch. 3, we know that mean-square convergence also implies that the SLWE exhibits consistency and efficiency. The results shown here are asymptotic, and thus only valid as $n \to \infty$, but, as stated for the binomial case, realistically, the convergence takes place after a relatively small number of time steps. Results analogous to the binomial case will be demonstrated in the next chapter, showing that for a relatively "small" value of λ, the SLWE will converge with a variation from the asymptotic value of the order of 10^{-50}, after only 50 time steps. This demonstrates that even if the multinomial probability vector of the environment is switched after only 50 time steps, the SLWE is able to track this change.

Narendra and Thathachar discuss convergence in asymptotic systems with reinforcement learning schemes as the basis for their results in [43]. The schemes discussed are analogous to the updating scheme used by the SLWE, and the same results with regard to rate of convergence are equally valid for both cases. A small value of λ leads to fast convergence, but with a large variance. On the other hand, a large value for the learning parameter will lead to slow convergence

but with higher accuracy, i.e., with a small variance. This shows us that the user-defined learning parameter presents us with a trade-off issue concerning speed and accuracy.

With regard to computational complexity we observe that the SLWE requires time that is linear on the sample size, n, i.e., the worst-case time complexity for the SLWE to estimate parameters in the binomial case is $O(n)$. For the multinomial case, the time complexity is dependent on the size r of the parameter vector, $\boldsymbol{\theta}$, giving us a worst-case time complexity of $O(rn)$.

4.5 Using Other LA Schemes

The updating schemed used for the SLWE presented in this chapter has been based on the principles of stochastic learning, similar to the families of Variable Structure Stochastic Automata (VSSA) [30, 41, 58]. As explained previously the updating rules used are closely related to linear scheme, i.e., the L_{R-I} scheme. As a main motivator for using these kind of schemes, Oommen *et. al.* state that the analysis is considerably simplified and fairly elegant. They also mention that a lot of work has been done in the field of learning automata, and suggest variations of the SLWE that utilize the principles from the so-called families of Discretized LA [58], and the families of Pursuit and Estimator algorithms [59, 60]. These approaches are still left open. Another approach would be to adopt the L_{R-P} updating scheme in a similar fashion as we have done for the L_{R-I} scheme. This approach is also open. Oommen *et. al.* conclude that these approaches might be promising, but that the analysis of their properties will not be trivial.

In the next chapter we will demonstrate how the SLWE performs in comparison to the traditional window-based MLE approach in both binomial and multinomially non-stationary environments.

Chapter 5

Simulation on Synthetic Data

In this chapter we present simulation results comparing the Stochastic Learning Weak Estimator (SLWE) to the window-based variant of the maximum likelihood estimate (MLEW). We demonstrate their performance in both binomial and multinomial environments. All simulations were performed on *synthetic data* with both manually chosen, and randomly generated parameters. The simulations were performed to support the theoretical proofs presented in the previous chapter, concerning the efficiency and suitability of the SLWE in non-stationary environments.

To perform these experiments, we developed a simulation test bench in Java [1]. The test bench is able to perform simulations in both binomial and multinomial environments.

In Section 5.1, we discuss the performance measurements used for our experiments, and Section 5.2 discuss the choice of parameters for our simulations. Some of our results from the simulations with binomial and multinomial distributions are outlined in Sections 5.3 and 5.4, along with a brief discussion of our observations. We summarize these results and our observations in Section 5.5.

5.1 Performance Measurements

To compare our estimation algorithms, we need a performance measure. We recall from Ch. 3, that the estimation error is given by the difference between the estimate and the actual parameter value, as explained in Eq. (3.2). Furthermore, the Mean Squared Error (MSE) is defined as the expected

[1] The test bench can be made available upon request.

value of the square of the estimation error, as stated in Eq. (3.3). For our estimation methods we deal with both binomial and multinomial distributions in non-stationary environments, where the objective is to estimate a parameter value in the binomial case, and a parameter vector in the multinomial case.

5.1.1 Euclidean Distance

To measure the distance between two vectors, it is common to calculate what is called the *Euclidean distance*. This distance is defined as follows:

$$||S(n) - \boldsymbol{\theta}(n)|| = \sqrt{\sum_{i=1}^{r}(s_i - \theta_i)^2}\,, \tag{5.1}$$

where $S(n)$ is the estimate vector at time n, $\boldsymbol{\theta}(n)$ is the true value vector at time n, and r is the number of elements in the vectors, i.e., the dimension of the vectors.

For the binomial case it is not necessary to take both elements of the estimate vector into account, since we know that $s_2 = 1 - s_1$. Therefore, for the binomial case, we use a simplified distance measure, explained in Eq. (5.2).

$$||s_1(n) - \theta_1(n)|| = \sqrt{(s_1 - \theta_1)^2}\,. \tag{5.2}$$

5.1.2 Estimator Performance Measure

To establish the estimator performance measure, we make use of the Euclidean distance, and calculate the expected value of this for the simulation. As stated earlier in Ch. 3, the Mean Squared Error of the estimate is the expected value of the square of the estimation error. This measure is commonly used to evaluate the performance of an estimator, and is therefore an obvious choice for our simulations as well. We observe that the square of the estimation error is in fact the Euclidean distance described in the previous section, and that the MSE can easily be obtain by calculating the average of the euclidean distance for each simulation. This is often referred to as the *time average*. To summarize, we calculate the Euclidean distance at every time step for each simulation, then sum these individual distances together and divide by the number of time steps in the simulation, N.

It is well known that the more samples we take, the more precise time average we get. In our simulations, we introduce what is known as the *ensemble average*. The ensemble average is calculated by running an ensemble of experiments, then for each experiment measure the averaged euclidean distance at each time step. For each simulation in the ensemble, we sum the distance at time step n and finally divide this by the number of experiments in the ensemble, E. The ensemble average of the Euclidean distance is what we use as our performance measure for the simulations presented later in this chapter.

5.2 Parameter Choice And Consequences

In the previous chapter we proved that the rate of convergence of the SLWE is dependent on the learning parameter, λ. In this chapter we shall demonstrate that the rate of convergence of the MLEW is dependent on its window width parameter. Finding suitable values for these parameters is therefore important. In our experiments we empirically found such "good" values for both the learning parameter, λ, and for the window width, w. To assess the robustness of each estimation algorithm, we randomly select the parameters for each experiment from an interval of values that are regarded as suitable in the literature.

In Figure 5.1 we show a plot of the rate of convergence of the SLWE and the MLEW in a stationary environment, where the true underlying parameter $\theta_1 = 0.415955$. Here, we observe that with a high value for the learning parameter $\lambda = 0.995$, the SLWE is able to converge with a very low variance, to the true value. However, it takes almost 300 time steps before convergence is achieved. The SLWE converges to a value of 0.411558. The MLE estimate[2] converges relatively quickly, but we observe that the variance of this estimate is higher and the value that it converges to is 0.396168. The true value is approached after about 100 time steps. The MLEW with $w = 30$ indicates high variance, but is able to approach the true underlying value after about 40 - 50 time steps. Similar results are observed for the SLWE when we settle for a lower value for the learning parameter, such as $\lambda = 0.95$.

Higher values for λ performs in accordance to what is suggested in the literature on stochastic learning automata [30, 31, 41, 58], but we observe that because of the asymptotic nature of the estimates, these values may result in too slow convergence when we are dealing with non-stationary environments. This clearly presents us with an important trade-off question, often referred to as

[2]The ML estimate here is the regular, non-windowed MLE.

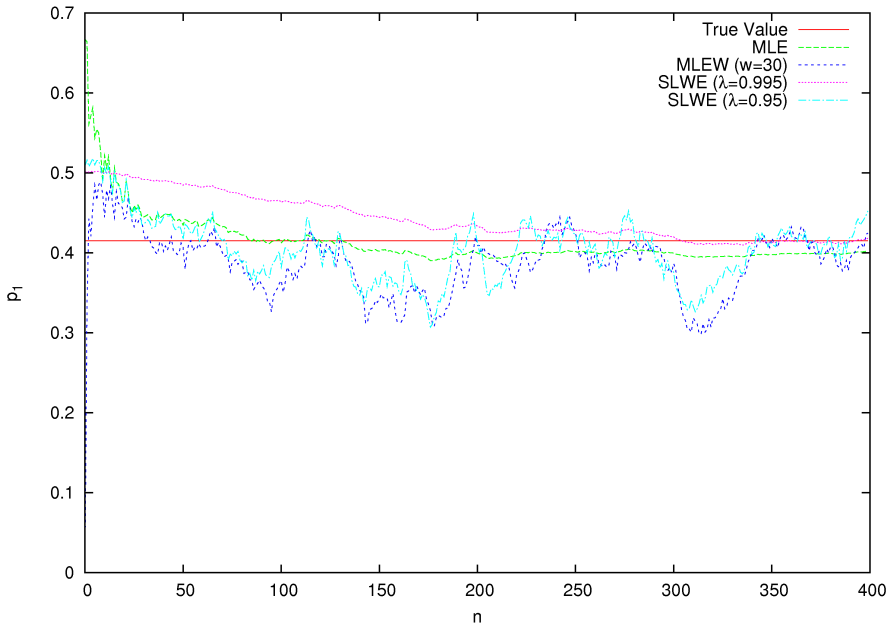

Figure 5.1: Plot of the MLE, MLEW and SLWE operating in a stationary environment where $\theta_1 = 0.415955$.

the *time and accuracy conflict*.

For the binomial case, Oommen *et. al.* use values obtained randomly from a uniform distribution in $[0.55, 0.95]$ and $[20, 80]$ for λ and w respectively [46]. In the multinomial case, they choose λ close to 1, which is in accordance with the suggested values for multi-action LA schemes [43]. Oommen and his co-authors operate with values in $[0.9, 0.99]$, and states that

> *"It is well-known in LA schemes that choosing a value of λ close to 1 (for the binomial case, we have empirically found that $\lambda \approx 0.9$) gives quite good results."*

The choice of such suitable parameter values will be explained in our simulation results, were we will look at some test cases involving "good" versus "bad" values for both λ and w.

5.3 Simulations With Binomial Distributions

In this section we investigate the performance of the SLWE and the MLEW in a binomial environment. The simulations have been performed with our test-bench, as described previously in this chapter. We experiment with three different test cases. The first one involves finding empirically

suitable values for the estimation algorithms to achieve satisfying rate of convergence and accuracy. The second test case involves using randomly drawn variables from given intervals to assess the efficiency and robustness of the two estimation algorithms. The last test case for the binomial case involves experiments with different frequencies of change in the environment to assess how well the estimation algorithms adapts to such changes.

5.3.1 Simulation Setup

For each simulation, the estimators were given a sequence of bits generated randomly by the environment, based on its Bernoulli parameter, $\boldsymbol{\theta}$. This parameter was drawn from a uniformly distributed random variable in $[0, 1]$. As described in the previous chapter, for the binomial case, $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$, where θ_1 is the probability of the random variable $\boldsymbol{x}$ taking on the value 1 and $\theta_2 = 1 - \theta_1$ is the probability of $\boldsymbol{x} = 2$. The task of the estimation algorithms is to estimate this underlying parameter, based on the given sequence of bits.

Since we wish to demonstrate non-stationary distributions, the environment switches its Bernoulli parameter regularly. In our simulations, we deal with two different ways of switching this parameter; either by changing it after a given number of time steps, or randomly at every time-step, with a certain probability.

For our experiments we generate a sequence of N samples, representing N time steps. As described previously, we run an ensemble of experiments, where E represents the number of simulations in the ensemble. We then report the respective ensemble average at every time instant. By performing a high number of simulations in the ensemble, the variance of the estimates will be smoothed, yielding more readable plots. For each test case, we plot the estimated probability of 1 which is denoted p_1. We also report the averaged performance measure for each estimation algorithm over the ensemble. As outlined in the previous chapter, the SLWE keeps a running estimate, which, for our binomial simulations, was initialized to $P(0) = [0.5, 0.5]^T$.

5.3.2 Results

As mentioned previously, we have three different test cases for the binomial simulations, each with several different sub-cases. The objective of the first case is to empirically decide on suitable parameters that yields fast enough convergence so that the estimators are able to track the changes in the environment in addition to yielding high accuracy. We let the environment switch

its Bernoulli parameter regularly at given intervals to simulate a non-stationary environment. In the second test case, we demonstrate how well each estimation algorithm performs with randomly drawn parameters from a predefined set of suitable values, obtained from the first test case. Here, we experiment with both regular and random change points to assess the robustness of the estimation algorithms with regard to changes in the environment. The last test case involves simulations where we try different frequencies of change in the environment to assess both the efficiency and the robustness of the estimation algorithms when no a priori information about the changes in the environment is known to the algorithms. We include this last test case to further demonstrate how the estimation algorithms are able to adapt to changes in the environment with regard to both magnitude and frequency of change. In real-life applications such as PR systems, the classifier does not have any knowledge regarding frequency or magnitude of change in the environment, and it is therefore interesting to investigate how well the the estimation algorithms are able to deal with these problems.

Test Case I: Finding Suitable Parameter Value

In this test case we performed experiments with 500 time steps on an ensemble of 2000 simulations. The aim of this first test case was to demonstrate the impact of high versus low values for both the learning parameter of the SLWE and for the window width of the MLEW. The environment switched its Bernoulli parameter every 50 time steps.

We ran five experiments using the same environment (e.g. the true underlying parameter was the same for these simulations.) The test was performed with several different values of λ for the SLWE and w for the MLEW. Each individual ensemble is outlined in Table 5.1 and the corresponding averaged estimation errors are accordingly reported.

			Averaged Estimation Error	
Experiment	λ	w	SLWE	MLEW
1	0.85	40	0.1158	0.1534
2	0.85	20	0.1158	0.1233
3	0.85	10	0.1158	0.1302
4	0.8	20	0.1223	0.1233
5	0.9	20	0.1161	0.1233

Table 5.1: Test Case I - Experiments

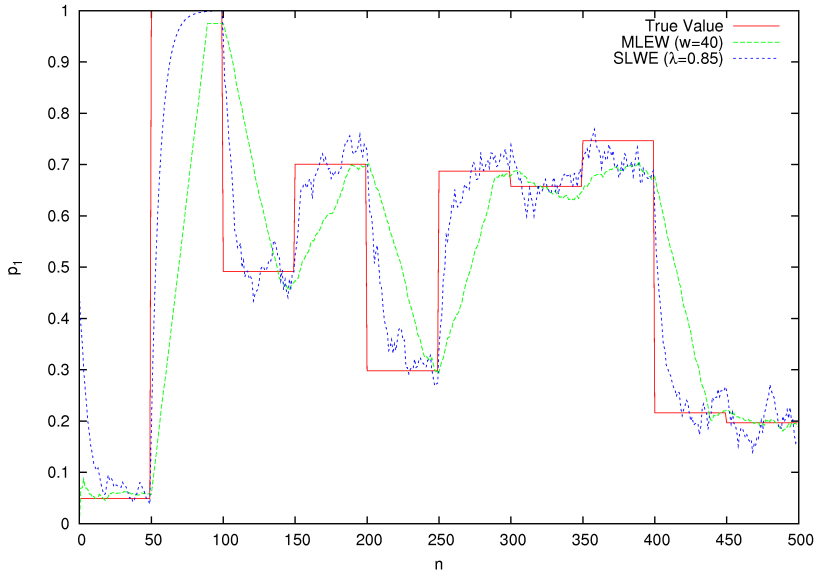

Figure 5.2: Plot of the averages of the estimates of $p_1(n)$, at time n, obtained from the SLWE and the MLEW using $\lambda = 0.85$ and $w = 40$ for the first experiment in the first binomial test case. The environment switched its parameter randomly, every 50 time steps.

Plot of the estimated probability p_1, for each estimation algorithm, for the five experiments are shown in Figs. 5.2, 5.3, 5.4, 5.5 and 5.6.

Fig. 5.2 indicates how well the SLWE is able to track changes in the environment, and estimate the true value with low variance. When using a window size of 40, the MLEW is slow and the convergence is less accurate than the SLWE. We also observe this in the reported performance measures in Table 5.1, where the averaged estimation error of the SLWE is 0.1158 and 0.1534 for the MLEW.

Figs. 5.3 and 5.4 shows the plot of the same SLWE estimate, but with window sizes for the MLEW set to 20 and 10, respectively. Here we observe that by lowering the window width, the MLEW is able to achieve faster convergence, but with lower overall accuracy in the sense of how close the estimate approximates the true value of the environment. Despite this, faster convergence does lower the averaged estimation error of the ensemble. Yet, the MLEW is not able to achieve as high accuracy as the SLWE does. The best performance measure is achieved with the window size being set to 20 resulting in an averaged estimation error of 0.1233, compared to 0.1158 which was the reported averaged estimation error of the SLWE in the same experiment.

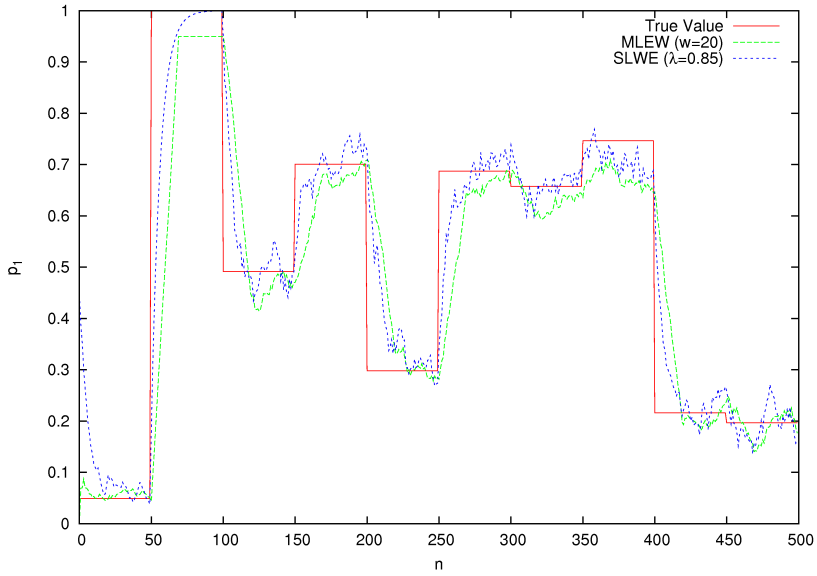

Figure 5.3: Plot of the expected value of the estimates of $p_1(n)$ for experiment 2 in the first binomial test case, obtained from the SLWE and MLEW, where $\lambda = 0.85$ and $w = 20$.

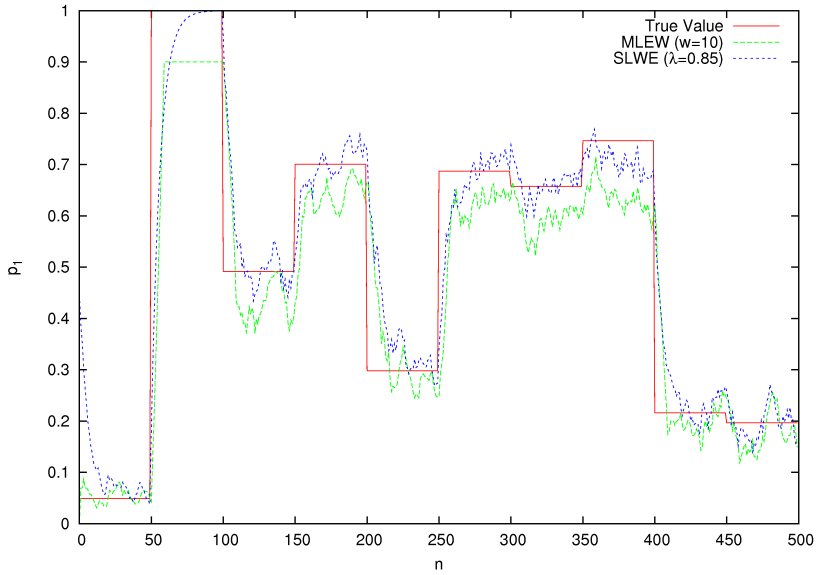

Figure 5.4: Plot of experiment 3 for test case one in the binomial simulations, which was estimated by using the SLWE and the MLEW, where $\lambda = 0.85$ and $w = 10$.

In experiment 4 we wanted to investigate how the SLWE was able track the changes when we lowered the learning parameter of the SLWE to $\lambda = 0.8$ and kept a window size of $w = 20$. These results are illustrated in Fig. 5.5. This demonstrates the power of the SLWE and that it is able to

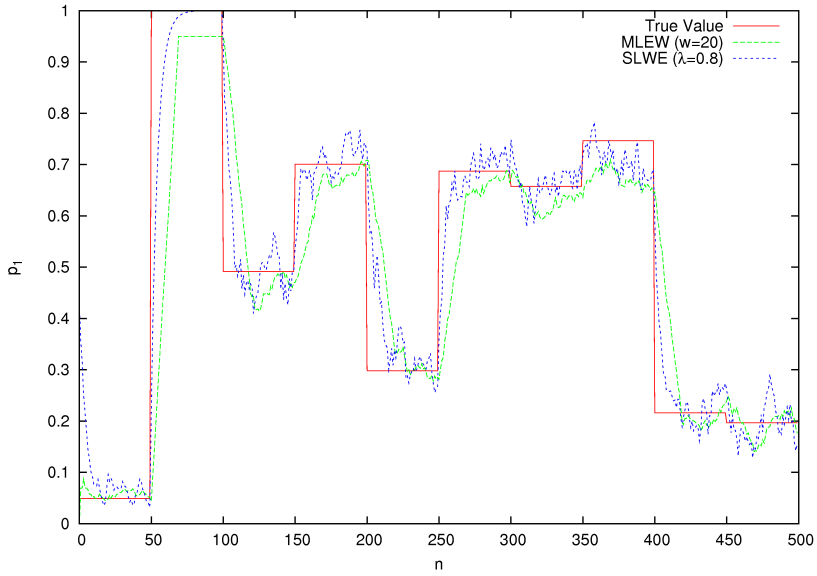

Figure 5.5: The expected value of $p_1(n)$, at time n, for experiment 4 in test case one, which was estimated by using the SLWE and the MLEW, where $\lambda = 0.8$ and $w = 20$.

track the change quickly and still perform with a slightly higher accuracy than the MLEW with a window size as low as 20. The reported estimation error of the SLWE in this experiment was 0.1223, compared to the estimation error of the MLEW which was 0.1233.

In the last experiment we increased the learning parameter to $\lambda = 0.9$ to investigate how the SLWE was able to operate using a larger value for the learning parameter. We know that higher values yield slower, but more accurate, convergence results and it was thus of interest to examine how this affects the overall performance of the estimation algorithm, The reported estimation error for the SLWE in this case was 0.1161 and 0.1233 for the MLEW with the window parameter set to $w = 20$.

From these results we see that the choice of parameters are vital with regard to the time and accuracy trade off question. The results also gives us a pointer on what values of the parameters that yields good results. From our testing we experimentally found values in the interval $[0.8, 0.95]$, for the learning parameter, and $[10, 50]$, for the MLEW, to yield satisfying results. The reason for operating with larger window sizes than 50 is that we generally have no knowledge about the environment with regard to the frequency or magnitude of the changes. We shall further investigate this in the next two test cases.

We know that a large value of λ yields low variance, i.e., the accuracy is high. The problem is

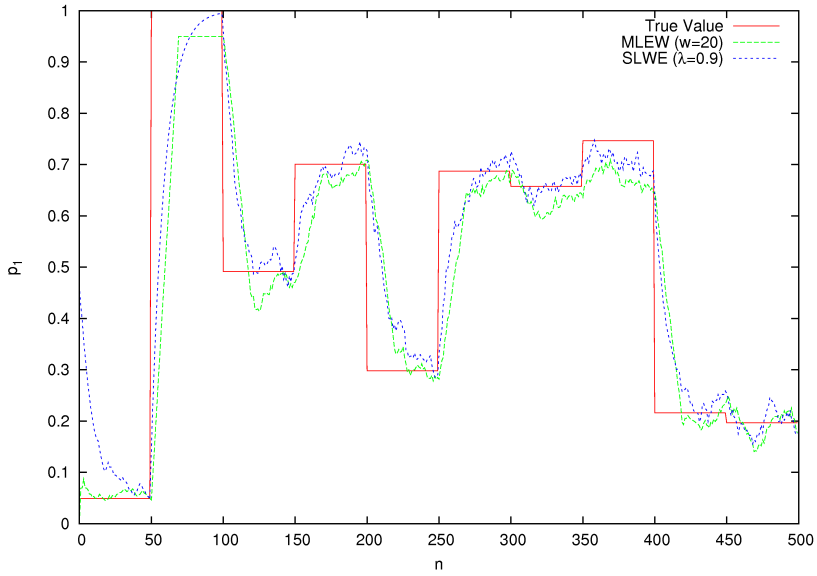

Figure 5.6: Plot for experiment 5 in the binomial test case one, where the averages of the estimates of $p_1(n)$, at time n was obtained from the SLWE and the MLEW. In this case, $\lambda = 0.9$ and the window size is 20.

that the rate of convergence is slower than when we are using a lower value of λ. A large value of λ gives weight to the history of the samples, and a small value of λ gives large weight to the value of the next sample. This results in faster convergence, but yields higher variance from the true underlying parameter. Similar results are observed for the window width parameter of the MLEW as well. A small value for the window width, w, yields less accuracy, but faster convergence. This implies fewer data points within each window and therefore larger estimate variances. A large window size yields accurate, but slower estimates.

It is important to point out that the rate of convergence also affects the overall performance of the estimation algorithms. We have observed that slow convergence with low variance yields lower overall results than faster convergence with higher variance. We have also seen that too fast convergence may result in greater variance from the true value, and thus yield lower overall performance. This all boils down to the question of finding appropriate parameters that yields fast convergence with a sufficiently low variance.

Test Case II: Random Values

To assess the efficiency of the estimation methods in a fair way, we randomly chose values for the learning parameter and for the window width from the intervals $[0.8, 0.95]$ and $[10, 80]$, respectively. Note that Oommen and his co-authors allows much lower values for λ in the binomial case in [46]. We choose to not do this in our experiments because of the high variance resulting from using such low values. Using values such as $\lambda = 0.55$ results in very fast convergence, but also high variance from the true underlying parameter, resulting in unreadable plots. To get a "smooth" result, one would have to perform ensembles of almost 10000 simulations. Therefore, we chose to leave out such low values, and focus on higher values, drawn from the interval $[0.8, 0.95]$. These values has been corroborated through empirical testing. For the window parameter of the MLEW, we chose to draw values from the interval $[10, 80]$ for the binomial case, and $[15, 80]$ for the multinomial case.

For this test case we ran simulations with 500 time steps on an ensemble of 2000 simulations. The true underlying value of θ_1 was randomly changed every 50 time steps. We did extensive testing with random variables, but in the interest of brevity, we include only the results from three of these experiments here. Table 5.2 shows the parameters and the corresponding estimation errors from each of the simulations. The plots from each experiment is presented in Figs. 5.7, 5.8 and 5.9. The results we show here are typical.

			Averaged Estimation Error	
Experiment	λ	w	SLWE	MLEW
1	0.9064	29	0.0986	0.1084
2	0.9132	48	0.1152	0.1888
3	0.8280	19	0.1188	0.1248

Table 5.2: Test Case II - Experiments with regular change points

We observe that when the window size of the MLEW becomes larger, e.g., 48, the estimation algorithm is unable to track the changes in the environment, yielding poor overall accuracy. In Fig. 5.8, we see that the MLEW approximates the true value fairly well for the first 50 time steps, but is thereafter unable to track the variations. In both Figs. 5.7 and 5.9, when the window size is smaller, the MLEW is capable of tracking the changes, but not nearly as fast nor as accurate as the SLWE. Even when the SLWE use a learning parameter as low as 0.8280, it clearly outperforms the MLEW both with regard to speed and accuracy.

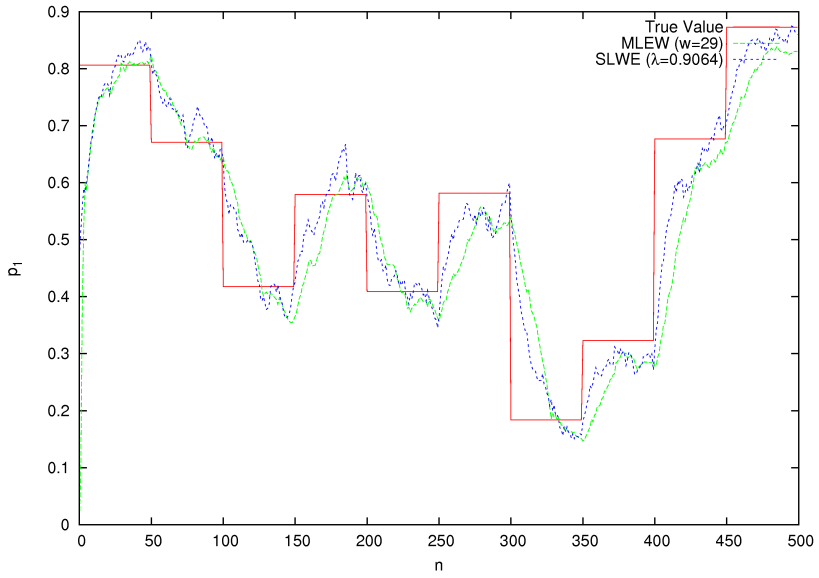

Figure 5.7: The expected value of the estimates of $p_1(n)$, for experiment 1 in test case two, obtained from the SLWE and MLEW, using $\lambda = 0.9064$ and $w = 29$.

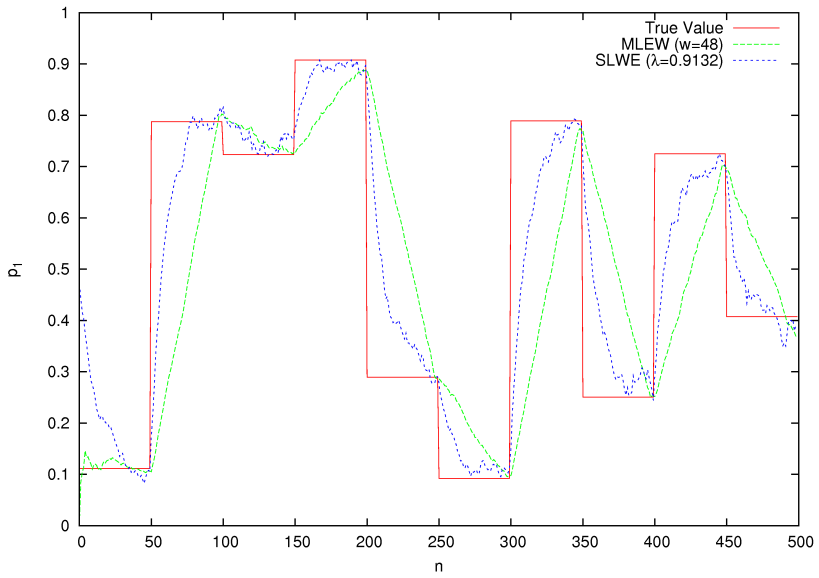

Figure 5.8: Experiment 2 for the second test case. The averages for the estimates of $p_1(n)$, which was estimated using the SLWE and the MLEW, where $\lambda = 0.9132$ and the window size is 48.

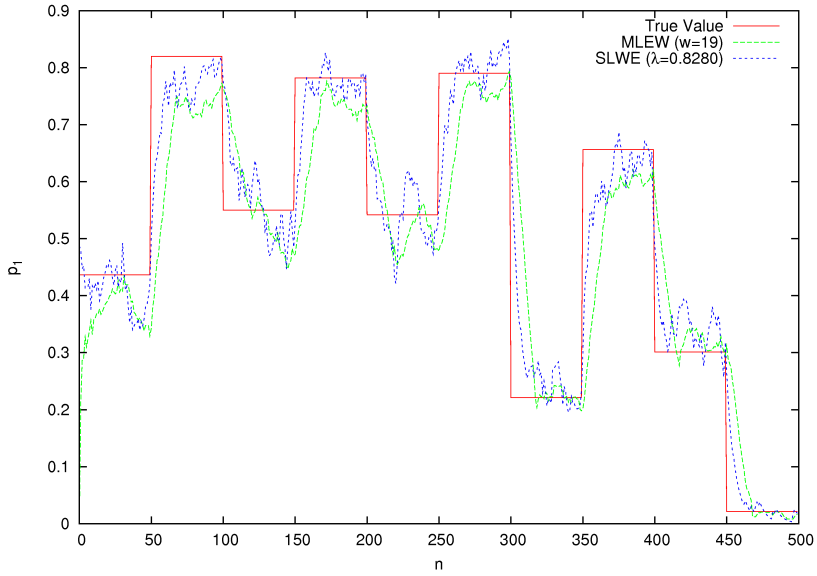

Figure 5.9: Plot of the estimates of $p_1(n)$ for experiment 3, for the second test case. Here, the estimates was obtained from the SLWE and MLEW using $\lambda = 0.8280$ and $w = 19$.

For the last part of this test case we let the environment switch its Bernoulli parameter with a certain probability at every time step. The reason for doing this is to see how well the estimation algorithms perform in environments when there is no prior knowledge about the magnitude or frequency of the changes. For each experiment, we ran an ensemble of 2000 simulations, each with 500 time steps. Change in the environment was done at every time step w.p. 0.0150, resulting in an average of $7,5$ changes per simulation. The results are outlined in Table 5.3 and the plots are shown in Figures 5.10, 5.11 and 5.12.

			Averaged Estimation Error	
Experiment	λ	w	SLWE	MLEW
4	0.9193	73	0.0973	0.1372
5	0.8735	54	0.0935	0.1213
6	0.8969	17	0.0965	0.1093

Table 5.3: Test Case II - Experiments with random change points

These experiments show that the SLWE adjusts to the changes faster than the MLEW, and as expected, in a geometric manner. Again we observe that high window sizes yield poor results for the MLEW. It is also unable to track small changes. Fig. 5.10 demonstrates this, and we see how the SLWE is capable of doing so at time step $n = 75$.

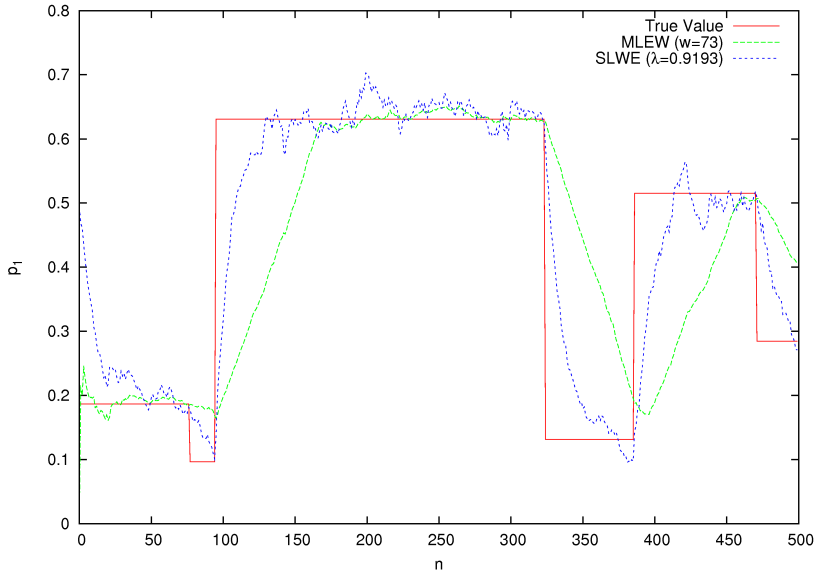

Figure 5.10: Plot for experiment 4, depicting the averages for the estimates of $p_1(n)$, obtained from the SLWE and the MLEW. The parameter of the environment was switched randomly at every time step with probability 0.0150. In this case, $\lambda = 0.9193$ and the window size is 73.

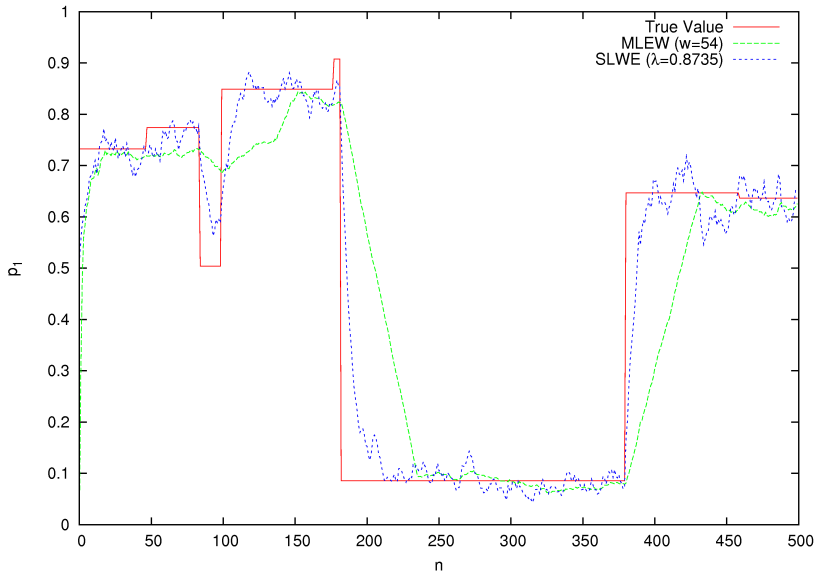

Figure 5.11: The expected value of the estimates of $p_1(n)$, at time n, for experiment 5 in the second binomial test case.

Similar results are shown in Fig. 5.11, where the window size is 54.

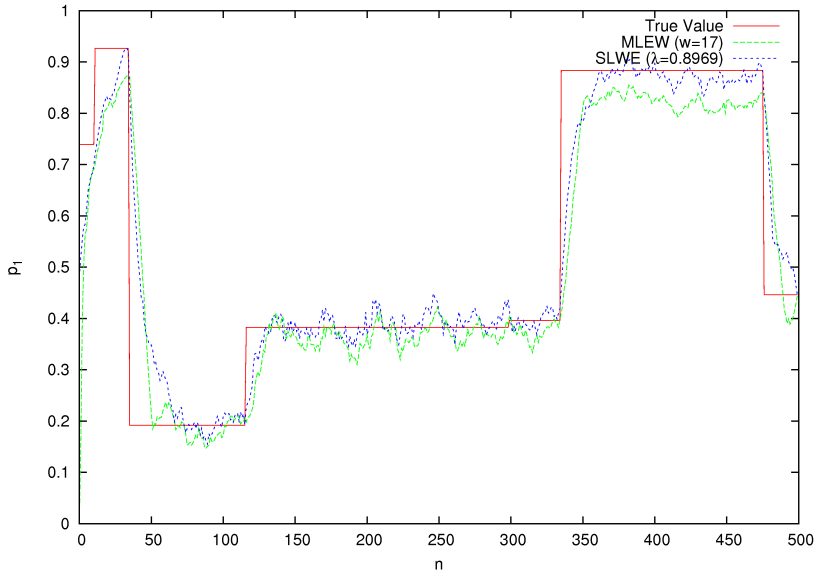

Figure 5.12: Experiment 6 in the second test case. Here, the estimates were obtained by the SLWE and MLEW, using $\lambda = 0.8969$ and $w = 17$.

In the last experiment, shown in Fig. 5.12, we see that even with a small window size, such as 17, the MLEW converges fairly quick, but with a lower accuracy than the SLWE. This is notable especially when the magnitude of the change is large, such as in the range from $n = 350$ to 475. Here, the SLWE has a clear advantage over the MLEW.

Test Case III: Frequency of Change

In the last test case we wish to investigate how the estimation algorithms are able to adapt to different frequencies of change in the environment. To show this we investigate two cases; one where the change rate is every 25 time steps, and one where the changes are done every 100 time steps. Again, the experiments were performed on an ensemble of 2000 simulations. For the experiments when the parameter was changed every 25 time steps, we ran each simulation with 400 time steps. In the case when it switched every 100 time steps, we used 500 samples per simulation. For each of these two scenarios, we ran two experiments with different window sizes. The results are outlined in Table 5.4. Note the Δ column, representing the change frequency for the experiment.

The plots of these experiments are shown in Figures 5.13, 5.14, 5.15 and 5.16.

				Averaged Estimation Error	
Experiment	Δ	λ	w	SLWE	MLEW
1	25	0.81	10	0.1282	0.1551
2	25	0.81	20	0.1282	0.1787
3	100	0.9	40	0.0863	0.1007
4	100	0.9	100	0.0863	0.1745

Table 5.4: Test Case III - Experiments

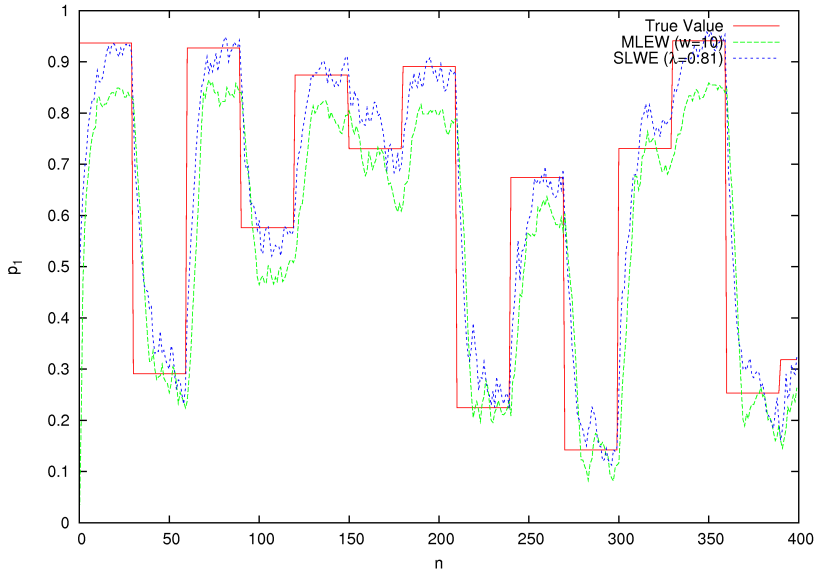

Figure 5.13: The expected value of the estimates of $p_1(n)$, for the first experiment in the third test case, obtained from the SLWE and the MLEW, where $\lambda = 0.81$ and the window size is 10. The environment switched its parameter randomly, every 25 time steps.

We notice how the SLWE comes much closer to the true value than the MLEW throughout the experiment, shown in Fig. 5.13, even when the window size is as small as 10. By using a larger window size, we observe in Fig. 5.14 that the MLEW comes closer to the true value but that the SLWE is still faster and yields a lower estimation error than the MLEW.

Figure 5.15 depicts the experiment when the change in the environment occurs every 100 time steps. We observe that a large window (such as 100 in this case) results in the MLEW not being able to track the changes. Smaller window sizes result in slow convergence and less accuracy than the SLWE, as illustrated in the plot in Fig. 5.16.

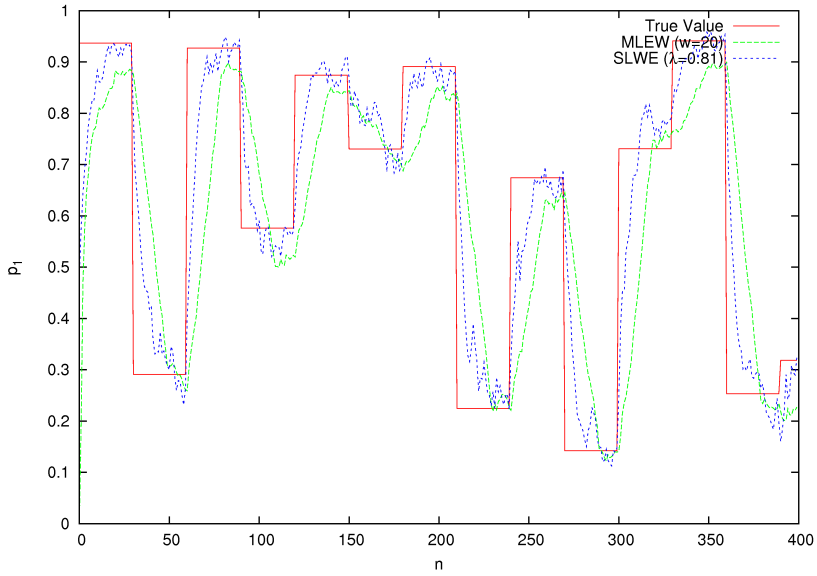

Figure 5.14: Plot of the averages for the estimates of $p_1(n)$ in experiment 2 for the third test case, which was estimated by using the SLWE and the MLEW. Here $\lambda = 0.81$ and $w = 20$.

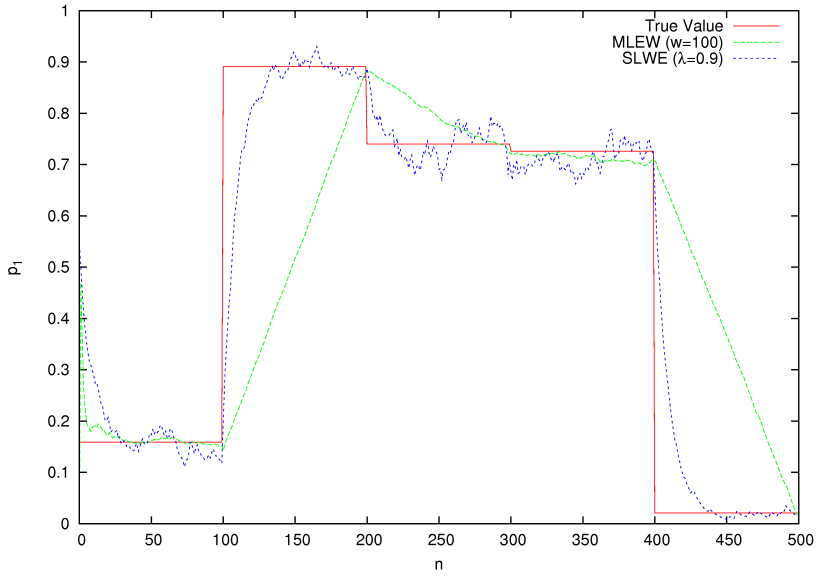

Figure 5.15: The expected value of the estimates obtained from the SLWE and the MLEW, using $\lambda = 0.9$ and $w = 100$, for the third experiment. Here the environment switched its parameter randomly every 100 time steps.

It is well-known that the choice of the window parameter depends on knowledge about both the frequency and the magnitude of the change. Using a window size of 80 when the change

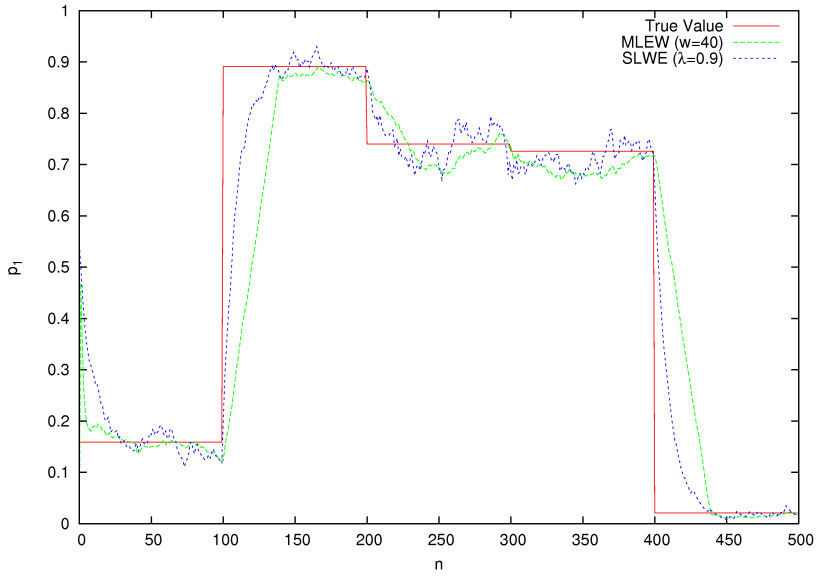

Figure 5.16: Binomial test case three, experiment 4. Plot of the averages for the estimates of $p_1(n)$, obtained from the SLWE and the MLEW. In this case, $\lambda = 0.9$ and $w = 40$.

occurs every 100 time step will yield a good result, but if the change occurs every 20 time step, a window size of 80 will result in the estimate not being able to trace the changes at all. This is a significant weakness of the MLEW which we have observed in the results from these simulations.

From these results we assert that the SLWE scales better than the MLEW and that it is adaptable to changes in the environment with regard to magnitude and frequency.

5.4 Simulations With Multinomial Distributions

In this section we investigate the performance of the two estimation algorithms on multinomial random variables. As explained in the previous chapter, we here deal with estimating a probability *vector*. Instead of dealing with two possible values like we did in the binomial simulation, we here deal with a multinomially distributed random variable, $\boldsymbol{x}$, which takes on values from the set $\{1, \ldots, r\}$. We assume that this random variable is governed by the distribution $\boldsymbol{\theta}$, which in the multinomial case takes on the form $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_r]^T$. Here, $\boldsymbol{x} = i$ w.p. θ_i.

5.4.1 Simulation Setup

Again, we deal with non-stationary environments, and the true underlying probability vector is switched randomly at given time steps. The environment generates a sequence containing r possible values, based on its probability vector. The number of trials (which is the number of observations or samples given to the estimators) is represented by N. As in the binomial simulations, we report the ensemble average at each time step for E simulations. In the multinomial case we plot the averaged Euclidean distance from the estimated vector to the true probability vector, and we also report the ensemble performance measures for each estimation algorithm. The running estimate for the SLWE was initialized to $p_i = (1/r)$ for $i = 1, \ldots, r$.

5.4.2 Results

The multinomial simulations are divided into three different test cases. The first involves finding suitable values for the multinomial case, the second involves testing randomly drawn parameters from such suitable values, and then comparing the performance of the two learning algorithms. We select these values randomly to perform a fair evaluation of the two methods. The last test case was performed to assess the scalability of the estimation algorithms with regard to dimensionality. From real-life applications such as Patter Recognition systems, we know that the dimensionality of the feature space may be very high, especially when dealing with tasks such as text classification. It is therefore of interest to see how well the methods scale when the number of features, which in our case is represented by the variable r, grows large. In the first two test cases we operate with $r = 4$, and in the last test case we perform experiments with $r = 10$, $r = 100$ and $r = 500$.

Test Case I: Finding Suitable Parameter Values

For the first multinomial test case we ran the simulations with $N = 400$ time steps on an ensemble of $E = 1000$ simulations. This test case deals with a multinomial probability vector associated with the set of $r = 4$ possible outcomes. For the multinomial case we used parameter values for the SLWE drawn from the same interval as Oommen *et. al.* use in [46], which is in accordance to what is suggested in the literature on multi-action LA schemes [43].

The test case is divided into 6 different experiments. For the first three experiments, we tested different values for the window size of the MLEW and used $\lambda = 0.95$ for the SLWE. The probability vector of the environment was switched randomly every 50 time steps. In all three experiments

we used the same environment for comparison, meaning that the same probability vector was used to generate the sequences for all three experiments, and all three experiments deal with estimating this probability vector.

In experiment 4, 5 and 6 of this test case we used $\lambda = 0.9$ as the learning parameter of the SLWE and also here we experimented with different values for the window parameter of the MLEW. The environment switched its parameter vector every 50 time steps, and the same environment was used in all three experiments. In the binomial simulations we used values for the window parameter of the MLEW as low as $w = 10$, but for the multinomial case we empirically found that this value yielded very low performance measures, and was therefore regarded as an unsuitable parameter value. Empirically we tested with $\lambda = 0.95$ and $w = 10$, and the resulting performance measures were 0.1941 for the SLWE and 0.2530 for the MLEW.

The results from our six experiments are outlined in Table 5.5. The plots of the ensemble average of the Euclidean distance between the estimated probability vector and the true underlying probability vector, for each estimation algorithm, is depicted for some of these experiments in Figs. 5.17, 5.18, 5.19 and 5.20.

			Averaged Estimation Error	
Experiment	λ	w	SLWE	MLEW
1	0.95	20	0.1950	0.2135
2	0.95	40	0.1997	0.2249
3	0.95	70	0.1942	0.2721
4	0.9	20	0.1964	0.2036
5	0.9	40	0.1951	0.2033
6	0.9	60	0.1964	0.2302

Table 5.5: Test Case I - Experiments

The first plot, depicted in Fig. 5.17, demonstrates how a small window width for the MLEW results in fast convergence but with higher estimation error than the SLWE. In this experiment we observe that the SLWE converges quickly despite using a fairly high value for the learning parameter, in this case $\lambda = 0.95$. From the reported performance measures in Table 5.5, we see that the SLWE clearly outperforms the MLEW.

When we increase the window width of the MLEW to $w = 40$, we notice that the estimation error is small and it converges to approximately the same value as the SLWE does. Despite this, the SLWE has the advantage of converging faster than the MLEW, and therefore yields a lower averaged estimation error than the MLEW. The reported averaged estimation error of the SLWE

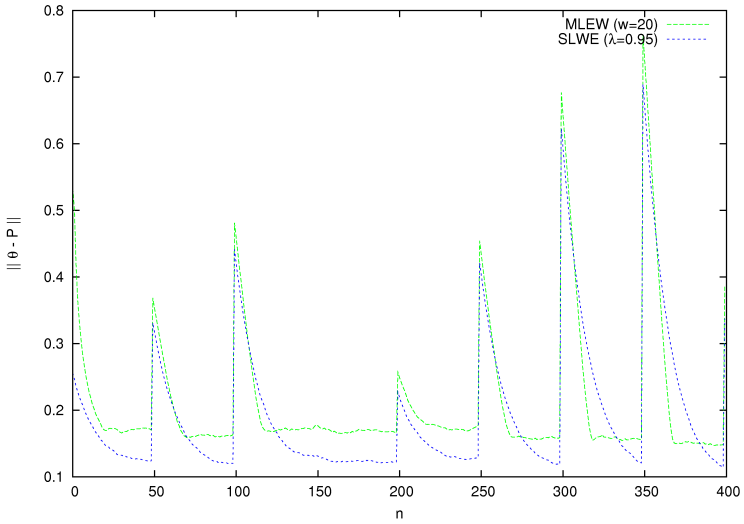

Figure 5.17: Plot of the averaged Euclidean distance between the estimates and the true underlying parameter vector, for both the SLWE and the MLEW, for the first experiment in the first multinomial test case. Here, $\lambda = 0.95$ and $w = 20$.

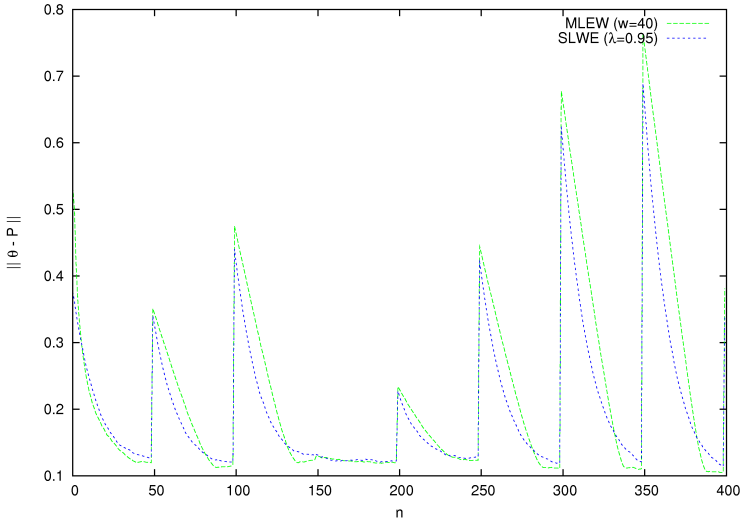

Figure 5.18: Multinomial test case one, experiment 2. Plot of the Euclidean distance between P and $\boldsymbol{\theta}$, where P was estimated using both the SLWE and the MLEW, where $\lambda = 0.95$ and the size of the window is 40, respectively.

for experiment 2 was 0.1997 and the estimation error of the MLEW was 0.2249. The plot for this experiment is depicted in Fig. 5.18.

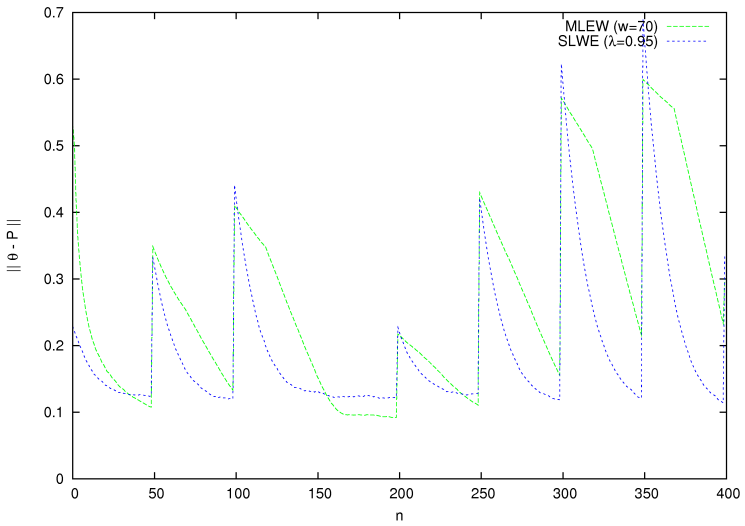

Figure 5.19: Experiment 3 for the first multinomial test case. Plot of the Euclidean norm of $P - \boldsymbol{\theta}$, for both the SLWE and the MLEW, where λ is 0.95 and w is 70.

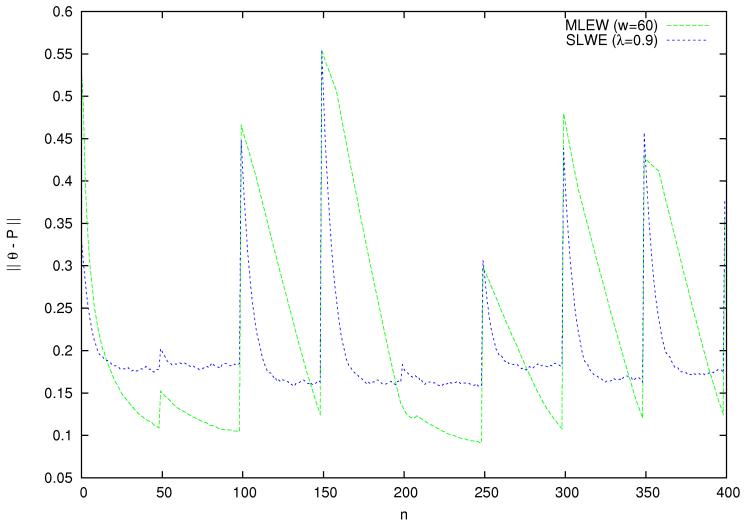

Figure 5.20: Multinomial test case one, experiment 6. Plot of the Euclidean distance between the estimate of $\boldsymbol{\theta}$, P, and $\boldsymbol{\theta}$, estimated by using both the SLWE and the MLEW. In this case, $\lambda = 0.9$ and $w = 60$.

For the binomial case we observed that the MLEW was not able to track the changes in the environment properly when the window size is relatively large. We observe these results for the the multinomial case as well, which we clearly see in the plots depicted in Figs. 5.19 and 5.20.

From the results of the last three experiments in this test case, we observe that a low value for the learning parameter of the SLWE results in fast convergence, but also a larger variance from the true probability vector. These results are supported by experiment 5 where we observe the results from increasing the window width, allowing the MLEW to converge with lower variance. However, despite the low variance, the MLEW shows a clear handicap with regard to its speed of convergence compared to the SLWE. Due to the faster convergence of the SLWE, it is able to yield superior performance measures over the MLEW, even with as low values as $\lambda = 0.9$. The reported estimation error of the SLWE in this experiment was 0.1951, compared to the estimation error of the MLEW which was 0.2033.

Test Case II: Random Values

For the second test case we experimented with $r = 4$ using randomly drawn values for λ and w. The objective was to assert the efficiency and robustness of the estimation algorithms in a fair manner. The experiments were performed on an ensemble of 1000 simulations, each with 500 time steps. The environment randomly changed its probability vector every 50 time steps. The values for λ were drawn randomly from the interval $[0.9, 0.99]$, and the values for the window parameter were drawn randomly from the interval $[15, 80]$. As for the binomial case, we did extensive testing with different parameter values, but in the interest of brevity, we only include the results from 6 of these experiments here. The values used was $0.9682, 0.9353, 0.9590, 0.9285, 0.9541$ and 0.9625 for the learning parameter of the SLWE, and $74, 23, 56, 66, 17$ and 46 for the window parameter of the MLEW. The experiments and their respective performance measures are outlined in Table 5.6. The plots of the Euclidean distance between each estimate and the true probability vector of the environment for the six experiments are depicted in Figs. 5.21, 5.22, 5.23, 5.24, 5.25 and 5.26.

			Performance measure	
Experiment	λ	w	SLWE	MLEW
1	0.9682	74	0.1760	0.2178
2	0.9353	23	0.1842	0.1995
3	0.9590	56	0.1801	0.2201
4	0.9285	66	0.1766	0.2070
5	0.9541	17	0.1976	0.2161
6	0.9625	46	0.1643	0.1762

Table 5.6: Test Case II - Experiments

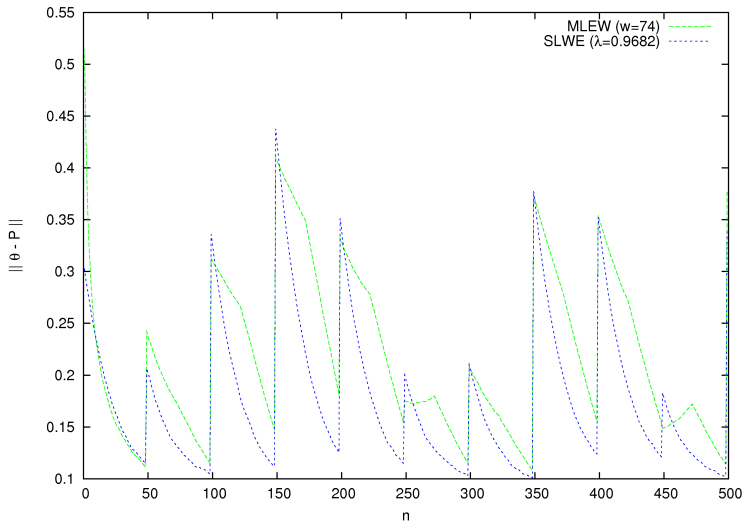

Figure 5.21: Plot of the Euclidean distance between P and $\boldsymbol{\theta}$, for the first experiment in the second multinomial test case. P was estimated by both the SLWE and the MLEW, where $\lambda = 0.9682$ and $w = 74$.

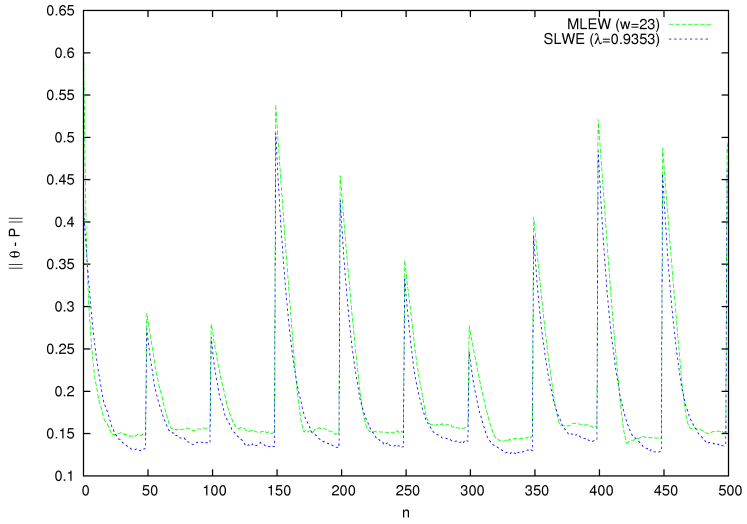

Figure 5.22: Multinomial test case two, experiment 2. Plot of the Euclidean distance between the estimates and the true parameter vector of the environment. The estimates were obtained for both the SLWE and the MLEW, where $\lambda = 0.9353$ and $w = 23$.

These experiments demonstrate how a low window parameter of the MLEW yields lower accuracy than the SLWE with regard to approximating the true probability vector. Using a too large window width results in slow convergence. Fig. 5.21 depicts the plot of using a window width of $w = 74$, and we observe that even though the MLEW is able to trace the true probability

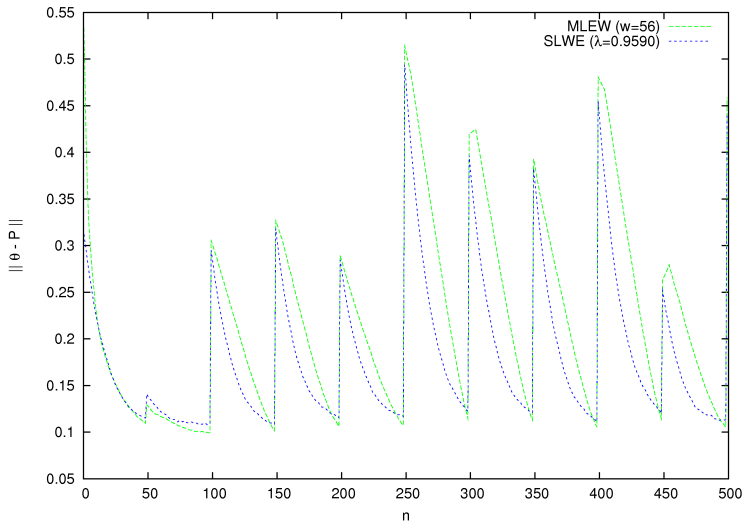

Figure 5.23: Plot of the Euclidean norm of $P - \boldsymbol{\theta}$, for both the SLWE and the MLEW, where $\lambda = 0.9590$ and $w = 56$, respectively. (Experiment 3, test case two.)

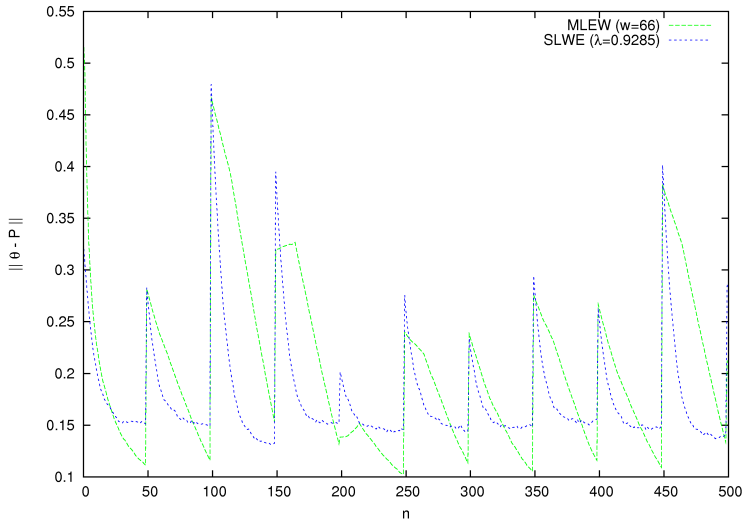

Figure 5.24: Experiment 4, test case two. Plot of the Euclidean distance between P and $\boldsymbol{\theta}$, where P has been estimated using both the SLWE and the MLEW. Here, $\lambda = 0.9285$ and $w = 66$.

vector fairly quickly for the first window, in successive epochs, it is unable to keep up with the changes, and thus clearly demonstrates its weakness compared to the SLWE.

It is important to note that even though the MLEW is able to trace the true probability vector with fairly low variance for instance in experiment 4, shown in Fig. 5.26, the slow convergence results in poor overall performance compared to the faster SLWE.

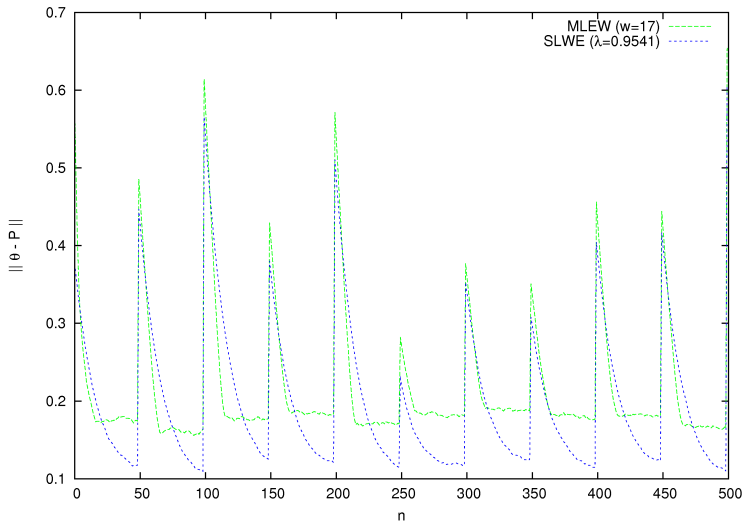

Figure 5.25: Plot of the Euclidean distance between P and $\boldsymbol{\theta}$, for the fifth experiment in the second multinomial test case. P was estimated by both the SLWE and the MLEW, where $\lambda = 0.9541$ and $w = 17$.

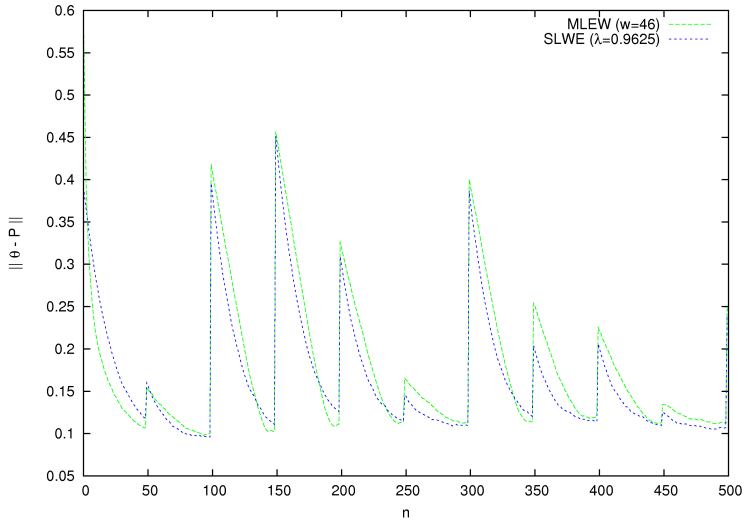

Figure 5.26: Plot of the Euclidean distance between P and $\boldsymbol{\theta}$ for experiment 6 in test case two. P was estimated by using both the SLWE and the MLEW, where λ is 0.9625 and the size of the window is 46.

Test Case III: Dimensionality and Scalability

In the last test case for the multinomial experiments, we try different values for r to assert the scalability of the two estimation algorithms. The experiments were performed on ensembles of

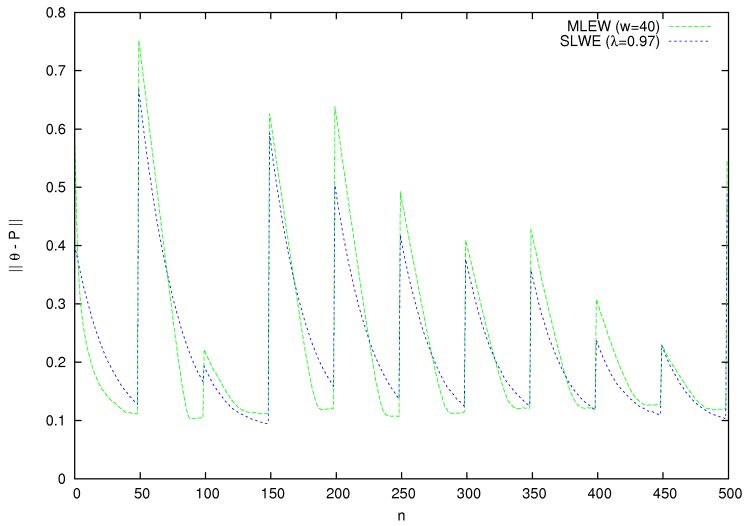

Figure 5.27: Plot of the Euclidean distance between the estimates and the true value of the parameter vector of the environment for experiment 1, in the third test case. Here, the dimensionality of the parameter vector was $r = 4$. The SLWE and the MLEW used $\lambda = 0.97$ and $w = 40$, respectively.

1000 simulations having 500 time steps. Again, we operate with changes in the environment every 50 time steps. To demonstrate the ability to scale with a large number of features we ran our experiments with the same values for λ and w for $r = 4, 10, 50$ and 100. We tested with different values for the learning parameter and the window size, but in this section we present the results from using $\lambda = 0.97$ and $w = 40$, since these values empirically demonstrated good results with $r = 4$ for both the MLEW and the SLWE. The results from these experiments and their corresponding performance measures are outlined in Table 5.7. We also include the plots of the euclidean distance between the estimates and the true value of the probability vector of the environment, depicted in Figs. 5.27, 5.28, 5.29 and 5.30.

				Performance measure	
Experiment	λ	w	r	SLWE	MLEW
1	0.97	40	4	0.2257	0.2289
2	0.97	40	10	0.1608	0.1853
3	0.97	40	50	0.1343	0.1713
4	0.97	40	100	0.1267	0.1664

Table 5.7: Test Case III - Experiments

From these results we observe that when the number of possible outcomes, r, increases, the

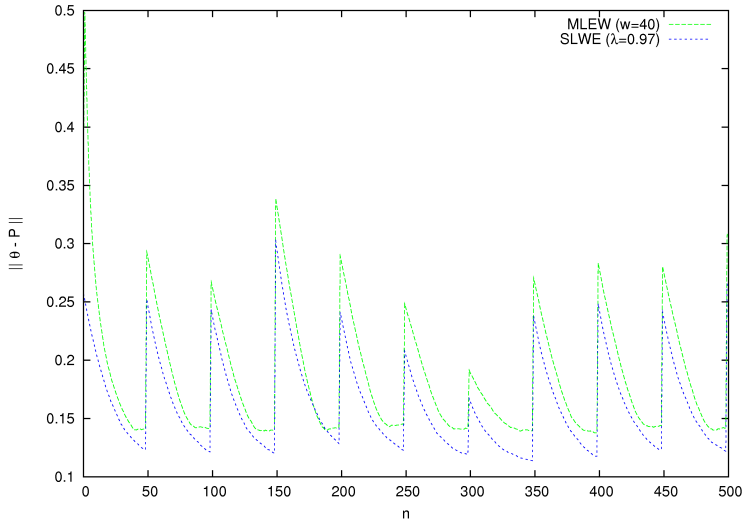

Figure 5.28: Plot of the Euclidean distance between the estimates and the true value of the parameter vector of the environment for experiment 2, in the third test case. Here, the dimensionality of the parameter vector was $r = 10$. The parameters used by the SLWE and the MLEW was $\lambda = 0.97$ and $w = 40$, respectively.

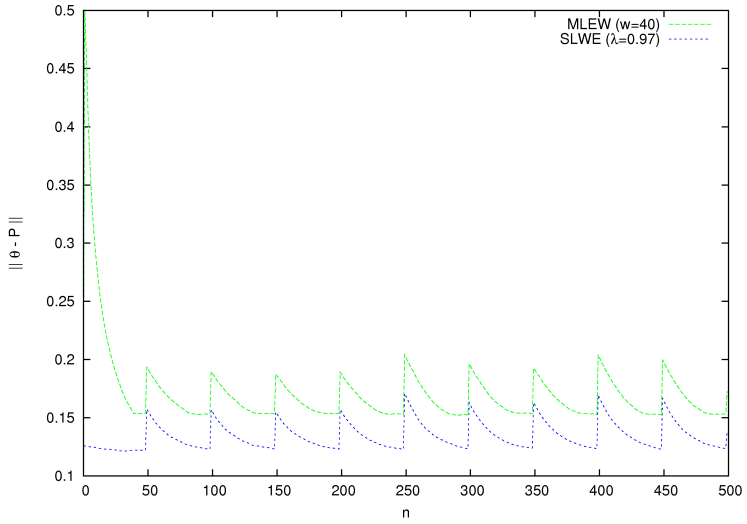

Figure 5.29: Plot of the Euclidean distance between the estimate, P, and the true parameter vector, $\boldsymbol{\theta}$, for experiment 3 for the third test case. The dimensionality of the parameter vector in this experiment was $r = 50$.

SLWE demonstrates its power over the MLEW with regard to both speed of convergence and accuracy in the sense of its Mean Squared Error. We observe that the reported averaged performance measures of the SLWE are significantly higher than the reported measures for the MLEW. These

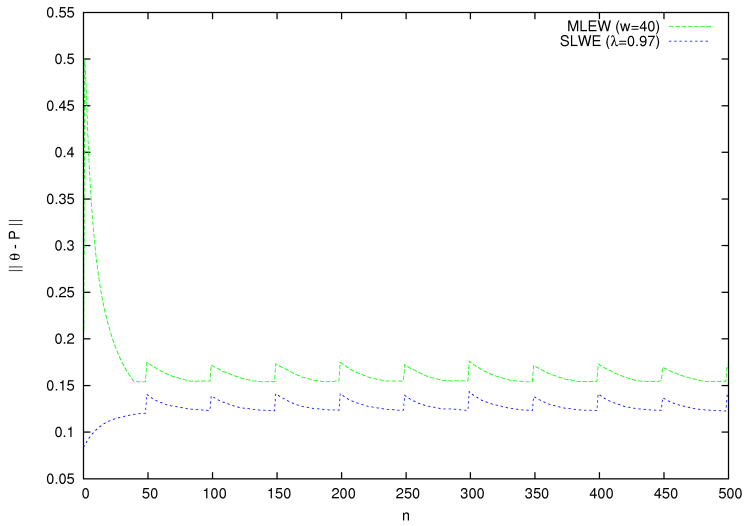

Figure 5.30: Plot of the Euclidean norm of $P - \boldsymbol{\theta}$, where P was estimated by using both the SLWE and the MLEW, where λ is 0.97 and the window size is 40. In this experiment the number of parameters in the parameter vector was $r = 100$.

results are of high importance to our research focus on estimation methods in PR systems. The scalability of the SLWE is a great motivation for using this estimation algorithm in non-stationary environments applied to Pattern Classification tasks such as text classification, where we know that the feature space may be very large.

5.5 Summary of Results

Our empirical laboratory experiments presented in this chapter have demonstrated the superiority of the SLWE over the MLEW, operating in non-stationary environments. For both binomial and multinomial distributions we have observed that even with a fairly low value for the learning parameter, λ, of the SLWE, it was able to track the changes in the environment fast and with a sufficiently low variance. In all of our simulations, the SLWE yielded superior performance over the MLEW.

It is important to point out that the window size of the MLEW can be tuned to better adapt to the changes in the environment, and approach the same performance as the SLWE. However, this requires knowledge about the nature of the environment with regard to the frequency and magnitude of the changes. This *a priori* knowledge is generally unknown in real-life application

areas, such as PR-systems dealing with data which possess a high level of non-stationarity.

We observe in our experiments that the MLEW is capable of tracking the changes of the parameters when the window size is small, or at least smaller than the intervals of change in the environment. However, when the size of the window is relatively large, the MLEW is not able to track the changes properly. This scenario demonstrates the weakness of the MLEW, and its dependence on the knowledge of the input parameters. We have also observed that when the size of the window is small, the accuracy of the MLEW is severely reduced, and thus yields poor overall performance. Extensive experiments with regard to the scalability also shows that the SLWE is capable of adjusting to different changes in the environment without having to change the learning parameter.

The theoretical proofs from the previous chapter regarding the convergence properties of the SLWE has been demonstrated through experiments, and we infer that a small value for λ leads to faster convergence and higher variance from the true underlying parameter that is being estimated. Large values for λ leads to slower convergence but with a smaller estimation error. Through our experiments we found values from the interval $[0.9, 0.99]$ to yield good results, which is also what is suggested in the literature on stochastic learning automata.

We also observe that it is sub-optimal to work with strong estimators in application domains involving non-stationary data, since it is indeed difficult for the learned distribution to emerge from a distribution that it has converged to.

The experiments presented in this chapter has demonstrated the power of the SLWE operating on synthetic data, but as Oommen *et. al.* also states, the intention is to demonstrate that the SLWE is not merely a theoretical contribution, but possesses the property of leading to superior solutions in real-life problems. Another important motivating result from our experiments is that the MLEW is inferior to the SLWE when it comes to dealing with estimating a multinomial distribution when the number of features becomes large. The ability do deal with a large feature space is an important factor in real-life problems such as PR-systems, and gives us reason to believe that the SLWE will demonstrate good results applied to such problems. In the next chapter we present a novel approach to one such PR system, dealing with language classification in so-called multilingual documents.

Chapter 6

A Novel Approach to Language Classification in Multilingual Documents Based on SLWE

The previous chapter demonstrated the power of the SLWE in non-stationary environments dealing with synthetic data. The SLWE showed its ability to scale well with regard to the number of features and also with regard to the frequency and magnitude of changes in the environment. In this chapter we will investigate PR-systems dealing with text classification, where the class-conditional distributions of the features are non-stationary. In particular we present a novel approach to language classification in so-called multilingual documents, based on the SLWE. Our approach involves a completely new way of classifying documents containing several different languages.

In Section 6.1 we briefly present some background on text-classification and in Sections 6.2 and 6.3 we introduce the state-of-the-art within the area of language classification, for so-called monolingual and multilingual documents, respectively. In Section 6.4 we present a novel approach to language classification in multilingual documents, and in Section 6.5 we introduce our proof-of-concept prototype. We demonstrate how this classifier operates with different languages through experiments in Section 6.6 before we conclude the chapter with a brief discussion of our results in Section 6.7. Since our prototype implementation is a proof-of-concept implementation, we do not directly compare our results to the results of the mentioned state-of-the-art approaches, but dedicate a brief section to this subject in Section 6.7.1.

6.1 Text Classification

The task of text classification can be defined as assigning a document to one or more pre-defined categories, based on its content. It is important to note the difference between classification and clustering. When clustering documents, one does not know the set of classes. In the task of text classification, one deals with a predefined set of classes. One example of this would be classification of news articles into categories such as *politics*, *sport* or *business*. This is often referred to as topic classification. Another text classification task is the classification of sentiment or opinions such as classifying movie reviews into *positive* or *negative* reviews. In this chapter we deal with a different task of text classification, namely language classification. The task at hand is to classify a set of documents into classes, based on the language they are written in. Even though this is the proof-of-concept task we have chosen, we believe that our approach can be applied to tasks such as topic or sentiment classification as well.

In particular, we deal with so-called Word of Mouth (WoM) discussions, such as discussions found on online discussion boards. The main difference between classification of news articles or journal papers and WoM discussions, is that discussions generally contain the opinions of several different authors, as described in Ch. 2. Considering a discussion where several authors write parts of it means that we have a document with continuous content changes. Not only may the subject or topic of each segment in the discussion change, but the discussion can have multiple sentiments and it can contain multiple languages as well.

Treating the whole discussion as one contiguous document, the task at hand is thus to segment the discussion and classifying each segment according to the pre-defined classes, whether they are topics, sentiment or language. Generally, online discussions are unbiased and the opinions of the different authors changes. Since many online discussion boards have users from all over the world, the users may also write in different languages. A discussion containing multiple languages is exemplified in Fig. 6.1.

Another important aspect of text classification of such WoM discussions is that the postings often are composed on the fly by the different users, without any form of spell checking. Thus, when performing text classification on such data, one must tolerate the presence of different kinds of textual errors, such as spelling and grammatical errors. Abbreviations and Internet “slang” may also be present. The classification process must work reliably on all input, and must tolerate these kind of errors to some extent.

In Ch. 2 we introduced the *term normalization process* which is usually applied in IR systems

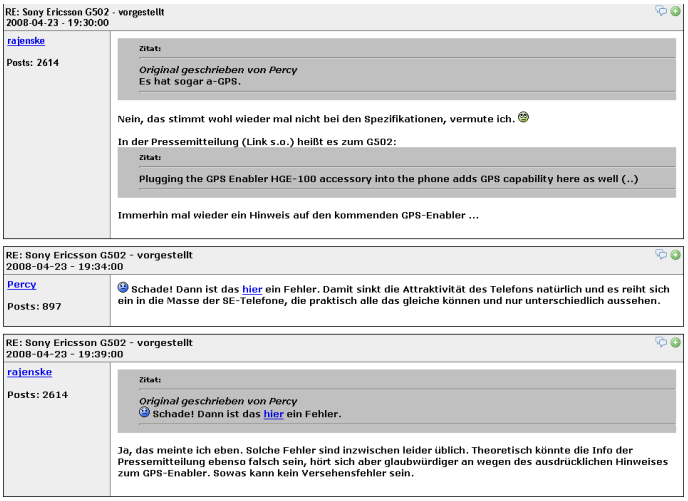

Figure 6.1: Example discussion containing English and German writing, collected from the German discussion board *www.telefon-treff.de*.

and also in text classification tasks. This process may consist of stop word removal, stemming and spell checking. These are all tasks that costs both time and memory and may result in slow speed when classifying. In addition, many of these tasks depend on knowledge about the language that the text is written in. Central to text classification is also the choice of features and document representation. We discussed the vector-space-model by Salton *et. al.* in Ch. 2. This model allows some form of weight associated with each token to be kept in the document vector. This is very useful when dealing with classification tasks involving a large feature space, such as text classification. Salton points out the term importance in a document in the following way:

> *"A single term can decrease the document similarity among document pairs if its frequency in a large fraction of the corpus is highly variable or uneven."*

When performing tasks such as topic classification, the TF-IDF scheme is often used for weighting the importance of a term in the different documents. When dealing with sentiment classification, one might not be interested in the same discriminatory features as in topic classification, and one would for instance look at the presence adjectives in the documents. In the case of language classification one will try to identify what features are unique for a given language, and use this when classifying. We will look further into this in the following section. Common for all of these classification tasks is that they deal with documents containing a large number of terms, and thus might result in high dimensionality of the feature space. We will look at different ways of reducing the dimensionality through selection or cut-off thresholds in the following section as well.

6.2 Language Classification in Monolingual Documents

Cavnar & Trenkle approached the task of text-classification in [6], by using N-Gram analysis. As Cavnar *et. al.* states, the basic idea is to identify so-called N-grams whose occurrences in a document gives strong evidence for or against identification of a text as belonging to a particular language. Their method proved to be a reliable and simple way to categorize documents for a wide range of classification tasks. The method is highly tolerant of data containing textual errors, which is an important motivation for using it in text-classification tasks that involve Word of Mouth discussions. Their method was presented and applied to both topic classification and language classification and demonstrated good results for both tasks. Before we explain the work of Cavnar & Trenkle in more detail, we dedicate a brief section to N-grams and Zipf's Law.

6.2.1 N-Grams

An N-gram is an N character long sub-sequences of a given sequence, for instance a string. There are several forms of N-grams, but with regard to text classification we are dealing with contiguous sequences, meaning that the order of the string is kept intact. Thus if the sequence is not contiguous, it would be possible to have N-grams made up of the second and fourth character of the given string. For language classification we are interested in the nature of a language and we are therefore only looking at contiguous parts of a string. Some approaches to processing n-grams remove spaces from the strings and it is common to do some *washing* or preprocessing of the input like removing punctuation etc. The general approach to generating N-grams of a given word is to take overlapping sequences of the the word of size N. For instance for uni-grams (where $N = 1$), the word *"sport"* would result in the following 5 uni-grams; 's', 'p', 'o', 'r' and 't'. The bi-grams (where $N = 2$), of the same word would be; 'sp', 'po', 'or' and 'rt'. The same approach is applied to tri-grams ($N = 3$) and N-grams of higher order. Cavnar *et. al.* use a similar approach for processing N-grams of several different lengths simultaneously, but in addition they pad each word with leading and trailing whitespaces. In general, a string of length *k*, padded with blanks, will result in $k + 1$ bi-grams, $k + 1$ tri-grams, and so on. From their approach, using underscore as a representation of a blank space, the word *"sport"* will yield the following 6 bi-grams; '_s', 'sp', 'po', 'or', 'rt' and 't_'. The same word represented as tri-grams would result in the following 6 tri-grams (where the word is padded with multiple trailing whitespaces); '_sp', 'spo', 'por', 'ort', 'rt_' and 't__'. By doing so, Cavnar *et. al.* gain a benefit when dealing with N-grams with regard to text classification, because the whitespaces conserves the knowledge of beginning and ending

of words. This is especially useful in language-classification where we have frequent prefixes and suffixes in a specific language, such as '-ing' or 'th-', which are frequent in the English language. Similarly, endings like '-er' and '-en' are high frequent endings in the Norwegian language.

As mentioned earlier, another important motivation for using N-grams in text-classification is that they are highly resistant to a wide variety of textual errors. Cavnar *et. al.* points out that this benefit is provided by N-grams since every string is decomposed into small parts, and any errors that are present tend to affect only a limited number of those parts. Thus, the remainder of the string is left intact.

Zipf's Law

Zipf introduced what is commonly referred to as Zipf's law in [67], stating that

> *"The* k*th most common word in a human language text occurs with a frequency inversely proportional to* k*."*

Zipf's law is often used to express the idea of human languages invariably having some words which occur more frequently than others. This implies that there is always a set of words which dominates most of the other words of the language in terms of frequency of use. For example, if we look at the Brown Corpus[1] [29], one will observe that the word "the" is the most frequently occurring word, and that it occurs approximately twice as often as the second most frequent word, which is the word "of". This word occurs approximately twice as often as the third most frequent word, which is the word "and". Analogous to topic classification, this law is also true for words that are specific to a particular subject, and can therefore be used for this application area as well. A Zipfian distribution of the top 1500 N-grams for the English language is outlined in Fig. 6.2. The N-grams were generated using a training corpus of approximately 600 English written documents, using mixed order N-grams for $N = 1$ to 4.

We observe that due to the the smoothness of the frequency curves, we do not have to worry too much about specific frequency thresholds. As stated by Cavnar *et. al.*, Zipf's law implies that classification of documents using N-gram frequency statistics will not be very sensitive to cutting off the distributions at a particular rank. It also implies that documents belonging to the same class should have similar N-gram frequency distributions.

[1] The Brown Corpus is a large collection of American English documents.

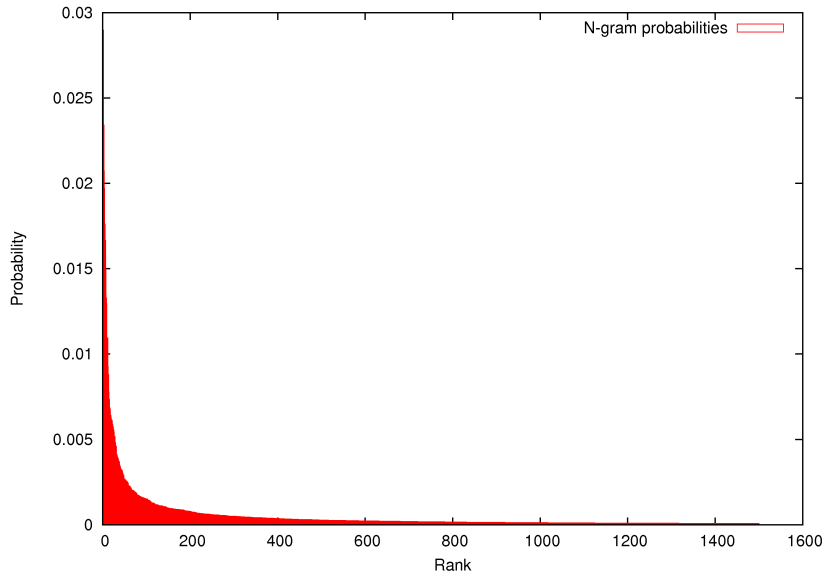

Figure 6.2: Zipfian distribution of the top 1500 mixed order N-grams for the English language.

6.2.2 Language Classification Using N-Grams

The approach to language classification by Cavnar *et. al.* is based on calculating and comparing profiles of N-gram frequencies for different languages. The first step of their approach is to compute such profiles on the training data that represent the various languages. After each language profile is computed, the classification process consists of computing N-gram profiles for the given document that is to be classified. Finally, their system computes a distance measure between the given document's profile and each of the language profiles. The document is classified as the language whose profile has the shortest distance to the document's profile.

The N-gram frequency profiles are generated in a straightforward manner; By using a collection of documents that are manually classified for a given language, the training phase of the approach by Cavnar *et. al.* consists of reading all documents in the collection and counting the occurrences of all N-grams. A phase of tokenization is done prior to this, were each document is broken down into separate tokens, punctuation is removed, and each term is padded with whitespaces. Salton's Vector Space Model is used to keep track of all N-grams, and the frequency count is incremented as the training phase progress. After each document is processed, the frequency counts are sorted into reverse order so that the N-grams with the highest frequency is kept in the top of the hash map. Once the map is sorted, the frequencies can be discarded and only the ranks (placement in the map) need to be kept. According to Zipf's law and the smoothness of the Zipfian

distribution, Cavnar *et. al.* chose to use the top 300 N-grams in the profile and discard the rest. By inspection, the highest ranking N-grams are uni-grams which reflect the letters in the alphabet of the language. Following these top N-grams comes the function words and very frequent pre- and suffixes.

In the classification phase, the document that is to be classified goes through the same process of having its N-gram profile calculated. A measure between the document's profile and each of the language profiles is taken, and the profile with the shortest distance from the document is chosen as the most likely language of the document. Cavnar *et. al.* use a rather simple ad-hoc rank ordering statistic, which they call an "out-of-place" measure. If an N-gram is ranked as rank 4 in the document profile, and rank 7 in the language profile, this N-gram will be 3 ranks out of place. The distance measure is thus the sum of all of these out-of-place values. If an N-gram is not found in the language profile, it is given some maximum out-of-place value. As Cavnar *et. al.* states, this way of measuring the distance is simple and intuitive and works well for their proof-of-concept tests, but more advanced statistical measures for ranked lists could profitably be used instead. The Wilcoxon rank sum test is mentioned as one of these possible measures [63].

For their language classification, Cavnar *et. al.* experimented with generating all possible N-grams, for $N = 1$ to 5. For their cut-off threshold, they experimented with different lengths of the language profiles but found 400 to yield the best results, with $99,8\%$ overall classification accuracy. Out of 3478, only 7 documents were misclassified. The documents being classified were rather long (4K bytes which is about 700 words in English or Spanish). They point out that the system was relatively insensitive to the length of the sample documents, but the shortest ones that they reported classifying were around 300 bytes, which would be a document with about 50 words in English.

The obvious advantage of their approach is that the system has modest computational and storage requirements, and is very effective. No semantic or content analysis apart from the N-gram frequency profile itself is required. It is ideally suited for text coming from noisy sources, such as email or OCR systems[2]. For our application area, we see the same parallel in Word of Mouth discussions, which are also subject to noise. Cavnar *et. al.* points out the advantage of using N-grams instead of whole word statistics, especially when dealing with shorter passages. This is also an essential characteristic of online discussions were the individual postings may be very short. By definition, there are more N-grams in a given sentence than single words, thus

[2]Optical Character Recognition systems are used for automatic recognition and interpretation of digitized text documents.

it will be easier to collect enough N-grams to be significant for classification. Also, using whole word statistics would be much more sensitive to errors – a single misspelled character would result in throwing off the statistics for a whole word.

Martins & Silva presented an approach to language classification using N-grams complemented with heuristics and a new similarity measure in [37]. Their approach is also based on Cavnar & Trenkles approach. The approach deals with language classification of web pages, and their heuristics involves giving more weight to N-grams found in the title field of the page, or in meta-tags in the HTML document. They also give weight to the largest contiguous text segment of the page.

6.2.3 Other Approaches to Language Classification

There are several different approaches to selecting features for language identification. These include for instance the presence of particular characters as discriminators [66, 39] or the presence of particular character N-grams [18, 4]. Another frequently used approach to language classification is the dictionary approach, where a dictionary is kept for each possible language, and a look up on every word in the sample document is done to see in which dictionary it occurs in. Then, the dictionary which contains the most words from the document indicates which language the document was written in. A simplified version of this approach is to use a lists of words that commonly appear in the languages of interest [21], instead of the full dictionary. For example, '*the*', '*of*' and '*and*' are common words in the English language. For Norwegian, '*jeg*', '*det*', '*å*' are examples of common words. When classifying a document and observing words like '*jeg*' occurring with a high frequency, it would be a good guess to assume that the text is written in Norwegian. Such non-linguistically motivated features generally perform well for documents of moderate length, but their performance is significantly decreased when the length of the sample text gets shorter.

More advanced approaches to language classification, using linguistic factors that differ among languages is also found in the literature. One such approach is based on the use of *morphological* features presented in [10]. Morphemes are the smallest components of a language that carry semantic value. The problem with this approach is that construction of a morphological lexicon for a given language requires a large amount of work by trained experts. There are ways of generating simplified morpheme "language models", and one such algorithm that has demonstrated high accuracy for various languages, has been developed by Creutz [11]. Creutz uses an

Hidden Markov Model to model morpheme sequences, without assuming prior knowledge of the morphemes themselves, nor of their individual functional categories.

There are many possible classifiers to choose from when dealing with language classification and among these, the most frequently used are the Naïve Bayes classifier, the k Nearest Neighbor (kNN) classifier, or Support Vector Machines (SVM) [48, 27, 25]. Approaches using Artificial Neural Nets (ANN) are also found in the literature [8]. Rocchio's algorithm is another popular approach to text classification [52]. In this approach, classification is done by using the distance to the centroid of all documents from each class. We will not delve into the details of these approaches, but include them here for reference.

Common for all of these approaches to language classification is that they deal with monolingual documents, meaning that the documents are written in one language. In this thesis we are interested in language classification of multilingual documents. We believe that the same approach could be applied to problems such as topic classification where one document could be segmented into smaller parts with different topics, or sentiment classification where a Word of Mouth discussion contains the opinion of several different users. We will present the state-of-the-art of language classification in multilingual documents in the following section.

6.3 Language Classification in Multilingual Documents

Traditionally, the task of language classification has been applied to documents where the entire document is assumed to be written in a single language. As we have seen, classification of monolingual documents is widely studied and a rather trivial task. Multilingual documents are documents with multiple languages within each document. Moreover, we do not know the boundaries of the different language segments in the multilingual document. One example of a multilingual document is news articles that contain citations in a foreign language or phrases from one language appropriated into the context of another. In this case a span of text in a document differs from the primary language of the document. Another example is online discussions were the posters may alternate between posting in different languages. For instance, a Spanish discussion board may have users posting both in Spanish and English. The task is illustrated in Fig. 6.3.

The same approaches that performs well in monolingual documents need not work at all in multilingual documents because a single-language segment of the document may be too short to

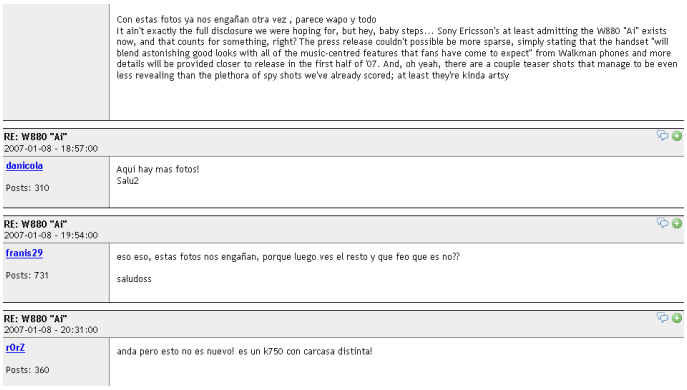

Figure 6.3: Example discussion containing English and Spanish writing, collected from the Spanish discussion board *www.gsmspain.com*.

contain sufficient characters for the character-based classification methods.

Ozbek et. al. presents an approach in [47], where they make use of Creutz algorithm which we briefly mentioned in the previous section. Their approach demonstrated good results for the Turkish language, but for the English language, the results were discouraging, with worst performance as low as 40% accuracy.

Ludovik & Zacharski propose an algorithm for classifying multilingual documents that is based on mixed-order n-grams, Markov chains, maximum likelihood and dynamic programming in [34]. They use a similar approach to that of Cavnar & Trenkle, by building a mixed order n-gram model for each language. Their approach differs in that they do not use an ad hoc rank-order distance measure, but instead use a maximum likelihood approach. To deal with the multilingual nature of the documents, Ludovik *et. al.* use a dynamic programming algorithm based on a Markov model of multilingual documents. This approach is used to segment the documents into monolingual parts, and each segment is classified individually. They also include a verification step to determine how well every segment fits the language assigned to it. Their approach shows promising results with an average error rate of 4.7% for average segment sizes of 49 bytes, and 12.88% for segments of an averaged size of 20 bytes.

Language classification in multilingual documents using a word-window approach has been studied in [35] by Mandl *et. al.* In their approach, they use tri-grams and a fixed window size of eight words. Then, for each window, the most probable language is determined based on the transition probability between tri-grams. For windows in which a different language occurs, a language change is assumed and the position of the change is determined by the first position of the window, plus two words. Their results showed high accuracy for detecting the languages,

but they point out that the location of the language shift is the hardest challenge, reporting a cumulative precision of 81% for locating the change point with at most 2 words off the real change point. Their approach has the same issues with regard to computational complexity as the MLEW method examined earlier in this thesis, being that they need to keep all the words in the entire window in memory at each step of the classification.

Multilingual language classification could also be used in OCR systems where the document being read is written in several different languages, or machine translation systems that seek to translate documents that are multilingual in their nature.

6.4 Proposed Solution

In this section we present a novel approach to the task of language classification in multilingual documents. By using the Stochastic Learning Weak Estimator (SLWE), presented earlier in this thesis and combining this with mixed order N-gram models, we introduce a PR system for classifying multilingual documents.

The approach is based on the ideas of Cavnar & Trenkle, using mixed order N-gram models, and building N-gram profiles for each language that is being classified. In the following sections we describe our algorithm and introduce a proof-of-concept prototype which we also present experimental results for. One of the main motivations for using mixed order N-gram models in our approach is the ability to handle noise and that it is possible to get substantial information from shorter phrases. There is no need for preprocessing in the sense of spell checking or stemming since N-grams essentially gives us the information bearing content of a word without performing such costly procedures. In addition, stemming requires sophisticated knowledge about the language, and is thus useless for our task since we do not know the language of the input text. We have empirically demonstrated the SLWE estimating synthetic data and shown good results with regard to adapting to changes in the environment, both in the sense of frequency and magnitude of change. The SLWE also showed better scalability than for instance the MLE, which is used by Ludovik *et. al.* in their approach, with regard to a large number of features. Another important motivation for using the SLWE for this task is that there is no need for a separate segmentation process by using complex methods such as dynamic programming which Ludovik and his co-authors use in their approach. Instead, the SLWE is able to adapt to changes quickly if the environment switches its probability vector, which in our case is the distribution of top N-grams for the possible languages being classified.

Every Pattern Recognition problem involves a training and a testing phase. In addition, our approach includes a validation step where we validate the result and measure the classification accuracy. The training phase consists of learning the N-gram language profile for each language, based on a sample training set that is pre-classified for each language. This phase is fairly equal to the approach presented by Cavnar & Trenkle, but instead of only keeping the ranks of the top N-grams for each language, we convert the frequencies to a probability distribution by dividing the frequency of each N-gram by the total number of observed N-grams in the training set. The classification or testing phase of our solution use the SLWE to continuously update the running estimate as new N-grams arrive. A distance measure between the estimated probability distribution and each of the language profiles is used to classify the document according to the shortest distance. One possible distance measure would be the Euclidean distance. Another distance measure could be the cosine similarity measure that we introduced in Ch. 2. Our approach allows us to classify at different levels of granularity, for instance by measuring the distance and classifying after each N-gram, or collecting N-grams and classifying each word separately. By adding sentence boundaries to the process, it is possible to perform classification on sentence level as well. This may be useful if one assumes that sentences generally are monolingual. In our proof-of-concept prototype we collect all the N-grams for each word and classify on the word level. We also implement a trivial sentence boundary algorithm that divides the sample into sentences by looking at punctuation and then use this to classify whole sentences.

6.4.1 The Basic Algorithm

The PR system presented here for classification of language in multilingual documents, consists of three phases. The first phase involves training mix-order N-gram profiles for each language that the system should support. These language profiles are probability distributions consisting of the N-grams for a given language. It is possible to mix several different N-grams in these profiles, but in the interest of brevity, we operate with N-grams of order $N = 1$ to 4. It is possible to experiment with different lengths of each language profile by introducing a cut-off threshold to the profiles. By doing so, only the top N-grams for a given language is kept in the profile. According to Cavnar *et. al.*, satisfying results can be obtained when using values around 300 for this cut-off threshold. The second phase of the PR system consists of the actual classification, or testing phase. In this phase, the estimate of the SLWE is initialized at the beginning of each document, with a feature vector consisting of all unique N-grams from each of the different language profiles. Each document in the testing corpus is processed and for each document, each word is processed and

classified according to a distance measure between the estimated probability vector and each of the language probability distributions. The running estimate of the SLWE is updated after every word is processed. By separating each document into sentences, based on given rules or sentence boundaries, we are able to perform classification on the sentence level. As mentioned previously, this may be useful if one assumes that a sentence is monolingual. The final phase of our PR system consists of validating the classification results.

Training Language Profiles

The first step of our PR system is outlined in Fig. 6.4. The system is given a training set of documents for each language that it should support. The training set consists of monolingual documents, pre-labeled with the language they are written in. This may either be done manually, or one can for instance make assumptions where one take samples from a given discussion board that is assumed to be written in a specific language. This approach may result in errors because of the multilingual nature of online discussions. Still, it is fair to assume that such errors will only reflect a small part of the training set, and can therefore be ignored. Another training set could be made up of news articles collected from different news papers written in different languages.

After the system is given a training set for the given language that it is training its profile for, each document in this training set goes through a tokenization process. Here, the document is broken down into tokens, which in our system is represented as words. We also remove all non alphanumerical characters from the text. After the tokenization process is done, each word in the document is expanded to their mixed order N-gram form. A global counter of all unique N-grams is kept in a hash map, and each time an N-gram is examined, the counter is incremented. After all N-grams are read, the frequencies are converted into probabilities by dividing each frequency by the total number of observed N-grams. By doing so, we get an N-gram probability distribution for the given language.

To reduce the dimensionality of the feature space of our classifier, we introduce a cut-off threshold for the N-gram language profiles. The threshold is applied by keeping the top k N-grams for the profile, and discarding the rest. This will not only reduce the computational complexity of each language profile, but it also affects the running estimate of the SLWE, which consists of all unique N-grams from each of the different language profiles. We are able to use such a cut-off threshold without worrying too much about where the cut-off takes place due to Zipf's law, as mentioned earlier in Section 6.2.1.

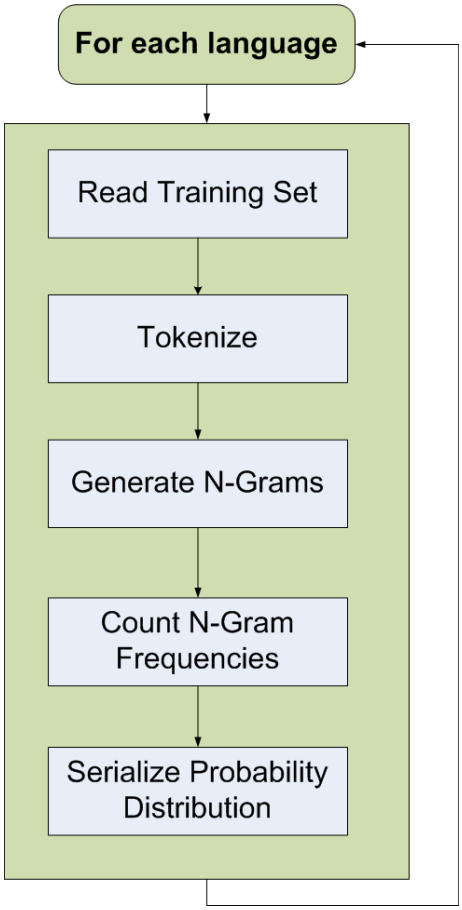

Figure 6.4: Overview of the training phase of the classifier. Learning mixed order N-gram probability distributions for each language.

Testing Classification

The second phase consists of classifying each document in the testing corpus, using the SLWE and the probability distributions for each language. The overview of this step is given in Fig. 6.5. Each document to be classified is read into the system and is put through the same tokenization process as described for the training phase. The feature vector of the SLWE consists of all the unique N-grams from all language profiles defined for the system. The SLWE keeps a running estimate of this feature vector, where each N-gram is associated with a given probability. These probabilities are initialized evenly (e.g., if there are $r = 500$ features in the vector, each N-gram is initialized to probability equal to 0.002.)

After the SLWE is initialized, and the document is tokenized into a list of words, the system is ready to perform the actual classification procedure. The overview of this procedure is depicted

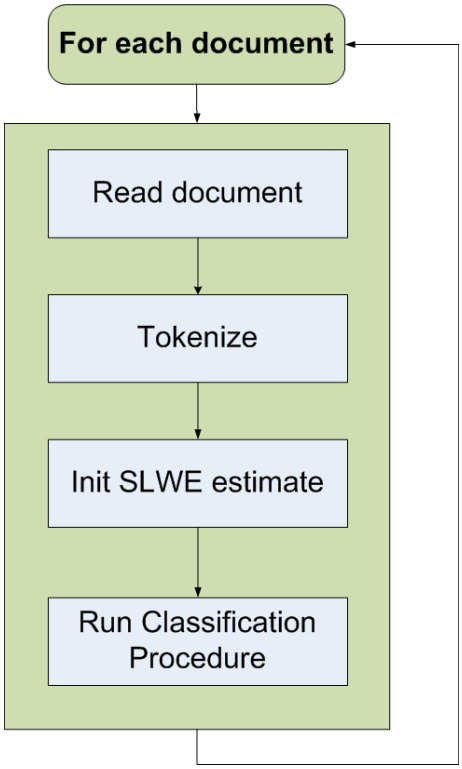

Figure 6.5: Overview of the testing phase of the classifier.

in Fig. 6.6. For each of the words that the sample document contains, the system expands the word into mix-ordered N-grams. Then, for each of these N-grams, the probabilities in the running estimate is updated as per the multinomial updating scheme of the SLWE as described in Ch. 4. If the N-gram is found in the estimate probability vector, its probability is increased according to the updating rules. The probability of all other N-grams are accordingly reduced. If the N-gram does not exist in the estimate vector, it is merely ignored.

After all N-grams for the given word is processed, the system measures a distance between the estimated probability vector and each of the language probability distributions. The word is then classified as written in the language represented by the language profile that measured the shortest distance from the estimate. The distance measure can for instance be the Euclidean distance or the cosine similarity between the vectors.

With the assumption that a sentence is monolingual, we count the number of words in a sentence and classify the sentence as written in the language that has the highest word classification count. One could also assume that individual postings in a discussion are monolingual, and thus making it possible to classify individual postings based on the words or sentences that the postings contain.

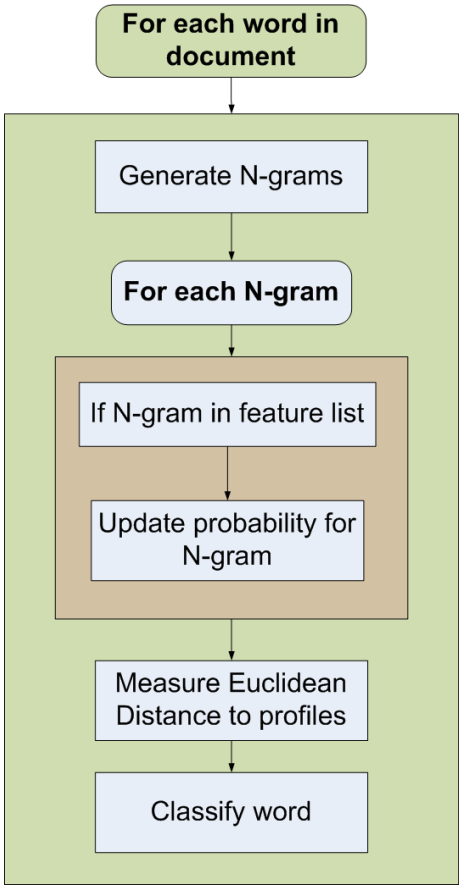

Figure 6.6: Overview of the classification procedure.

Validation of Classification Results

The validation phase of our system is done by comparing the estimated probability vector of the classifier with the correct language profile at every step of the classification. The test documents are generated by our system, by concatenating segments from monolingual documents. This approach makes it possible to pre-label each segment of the multilingual sample document, allowing us to validate the classification results for each segment. By counting the number of correctly classified words or sentences, and comparing this to the validation data, we are able to calculate the classifier accuracy. This approach to validation also allows us to keep track of the misclassified words or sentences and what language they were mistakenly classified as. This can be useful when interpreting the results and may point out differences or similarities between languages.

The validation results are kept in a so-called *confusion matrix*. A confusion matrix for classification with c different classes is a $c \times c$ matrix where element (i, i) corresponds to the percentage

	eng	fre	ger
eng	0.986	0.007	0.007
fre	0.004	0.993	0.003
ger	0.0	0.027	0.973

Table 6.1: A sample confusion matrix used for validation of classification results using three possible languages.

of correctly classified documents for class i. This implies that a classifier with 100% accuracy will yield a confusion matrix equal to the identity matrix, i.e., all elements $(i, i) = 1.0$ for $i = 1, \ldots, c$. Table 6.1 exemplifies a confusion matrix used for validation of a classifier with three possible classes. The rows of the confusion matrix correspond to the true values of the sample, and the columns correspond to the classifier result. Thus, for the example below, if the sample corpus consists of 1000 segments written in English, seven of these were mistakenly classified as French, and seven as German. To compute an overall classifier accuracy one simply sums the diagonal elements and divide by the number of classes. For our example, the overall classifier accuracy is thus 0.984. The classifier error rate is thus $1 - accuracy$ which in this case is 0.016.

Precision and *Recall* are two widely used measures for evaluating the quality of results in statistical classification and Information Retrieval. Precision refers to the exactness and recall is a measure of completeness. The precision of a class in classification is the number of correctly classified documents (true positives) divided by the total number of documents that were classified as belonging to that class, e.g., both correctly (true positives) and incorrectly classified documents for that class (false positives). Recall is defined as the number of correctly classified documents for a given class (true positives), divided by the total number of documents belonging to that class (true positives and false negatives). A precision score of 1.0 for a given class means that every document classified as belonging to this class does indeed belong to the class (but says nothing about the number of items from the class that were not classified correctly). A recall of 1.0 means that every item from the class was classified as belonging to the class (but says nothing about how many other items were incorrectly classified as belonging to the same class). The Precision and Recall measures are defined as outlined in Eqs. (6.1) and (6.2).

$$Precision = \frac{truepositives}{truepositives + falsepositives} \tag{6.1}$$

$$Recall = \frac{truepositives}{truepositives + falsenegatives} \tag{6.2}$$

The *F-measure* is a popular measure that combines Precision and Recall. It was introduced by van Rijsbergen in [62], and is the weighted harmonic mean of precision and recall.

$$\mathrm{F}_\beta = (1+\beta^2) * \frac{\text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}}, \tag{6.3}$$

where β is used to weight one of the two measures. If $\beta = 1$, precision and recall are evenly weighted. Using $\beta = 2$ will give twice as much weight to Precision, and $\beta = 0.5$ gives twice as much weight to Recall.

If we assume that there are 1000 documents for each language in the sample confusion matrix in Table 6.1, the Recall measure for English would be what is reported in the confusion matrix as 0.986. The Precision measure for English is then $986/(986+4) = 0.9959$. The F_1 measure is then approximately 0.9909. In our results we present the classifier accuracy as the Recall measure, but we have shown here that it is fairly straightforward to also calculate the precision measure and the F-measure from the reported confusion matrix.

6.4.2 Discussion of Design Issues

In this section we mention some of the design issues of our approach. One important issue in all PR systems is that of selecting the features space of the classifier. We have chosen to operate with mixed order N-gram language models since they have demonstrated good results for several different text classification tasks that has been studied in the literature. We have mentioned several other possible features, such as character level recognition or word level recognition. More advanced features such as morphological features was also mentioned, but to make our system both robust and maintain the possibility of expanding the system to deal with other classification tasks such as topic classification, we utilized the N-gram language model in our solution. The nature of WoM discussions were also a key motivating factor in choosing N-grams as features, due to their robustness with regard to noise in the input text and that the segments may be too short for word-based features to encapsulate sufficient information. One power that our system has is that the magnitude of the order of N-grams can easily be changed. For instance, using mixed order N-grams for $N = 1$ to 4 can easily be expanded to using mixed order N-grams that range from $N = 1$ to 6.

We chose the Euclidean distance as the distance measure for our classifier, since this is a low computational distance measure for obtaining the distance between two vectors. Cavnar &

Trenkle use an ad-hoc out-of-place measure, but this was not suitable for our approach since we are dealing with a probability distribution and not a ranked list. More sophisticated measures could be evaluated and used to replace the current distance measure. Lin investigated the theoretical basis of similarity and proposed a similarity measure to calculate the similarity between two strings in [33]. A similar measure was proposed by Jiang *et. al.* in [24]. Either of these measures could be suitable for use with our prototype.

6.4.3 Discussion of The Feature Space

The feature space is affected by the choice of N-gram orders to be used, and the number of languages that the classifier should support. As mentioned earlier, we introduce a cut-off threshold for the language profiles to reduce the dimensionality of our feature space. Using a cut-off threshold of 300 and setting the number of supported languages for the classifier to 3 would result in a worst-case feature space of 900 N-grams. However, realistically, most languages have a large number of common N-grams, for instance the uni-grams that compose the alphabet for the given language. From the experiments that we performed with our classifier we observed that using three languages yielded a feature space of 580 unique N-grams, with mixed order N-grams for $N = 1$ to 4 and a cut-off threshold of 300. Moreover, using 8 languages yielded only 957 unique N-grams with the same cut-off and N-gram orders. The worst-case feature space would have been 2400 unique N-grams. We observe that the dimensionality of the feature space of our classifier can be reduced through the use of a cut-off threshold and by reducing the number of mixed-order N-grams. For instance we could use a combination of top bi-grams and uni-grams. For word level classification it is common to reduce the feature space by for instance using only adjectives, or a combination of verbs and nouns.

6.5 Prototype

We implemented a prototype of our proposed PR system in Java, using training and testing data collected from online Word of Mouth discussions from various discussion boards. The prototype is a proof-of-concept implementation of our method and the prototype follows the same three phases as described for our proposed algorithm. In the following sections we will describe these phases, followed by the results from our testing.

6.5.1 Limitations

For our prototype we chose to operate with N-grams of order $N = 1$ to 4 and using cut-off threshold set to either 400 or 500 N-grams per language profile. Since our prototype is a proof-of-concept implementation of our proposed method, we chose to limit the classifier to supporting a maximum of 8 different languages. This can easily be expanded by providing training data for other languages and building the language profiles from these training sets as well.

The sentence boundary used by our prototype is rather trivial and the classifier could benefit from having a more sophisticated algorithm for detecting sentence boundaries. We also believe that using a more sophisticated distance measure could yield improved accuracy.

6.5.2 Training

The training phase of our prototype consists of reading monolingual documents for a given language and then build the N-gram probability distribution for this language. For our training we use documents gathered from online discussion boards. We make the assumption that discussions obtained from an English discussion board are monolingual, written in English and discussions obtained from a Spanish discussion board are generally written in Spanish. This assumption is not completely true, since users of these discussion boards can post in any language they want, and postings can contain parts which are for instance cited from sources written in a different language than the majority of postings found on the board. Despite this, we assume that the majority of postings are representative for the given language, and thus regard the possible postings that might be written in a different language as noise. This is obviously not optimal when training our language profiles, but the task of manually labeling each discussion for all languages is infeasible. This is something that could be improved in future work, by for instance using pre-labeled documents. It would also be possible to test the classifier on the training corpus and then manually inspect the documents that are misclassified, to see if they represent noise in our training data. These documents can then be removed from the training corpus, and the classifier can train its language profiles on the cleaned training corpus. The training corpus we used for our classifier was retrieved from general discussion boards, implying that the topics of the different discussions vary and does not bias the training data. For instance, using discussions from a discussion board dealing with car-related talk might bias the language profiles with terminology relevant to the topic of the discussions. This is undesirable when training our classifier.

Dutch	English	French	German
Italian	Norwegian	Spanish	Swedish

Table 6.2: Languages that we trained N-gram probability profiles for with our prototype.

We built language profiles for the languages outlined in table 6.2. The training of each language profile was done on a training set of approximately 600 monolingual samples for each language[3]. Each sample had an average length of $15 - 20$ kB.

We chose to operate with N-grams with order $N = 1$ to 4 for our prototype. To be able to experiment with different cut-off thresholds, we keep all N-grams in the training phase and serialize the whole N-gram profile for the given language. The cut-off is then done in the testing phase when the language profiles are loaded into the system. After the cut-off has been applied, we normalize the probabilities for each language profile so that they sum to unity.

In our prototype we use a slightly different approach to generating the mixed order N-grams than the one described in Section 6.2.1. Instead of padding each word with multiple trailing whitespaces, we pad each word with one leading and one trailing whitespace. This way, the information regarding beginning and ending of each word is kept intact, and we limit the feature space of our estimator. Essentially, the three N-grams "e_", "e__" and "e___", all contain the same information. Through our approach these are all represented as a single N-gram; "e_". Cavnar & Trenkles approach yields $k + 1$ N-grams per rank, for a string of length *k*, padded with blanks. Our approach results in $(k + 3) - n$ N-grams per rank when $(k + 2) >= n$. Thus a word may not generate N-grams for ranks that result in $(k + 2) < n$.

Tables 6.3 and 6.4 shows some of the top N-grams for the English and French language profiles generated by our prototype, based on our training data. For the English language profile we observe that words beginning with the bi-gram "_t" or "_i" occurs frequently, and the bi-gram "th" also ranks high. For the French language we observe that bi-grams like "_p" and "_d" are commonly used prefixes. "es" and "le" are some of the other high-ranking bi-grams found in this profile.

6.5.3 Testing

To test our classifier we constructed our testing-sets by concatenating sentences from monolingual documents. This way, we were able to pre-label each word of the testing document, making it

[3]The training and testing data used for our experiments can be made available upon request.

1.	'e'	0.02898303432788194
2.	't'	0.023419623676371827
3.	'o'	0.02072723947005473
4.	'a'	0.019712329919370667
5.	'i'	0.017598997574043738
...		
13.	'_t'	0.00879038643066665
14.	'u'	0.007777279324096808
15.	't_'	0.007732943471347147
16.	'th'	0.006833265908572024
...		
24.	'_i'	0.005724762778327412
25.	'_th'	0.005538376705479297
...		

Table 6.3: A selection of top N-grams for the English language profile, based on our training data. The complete profile contained approx. 41100 unique N-grams.

1.	'e'	0.03585800037795376
2.	'a'	0.019563550458459363
3.	's'	0.01895590454817659
4.	'i'	0.01717803472023986
5.	't'	0.01667467067786116
...		
18.	'_p'	0.005852026387121135
19.	'_d'	0.00521445611474415
20.	'n_'	0.004755615895966232
...		
23.	'_c'	0.0044992638606924824
24.	'es'	0.004424180915801551
25.	'le'	0.0043444732967687855
...		

Table 6.4: A selection of top N-grams for the French language profile, based on our training data. The complete profile contained approx. 41500 unique N-grams.

easier to validate our test results. When constructing the multilingual documents we experimented with different sentence lengths, i.e., the number of words in each sentence. For each document we concatenated 5 sentences for each language, and each multilingual document contained three segments of 5 sentences, where the language of each of these segments were randomly drawn from the list of possible languages for our classifier. By constructing the documents in this fashion, a test document could consist of three different languages, or of two different languages, were the

first and last segment is written in one language, and the second is written in some other language. To be able to perform sentence level classification, we simply add a period at the end of every sample sentence. This is a rather trivial way of determining sentence boundaries, but for our proof-of-concept implementation, this is sufficient. A sample multilingual document constructed by our system is outlined below in Listing 6.1.

Listing 6.1: Sample Multilingual document constructed by our system.

```
ist schon unschlagbar auf dem handysektor bei tageslicht sind die.
zu rettenden briefbeschwerer das kann push mail und hat einen.
vorgesehen und bei se gibt es weiterhin die moeglichkeit auch.

pour l'avoir eu en main il est tres agrable au.
j'ai d j  achet un carte sd et oui l'adaptateur est.
de configuration moi aussi alors qu est ce que ca.

you all next time yes lulu it was steep and.
my complaint about losing my recordings yet not sure if.
had bad news for me i had no idea at .
```

We observe that the sentences may not be "real", or grammatically correct sentences. The reason for this is that each individual sentence is constructed by concatenating m consecutive words from one of the monolingual documents for the given document. This may thus result in a sentence made up of partial sentences from the monolingual document, or a an overlapping segment from two different "real" sentences.

The first part of the testing phase is done by loading the different N-gram language profiles and the multilingual testing corpus. The cut-off threshold is applied to the language profiles, keeping only the top N-grams for the given language. The profile is normalized so that the probabilities sum to unity. The SLWE is then initialized with the feature space consisting of all unique N-grams from all the language profiles, after cut-off has been done. The probabilities are initialized evenly, so that the probability of each N-gram in the running estimate is $1/r$, where r is the number of N-grams in the estimate probability vector. The SLWE is re-initialized for each sample document to be classified.

Then, for each sample document, the document is first separated into sentences by a simple sentence boundary. Each sentence is tokenized into individual words, which are read by the classifier, one word at a time. The classifier process one sentence at a time, keeping track of the

number of words that are being classified as a given language. The sentence is then classified as written in the language that has the highest word classification count. Word classification is done by expanding the word into mixed order N-grams for $N = 1$ to 4. The SLWE updates the N-gram probability in the running estimate as outlined in the pseudo code in Listing 6.2.

Listing 6.2: Pseudo code for the updating of the running estimate of the SLWE

```
sum = 0.0;
FOR EACH element IN runningEstimate DO
    IF sample != element DO
                //penalize the elements that are
                //not equal to the incoming sample
        runningEstimate[element] = runningEstimate[element] * lambda;
        sum += runningEstimate[element]
    ENDIF
ENDFOR
//reward the element represented by the incoming sample
runningEstimate[sample] = (1.0 - sum);
```

After all the N-grams for a word is processed and the probabilities are updated, the Euclidean distance from the running estimate to each of the language profiles are measured, and the word is classified as written in the language of the profile that measured the shortest distance. We chose to use the Euclidean distance as our distance measure.

6.5.4 Validation

Our prototype make use of the confusion matrix approach, introduced earlier, for validation purposes. The updating of the confusion matrix after every sentence is classified, is done by incrementing the counter for the corresponding element in the confusion matrix. The rows of the matrix corresponds to the validation information and the columns correspond to the output from the classifier. After the classification and validation counts are done, the accuracy is calculated by dividing the classification count by the number of sentences in the sample. Averaged classifier accuracy is calculated by summing the diagonal elements of the matrix and dividing by the number of classes for the classifier. In our results, we report the maximum and minimum accuracy for the classifier in addition to the averaged classifier accuracy.

1. English, French and German
2. English, French, German, Norwegian and Italian
3. English, French, German, Norwegian, Italian, Spanish, Dutch and Swedish

Table 6.5: The languages used in our testing sets.

6.6 Language Classification Results

In this section we present the results from our empirical testing of the classifier. In the different training sets, we experiment with different languages and also try different segment lengths with regard to the number of words per sentence. We also use different learning parameters for the SLWE and vary the cut-off threshold used on the language profiles to investigate the impact this has on the classifier.

The motivation for these experiments is to investigate how well our algorithm is suited for language classification in multilingual documents. By testing several different languages we seek to investigate the ability to classify documents written in different languages and how well the classifier scales with regard to the number of supported languages. We use different values for the cut-off threshold to examine how well the classifier scales with regard to the number of features, and we experiment with different values for the learning parameter of the SLWE to evaluate the impact of slow versus fast convergence when dealing with language classification. We measure the accuracy of our classifier operating with different sentence lengths to see how well it is able to deal with short or long sentences. In the next section, we describe the experimental setup, explaining the different test sets and the use of different parameters. In Section 6.6.2 we present the results from these experiments, reporting the confusion matrix and the averaged classifier accuracy.

6.6.1 Experimental Setup

The classifier was tested on three different sets of languages, generated as described in Section 6.5.3. The languages used for our testing is outlined in Table 6.5.

For all of our test sets we generated documents containing three different segments with five sentences per segment. For each of these sets we generated different variants using different sentence lengths. The parameters used for generating these testing sets are outlined in Table 6.6.

The first language set was divided into three test sets. Test set I contained 495 sentences written in English, 495 sentences in French and 510 sentences written in German. Test set II consists of

Test set	Language set	Words per sentence	Corpus size
I.	1	10	100
II.	1	15	100
III.	1	20	100
IV.	2	10	100
V.	2	20	100
VI.	3	20	200

Table 6.6: The parameters used for generating our multilingual testing corpus.

Language	Test set IV.	Test set V.
English	245	300
French	310	275
German	325	310
Norwegian	335	315
Italian	285	300

Table 6.7: Sentence distribution for the second language set

515 English sentences, 515 sentences written in French and 470 German sentences. Test set III contained 520 English, 515 French and 465 German sentences. The sentence distribution for the second language set is outlined in Table 6.7, consisting of two test sets.

The distribution of sentences per language for the last language set, represented as test set VI, containing 8 different languages, is outlined in Table 6.8.

English	French	German	Norwegian	Italian	Spanish	Dutch	Swedish
450	400	320	405	360	325	405	335

Table 6.8: Sentence distribution for the third language set

With these six test sets we tested our classifier on four different test cases, using different values for the learning parameter, λ, and different cut-off thresholds. The values used for our experiments are outlined in Table 6.9.

As stated earlier, we chose to operate on sentence level classification for our prototype. Essentially, the sentence classification was done through word classification on all the words in a given sentence. Figure 6.7 shows a plot of the Euclidean distance between the estimate and three possible language profiles for a document that is monolingual. Despite the document being monolingual, the system assumes that the document is multilingual. The sample being classified contains 300 words written in French and in this example, the classifier operates on word-level, disregarding any sentence boundaries. We observe that the SLWE converges rapidly to the true

Test case	Learning parameter	Cut-off
A	0.98	400
B	0.99	400
C	0.98	500
D	0.99	500

Table 6.9: The parameters used for the four test cases that we ran on all test sets in our experiments.

language profile, which for this sample was French. Even though the variance of the estimate is rather high, we observe that the distance to the other language profiles is far greater than the distance to correct language profile. We used $\lambda = 0.99$ and 300 as the cut-off threshold in this experiment.

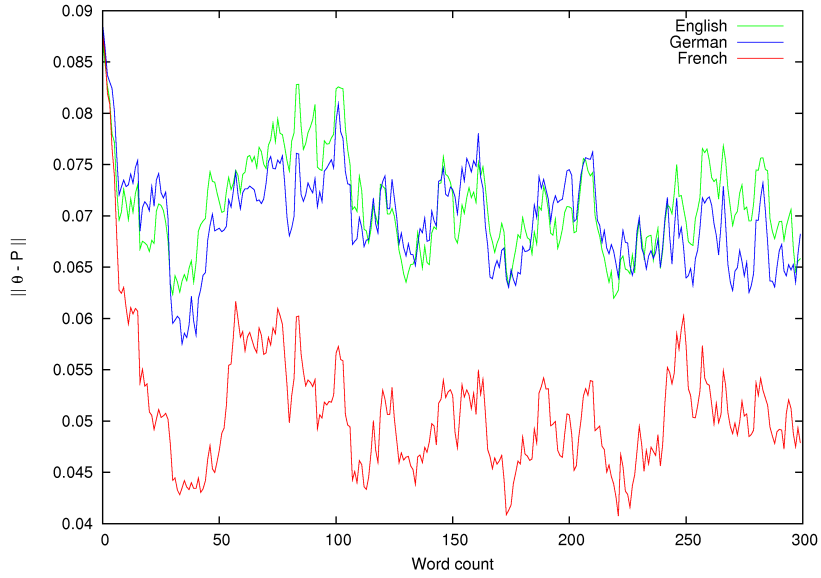

Figure 6.7: Plot of the Euclidean distance from the estimated probability vector to each of the language profiles. The document being classified was monolingual, written in French, containing 300 words.

6.6.2 Results

In this section we present our experimental results for each of the test cases described above. For each of the three language sets we report the results from the different test cases mentioned in the previous section. The classifier accuracy in each case is reported and some of the confusion matrices are included as well.

Language Set 1

The first language set consists of three languages; English, French and German. The classifier was tested on three different test sets with different sentence lengths for these languages, as described previously. Each of the test sets contained 100 sample multilingual documents. The classification accuracy is reported for each of the test cases in Table 6.10.

Test set	Test case	λ	Cut-off	Accuracy (eng)	Accuracy (fre)	Accuracy (ger)
I.	A	0.98	400	0.968	0.962	0.949
I.	B	0.99	400	0.941	0.891	0.920
I.	C	0.98	500	0.970	0.960	0.949
I.	D	0.99	500	0.945	0.905	0.925
II.	A	0.98	400	0.973	0.990	0.987
II.	B	0.99	400	0.951	0.963	0.966
II.	C	0.98	500	0.971	0.992	0.987
II.	D	0.99	500	0.961	0.965	0.974
III.	A	0.98	400	0.996	0.990	0.983
III.	B	0.99	400	0,987	0.986	0.974
III.	C	0.98	500	0.994	0.990	0.983
III.	D	0.99	500	0.988	0.986	0.974

Table 6.10: Reported classifier accuracy for each of our test cases for the first language set.

We observe that best accuracy for all the test sets is achieved with the learning parameter λ set to 0.98. We know from Ch. 5 that higher values of λ yields slower, but more accurate convergence. When classifying short sentences, it is important that the SLWE is able to converge rather quickly so that as few words as possible in the sentences are misclassified. We also observe that the different cut-off thresholds only to a small extend affects the classifier accuracy.

Table 6.11 shows the confusion matrix for test case A on test set III, which demonstrated an averaged classifier accuracy of 0.9896. In this experiment, the test set consisted of 520 sentences in English, 515 sentences in French and 465 sentences in German. Each sentence consists of 20

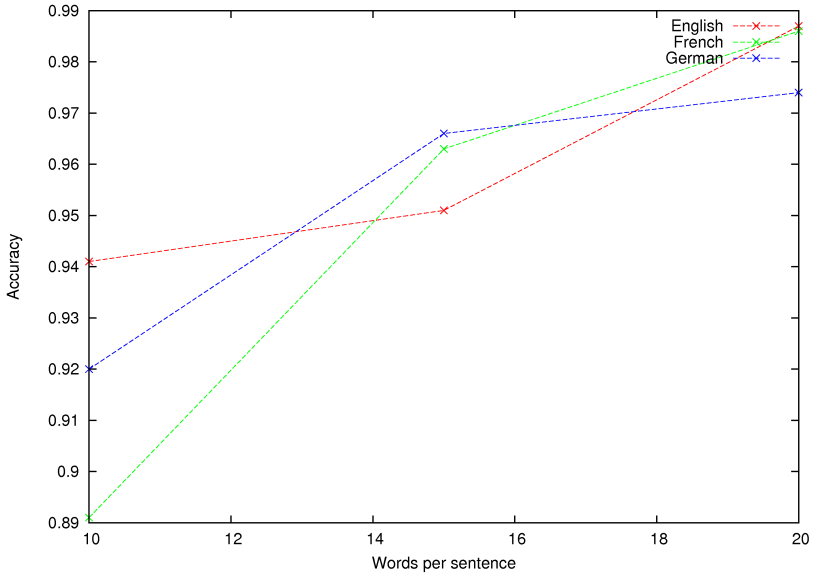

Figure 6.8: Plot of the classifier accuracy for the three languages in language set 1, using different sentence lengths with $\lambda = 0.99$ and cut-off set to 400.

words. By looking at the accuracies listed in Table 6.11, we observe that only two of the 520 sentences in English were misclassified. One of these as French and the other as German. Five of the 515 French sentences were mistakenly classified as English, and only 8 of the 465 German sentences were misclassified.

	eng	fre	ger
eng	0.996	0.002	0.002
fre	0.010	0.990	0.000
ger	0.013	0.004	0.983

Table 6.11: Confusion matrix for test case A, using test set III.

Language Set 2

For the second language set we experimented with five different languages, namely English, French, German, Italian and Norwegian. Two different test sets was used with 10 and 20 words per sentence, respectively. Each test set contained 100 sample documents.

The averaged classification accuracy for all languages from these experiments are outlined in Table 6.12.

Test set	Test case	λ	Cut-off	Averaged accuracy	Best accuracy	Worst accuracy)
IV.	A	0.98	400	0.9296	0.955 (nor)	0.895 (ita)
IV.	B	0.99	400	0.9030	0.925 (nor)	0.874 (ita)
IV.	C	0.98	500	0.9282	0.955 (nor)	0.844 (ita)
IV.	D	0.99	500	0.9070	0.929 (fre)	0.877 (ita)
V.	A	0.98	400	0.9848	0.997 (ger)	0.968 (nor)
V.	B	0.99	400	0.9814	0.997 (ger)	0.962 (nor)
V.	C	0.98	500	0.9856	0.997 (ger)	0.968 (nor)
V.	D	0.99	500	0.9814	0.997 (ger)	0.962 (nor)

Table 6.12: Reported classifier accuracy for each of our test cases for the second language set with five different languages.

Table 6.13 shows the confusion matrix for test case C on test set V, where the sentence length was 20 words. We observe that a large portion of the misclassified Norwegian sentences was classified as written in English.

	nor	eng	fre	ger	ita
nor	0.968	0.022	0.006	0.000	0.003
eng	0.000	0.987	0.003	0.010	0.000
fre	0.000	0.000	0.989	0.004	0.007
ger	0.000	0.000	0.003	0.997	0.000
ita	0.003	0.000	0.010	0.000	0.987

Table 6.13: Confusion matrix for test case C, using test set V.

We also performed one experiment on this language set using only uni- and bi-grams, using $\lambda = 0.98$ and the cut-off threshold set to 500. We used test set V, and this yielded an averaged classifier accuracy of 0.9748. This is only 0.0108 less accuracy than our experiment when using a combination of uni-grams, bi-grams, tri-grams and quad-grams.

Language Set 3

For the last language set we tested our classifier using all eight languages that we had generated language profiles for. The languages were English, French, German, Italian, Norwegian, Spanish, Swedish and Dutch. For this case we generated the test samples using a sentence length of 20 words. This testing corpus consisted of 200 documents.

Test set	Test case	λ	Cut-off	Averaged accuracy	Best accuracy	Worst accuracy)
VI.	A	0.98	400	0.9695	0.988 (fre)	0.928 (nor)
VI.	B	0.99	400	0.9701	0.986 (ita)	0.928 (nor)
VI.	C	0.98	500	0.9690	0.988 (fre)	0.916 (nor)
VI.	D	0.99	500	0.9717	0.986 (ita)	0.931 (nor)

Table 6.14: Reported classifier accuracy for each of our test cases for the third language set with eight different languages.

6.7 Discussion and Summary of Results

We have observed that our classifier is able to classify multi-lingual documents with high overall accuracy. Our experiments demonstrates that the classifier performs extremely well for moderate-sized segments (sentences of length 15 - 20 words), and that it performs adequately for shorter sentences with 10 words per sentence.

For the first language set, we obtained a classification accuracy for the English language as high as 0.996 using $\lambda = 0.98$ and the cut-off threshold set to 400. This accuracy was achieved with sentences consisting of 20 words. For shorter segments, with 10 words per sentence, we achieved an accuracy of 0.97. The worst-case accuracy for classification of the English sentences was 0.941 when using $\lambda = 0.99$ for the sentences with only 10 words per sentence. This is still a fairly good accuracy considering the length of the segments. We also know that using a high value for the learning parameter yields slower convergence, and this might be the reason for the errors in classification observed here. Since the changes are frequent and convergence is slower, the SLWE lags behind when a change takes place in the sample. By lowering the learning parameter to $\lambda = 0.98$, classification accuracy was increased to 0.968 for the same test set. The accuracy was further improved by increasing the feature space to using a cut-off of 500 N-grams per language profile, resulting in an accuracy of 0.970 for the English segments. The accuracies for the French and German segments in the first language set was slightly lower than for the English segments, with 0.99 for French sentences with 20 words, and 0.983 for German sentences with 20 words, compared to 0.996 for the English classification.

We observe that using a cut-off threshold around 400 yields satisfying results, which is in accordance to the suggested cut-off thresholds used by Cavnar & Trenkle in their experiments. This also shows us that by reducing or increasing the feature space, the classifier scales well and is not notably handicapped by working with a limited feature set compared to a larger one.

In the second language set we introduced segments written in Italian and Norwegian in ad-

dition to the three previous languages. The results from this language set was quite astonishing since the best reported accuracy for sentences containing 20 words was obtained for the German classification, with an accuracy of 0.997. It was encouraging to see that despite introducing more languages to the classifier, it still performs well, with an averaged classification accuracy of 0.9856 for sentences with 20 words, and 0.9296 for sentences with 10 words. We observe that the Italian segments yielded the lowest accuracy for short segments, but that the accuracy was severely improved when the sentences were longer. With segments of 20 words per sentence, all languages yielded accuracies over 0.968, which is a very motivating result.

We observe that most of the misclassified Norwegian sentences were classified as written in English. By analyzing some of the wrongly classified documents, we observed that proper names and words in other languages could lead to misclassification. There were also examples that showed how noise could affect the test samples. For instance, in one of the documents we found a segment labeled as Norwegian, containing one sentence that was actually written in English. This occurred because the English phrase was used as a quote in a Norwegian discussion. The classifier then reports this as a misclassification, but in reality the classification was correct, since the phrase was actually written in English. Obviously, when classifying discussions like this, we want the classifier to classify this sentence as written in English. The example segment is shown below, where we observe that the second sentence is written in English.

> *"noe som også er bekreftet av rockstar med tanke på hvor mye wii nunchuk kontrollerne løftet et relativt middelmådig spill."*
>
> *"targeting is handled on the left trigger and it automatically locks on to the closest enemy in sight if you."*
>
> *"mellom norges to største byer men vi er blitt flinke til å snuse opp billige biletter og er tidlig ute."*

Another example supporting these observations is the sentence *"jo glad i å synge han har jo bla en sang med gruppa si nine inch nails just the way"*, which contains both Norwegian and English words in the same sentence. In these cases, it is difficult to make a decision for the classifier. In such cases, one must decide whether or not a segment in a Norwegian discussion, containing mainly English words, should be classified as Norwegian or not. These observations can explain some of the classification errors, however not all errors can be manually assessed.

It would be of interest to investigate different approaches to smoothing the data to achieve better discrimination of the features in different languages. Chen & Goodman has studied different

smoothing techniques in [7], which could be subject to further study. Central to many of these smoothing techniques is the *Good-Turing* estimate, which states that for any N-gram that occurs r times, one should pretend that it occurs r^* times, where $r^* = (r+1)\frac{n_r+1}{n_r}$ and n_r is the number of N-grams that occur exactly r times in the training data.

For the last language set, using eight different languages, we observed through our experiments that our classifier is able to scale well with regard to the number of supported languages. The averaged accuracy reported for our experiments was slightly lower than for the case when dealing with only five languages, but the classifier still performs well with an error rate of only 0.0283 for eight languages, compared to an error rate of 0.0186 in the case of five languages. The most notable observation is that the accuracy of the Norwegian classification is the worst reported accuracy for the last language set. For the second language set, using short sentences of only 10 words, the Norwegian classification accuracy yielded the best result. Our experiments has showed that the more similar two languages are, the harder it was to discriminate them statistically. Even humans can find this task difficult, so the problem will always be hard to solve using only the textual information. We have observed that Italian, Spanish and to some extend French segments are difficult to discriminate, and we observe the same from our experiments with Norwegian and Swedish segments. One possible approach to improve the classification rate of languages that resemble each other is to introduce weighting to certain N-grams that may be unique for the given language. An example of this could be the use of characters like 'æ' and 'å', which are unique to the Norwegian and Danish alphabet. We have observed that segments written in Norwegian or Swedish are difficult to discriminate, and by giving more weight to such unique N-grams, the accuracy of the classification may increase and could be subject to further investigation.

The experiments we performed also supports our assumptions regarding the value of the learning parameter, λ, and how this affects the rate of convergence. In our simulations on synthetic data, presented in Ch. 5, we demonstrated that lower values for the learning parameter yielded fast and fairly accurate results for the binomial case, and we also observed that using a higher value for the learning parameter resulted in better performance in the multinomial case when the number of possible outcomes of the estimate was increased. The number of possible outcomes corresponds to the feature space of our classifier, and we observe in the last test case, when the number of supported languages is increased to eight, that the best accuracy is achieved when using $\lambda = 0.99$. We also observe that best performance is achieved using $\lambda = 0.98$ for the first two language sets, where the dimensionality of the features is smaller. In the last language set, using a cut-off threshold of 500, the feature space of the classifier consisted of 1714 unique N-grams. For comparison, the first language set with three languages, using the same cut-off threshold, resulted in a feature

space of 985 unique N-grams.

Our empirical testing was done on the sentence level, but the sentence classification is based on the classification of individual words in the sentence. We observed that using a large value for the learning parameter resulted in slower convergence, yielding misclassified words after a change in language occurred. However, when the sentence is of moderate length, this is not notable when dealing with classification on the sentence level since the larger part of the sentence is correctly classified. By lowering the value of the learning parameter, the number of misclassified words after a change of language has occurred is reduced, but the classifier is subject to less accuracy throughout the sentence. This means that a word in the middle of the sentence might be misclassified if it has similar features in several language profiles. Tuning the classifier to find optimal values for the learning parameter is subject to further research and may improve the overall classifier accuracy.

We have also observed that our proposed method is suitable for classifying texts containing spelling or grammatical errors. For instance, words like 'gettiing' or 'yestrday' are classified as English by our method, despite them being mis-spelled.

6.7.1 Comparison of Performance Against Other Techniques

As mentioned in the introduction to this chapter, our prototype implementation was not compared to other techniques. The most promising of the other approaches in the literature on language classification in multilingual documents is the approach by Ludovik *et. al.*, but since their test data is not publicly available it would be difficult to compare our results to their reported accuracies. Also, they do not report change point location, but merely count the number of correctly classified bytes in the document. Moreover, they do not report the errors per language, only an overall performance measure. Cavnar & Trenkles approach to language classification and their reported results are very good, but operate only on monolingual documents. We mentioned some other approaches to multilingual classification previously in this chapter, but most of these did not show as good results as Ludovik *et. al.* It would be interesting to further investigate Martins & Silvas approach were they make use of the similarity measure proposed by Lin in [33]. They demonstrated improved accuracy over the approach by Cavnar & Trenkle on monolingual documents by using the same N-gram based language models, but with different similarity measures.

In the next chapter we conclude our work and propose some ideas for further research.

Chapter 7

Conclusion and Further Work

In this thesis we have investigated estimation algorithms dealing with non-stationary environments, and in particular we have studied the application domain of PR-systems where the class-conditional distributions of the features are unknown and non-stationary.

We have studied both traditional estimation methods and a new learning-based approach to estimation, presented by Oommen *et. al.* in [46]. The method presented by Oommen *et. al.* is a novel estimation method, based on the principles of stochastic learning, referred to as the Stochastic Learning Weak Estimator (SLWE). The main and fundamental difference between the SLWE and the other methods is that the updating scheme used by the SLWE is multiplicative and not additive.

In this thesis, the SLWE has been formally presented, rigorously analyzed, and empirically tested. We have examined the convergence properties of the SLWE with regard to efficiency and accuracy, and in particular we have validated and proved the theorems presented by Oommen and his co-authors in their paper. Moreover, we have investigated these properties with regard to both binomial and multinomial distributions. The estimator converges weakly, and we have demonstrated that the convergence is independent of the value of the learning coefficient, λ. Furthermore, both the rate of convergence and the variance of the estimate is determined by this learning coefficient. We have demonstrated that the variance of the final distribution and the rate of convergence decrease with λ.

Through extensive simulations we have assessed the efficiency and accuracy of the SLWE and compared the results to the traditional windowed-based MLE approach (MLEW). The simulations involved both binomial and multinomial environments on synthetic data. These results

demonstrated the superiority of the SLWE over the MLEW, both with regard to its ability to adapt to changes in the environment and with regard to its scalability when the feature space grows large. In all of our simulations, the SLWE yielded higher overall accuracy than the MLEW. We believe that our results provide additional insight into the performance of the SLWE and the MLEW when operating in non-stationary environments.

Moreover, we have presented a robust and computationally efficient approach to the application domain that deals with language classification in so-called multilingual Word-of-Mouth discussions, using N-gram characteristics as our discriminatory features in combination with the SLWE. Our proposed solution to language classification in multilingual documents involves a training and a testing phase, where the training is achieved by learning the N-gram characteristics of every language. The testing however, utilize the SLWEs ability to adapt to changes that may occur in the environment.

We observe that it is sub-optimal to work with strong estimators in application domains involving non-stationary data. If the estimator used is strong (i.e., converges w.p. 1), it is difficult for the learned distribution to emerge from a distribution that it has converged to. Our approach has a clear advantage over other approaches in the sense that it is able to maintain a running estimate, and classify each incoming sample continously, without having to perform any pre-processing such as segmentation before the actual classification can take place.

We evaluated the performance of our proposed method through experiments with a prototype implementation, tested on multilingual documents, constructed from monolingual segments in different languages. We have demonstrated with different language sets and different segment lengths that our method is indeed able to classify multilingual documents with high overall accuracy. Not only is the classifier able to detect the languages in the documents, but also where the changes take place.

The computational complexity of our method is a motivating factor in comparison to for instance sliding window schemes, where all the observations in the sliding window must be retained and updated during the entire process. Using non-linguistic features also reduce the computational complexity compared to methods using for instance morphological features, or preprocessing such as stemming or spell checking.

Our experiments has demonstrated that our proposed classifier performs extremely well for moderate-sized segments, with a reported overall classifier accuracy of 0.989, for the case when classifying three different languages with a sentence length of 20 words. The experiments also

indicate that the classifier performs adequately for shorter sentences with 10 words per sentence, yielding an overall classifier accuracy of 0.9596. We also point out that the recall measure for individual languages yielded an accuracy as high as 0.994 for English sentences and 0.99 for French sentences, in the first language set.

We have also demonstrated how well our approach scales with regard to the number of features, and that the classifier was not notably handicapped by working with a limited feature set compared to a larger one. The experiments also showed that our approach scales well when adding support for more languages, yielding an error rate of only 0.0283 when dealing with eight languages, compared to 0.0186 when dealing with five languages.

The results obtained from our testing has shown that our approach is indeed capable of classifying multilingual documents. The concept of using N-gram based models for language classification has been widely studied and has demonstrated good results in monolingual classification. It is satisfying to observe through our experiments that this model is also applicable to multilingual language classification in combination with the SLWE used in our approach. Moreover we have demonstrated the capability of classifying sentences of different lengths with high accuracy. Since the samples used in our testing was constructed using segments obtained from online discussion boards, we also demonstrated the power of our approach with regard to dealing with texts that are subject to both spelling and grammatical errors. Using non-linguistic features has clearly shown its advantage with regard to the presence of such errors. The empirical results for our approach clearly shows that the SLWE is suitable for classification tasks were the samples being classified exhibits a non-stationary behavior, and it is our opinion that our proposed technique will be of benefit in applications dealing with PR in multilingual text documents.

The state-of-the-art on classification of multilingual documents are either computationally expensive with regard to morphological features or prior segmentation of the documents. In addition, most of these approaches are dealing with documents containing a substantial number of words. The approach presented in this thesis differs from these approaches in the way that the classification is done without prior segmentation, and by using low-computational, non-linguistic, discriminatory features. The proposed method has shown that it is able to deal with classification of short segments and that it is resistant to the presence of grammatical or spelling errors. Furthermore, we believe that our approach forms a solid foundation for a new way of classifying textual data that exhibits a non-stationary behavior. We also believe that our approach can be adapted to similar classification tasks, such as topic classification and sentiment classification in documents containing more than one topic or opinions of several authors.

7.1 Further Work

The method presented in this thesis has demonstrated its ability to classify multilingual documents with high accuracy. Although our approach has already demonstrated good performance on real-life experiments, there is still room for further improvements. We also believe that our approach can be utilized in other classification tasks where the class-conditional distribution is unknown and non-stationary. Quite a few possible directions for future research present themselves.

Finding Optimal Learning Parameters

Since both the rate of convergence and the variance is dependent on the learning coefficient, λ, tuning the classifier to find optimal values for the learning parameter for our problem area is subject to further research and may improve the overall classifier accuracy. One approach to this problem is currently being pursued, based on the principles of Genetic Algorithms. Some initial results in this direction are available.

Improved Sentence Boundary Detection

To be able to classify real-life WoM discussions, our prototype needs to be enhanced with an improved sentence boundary algorithm. This problem is still unexplored.

Improved Discriminatory Features

We have seen that the more similar two languages are, the harder it was to discriminate them statistically. Introducing weighting of features and experimenting with smoothing techniques is still an open problem. We believe that this may improve the discrimination of similar languages in our approach. Improved training data and adding support for more languages are also important improvements that could be done to our prototype. Also, better tuning of the language profiles with regard to using optimal mixed order N-grams and cut-off threshold could raise the performance of our classifier.

Investigate Other Similarity Measures

Martins & Silva demonstrated improved accuracy over Cavnar & Trenkles approach by introducing more sophisticated similarity measures in classification of monolingual documents. Using the similarity measure proposed by Lin, their experiments showed that it on average outperformed the original "rank-order" measure proposed by Cavnar & Trenkle by 32%. These results are motivating and gives us reason to believe that introducing one of the proposed similarity measures used in their approach may further improve the classifier accuracy of our approach. These problems are still open, and are currently being studied.

Other Application Areas

The work presented in this thesis forms a good foundation for examining similar classification tasks such as topic classification in documents with multiple topics, or sentiment analysis in documents where authors express their opinions. We would also like to compare our method to other approaches, using a standardized multilingual testing set. These problems are still unexplored.

Bibliography

[1] *Webster's Dictionary for Students: Special Encyclopedic Edition*. Bt Bound, 2003.

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] P. Bickel and K. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics*, 2nd ed. Prentice Hall, 2000, vol. I.

[4] J. H. J. H. C. Souter, G. Churcher and S. Johnson, "Natural language identification using corpus-based models," *Hermes Journal of Linguistics*, vol. 13, pp. 183–203, 1994.

[5] G. Casella and R. Berger, *Statistical Inference*, 2nd ed. Brooks/Cole Pub. Co., 2001.

[6] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," in *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US, 1994, pp. 161–175. [Online]. Available: citeseer.ist.psu.edu/68861.html

[7] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, A. Joshi and M. Palmer, Eds. San Francisco: Morgan Kaufmann Publishers, 1996, pp. 310–318.

[8] R. Cole, J. Inouye, Y. Muthusamy, and M. Gopalakrishnan, "Language identification with neural networks: a feasibility study," *Communications, Computers and Signal Processing, 1989. Conference Proceeding., IEEE Pacific Rim Conference on*, pp. 525–529, 1-2 Jun 1989.

[9] H. Cramér, *Mathematical Methods of Statistics*. Princeton, NJ: Princeton University Press, 1946.

[10] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," 2002.

[11] M. Creutz, "Unsupervised segmentation of words using prior distributions of morph length and frequency," in *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 280–287.

[12] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York, NY: John Wiley and Sons, Inc., 2000.

[13] V. J. Easton and J. H. McColl. Statistics glossary. STEPS. [Online]. Available: http://www.stats.gla.ac.uk/steps/glossary/

[14] J. B. Elsner and A. A. Tsonis, *Singular Spectrum Analysis*. Plenum Press, 1996.

[15] Word-of-mouth marketing: Winning friends and influencing customers. eMarketer. [Online]. Available: http://www.emarketer.com/Reports/All/Emarketer_2000419.aspx

[16] F. G. Frobenius, "ber matrizen aus nicht negativen elementen," *Sitzungsberichte der Kniglich Preusischen Akademie der Wissenschafen zu Berlin*, vol. 4, pp. 456–477, 1912.

[17] E. Gombay, "Sequential Change-point Detection and Estimation," *Sequential Analysis*, vol. 22, pp. 203–222, 2003.

[18] P. Henrich, "Language identification for the automatic grapheme-to-phoneme conversion of foreign words in a german text-to-speech system," in *Proceedings of Eurospeech 1989*, 1989, pp. 2220–2223.

[19] T. Ho, R. Stapleton, and M. Subrahmanyam, "Multivariate Binomial Approximations for Asset Prices with Non-stationary Variance and Covariance Characteristics," *The Review of Financial Studies*, vol. 8, no. 4, pp. 1125–1152, 1995.

[20] J. Horrigan and L. Rainie, "Online communities: Networks that nurture long-distance relationships and local ties," Tech. Rep., 2001. [Online]. Available: http://www.pewinternet.org/reports/pdfs/PIP_Communities_Report.pdf

[21] N. C. Ingle, "A language identification table," *The Incorporated Linguist*, vol. 15, no. 4, pp. 98–101, 1976.

[22] World internet usage and population statistics. Internet World Stats. [Online]. Available: http://www.internetworldstats.com/stats.htm

[23] Y. M. Jang, "Estimation and Prediction-Based Connection Admission Control in Broadband Satellite Systems," *ETRI Journal*, vol. 22, no. 4, pp. 40–50, 2000.

[24] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," 1997.

[25] T. Joachims, *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[26] B. Jones, P. Garthwaite, and I. Jolliffe, *Statistical Inference*, 2nd ed. Oxford University Press, 2002.

[27] T. Kerwin, "Classification of natural language based on character frequency," 2006.

[28] W. E. Kohler and L. Johnson, *Elementary Differential Equations with Boundary Value Problems*. Addison Wesley, 2003.

[29] H. Kucera and W. N. Francis, "A standard corpus of present-day edited american english, for use with digital computers," 1964. [Online]. Available: http://khnt.aksis.uib.no/icame/manuals/brown/

[30] S. Lakshmivarahan, *Learning Algorithms Theory and Applications*. New York: Springer-Verlag, 1981.

[31] S. Lakshmivarahan and M. A. L. Thathachar, "Absolutely Expedient Algorithms for Stochastic Automata," *IEEE Trans. on System, Man and Cybernetics*, vol. SMC-3, pp. 281–286, 1973.

[32] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals, Tech. Rep. 8, 1966.

[33] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1998, pp. 296–304.

[34] Y. Ludovik and R. Zacharski, "Multilingual document language recognition for creating corpora," New Mexico State University, Tech. Rep., 1999.

[35] T. Mandl, M. Shramko, O. Tartakovski, and C. Womser-Hacker, "Language identification in multi-lingual web-documents." in *NLDB*, ser. Lecture Notes in Computer Science, C. Kop, G. Fliedl, H. C. Mayr, and E. Mtais, Eds., vol. 3999. Springer, 2006, pp. 153–163. [Online]. Available: http://dblp.uni-trier.de/db/conf/nldb/nldb2006.html#MandlSTW06

[36] A. A. Markov, "Rasprostranenie zakona bolshih chisel na velichiny, zavisyaschie drug ot druga," *Izvestiya Fizikomatematicheskogo obschestva pri Kazanskom universitete*, pp. 135–156, 1906.

[37] B. Martins and M. J. Silva, "Language identification in web pages," in *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2005, pp. 764–768.

[38] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. Springer-Verlag, 1993.

[39] S. Mustonen, "Multiple Discriminant Analysis in Linguistic Problems," *Statistical Methods in Linguistics*, vol. 4, pp. 37–44, 1965.

[40] I. J. Myung, "Tutorial on Maximum Likelihood Estimation," *Journal of Mathematical Psychology*, vol. 47, pp. 90–100, 2003.

[41] K. Narendra and M. Thathachar, *Learning Automata. An Introduction*. Prentice Hall, 1989.

[42] ——, "Learning automata — a survey," *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-4, pp. 323–334, 1974.

[43] ——, *Learning Automata*. Prentice-Hall, 1989.

[44] Topic detection and tracking research. National Institute of Standards and Technology. [Online]. Available: http://www.nist.gov/speech/tests/tdt/

[45] P. Norvig and S. Russel, *Artificial Intelligence A Modern Approach*. Prentice Hall, 2003.

[46] B. Oommen and L. Rueda, "Stochastic Learning-based Weak Estimation of Multinomial Random Variables and Its Applications to Non-stationary Environments," *Pattern Recognition*, vol. 39, no. 1, pp. 328–341, 2006.

[47] G. Ozbek, I. Rosenn, and E. Yeh, "Language classification in multilingual documents," Stanford University, Tech. Rep., 2006.

[48] F. Peng and D. Schuurmans, "Combining naive bayes and n-gram language models for text classification," 2003.

[49] O. Perron, "ber matrizen," *Mathematische Annalen*, vol. 64, pp. 248–263, 1907.

[50] Word-of-mouth marketing forecast 2007-2011. PQ Media. [Online]. Available: http://www.pqmedia.com/about-press-20071115-wommf.html

[51] H. Raiffa and R. Schlaifer, *Applied Statistical Decision Theory*. MIT Press, 1968.

[52] J. J. Rocchio, "Relevance feedback in information retrieval," in *The SMART Retrieval System - Experiments in Automatic Document Processing*, G. Salton, Ed. Prentice Hall, 1971.

[53] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw Hill Book Company, 1983.

[54] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[55] G. Salton, C. S. Yang, and C. Yu, "A theory of term importance in automatic text analysis," Ithaca, NY, USA, Tech. Rep., 1974.

[56] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Ithaca, NY, USA, Tech. Rep., 1987.

[57] H. Sorenson, *Parameter Estimation: Principles and Problems*. Marcel Dekker, 1980.

[58] M. A. L. Thathachar and B. J. Oommen, "Discretized Reward-Inaction Learning Automata," *Journal of Cybernetics and Information Sceinces*, pp. 24–29, Spring 1979.

[59] M. A. L. Thathachar and P. Sastry, "A Class of Rapidly Converging Algorithms for Learning Automata," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-15, pp. 168–175, 1985.

[60] ——, "Estimator Algorithms for Learning Automata," in *Proc. of the Platinum Jubilee Conference on Systems and Signal Processing*, Bangalore, India, December 1986.

[61] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 2nd ed. Elsevier Academic Press, 2003.

[62] C. J. Van Rijsbergen, *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[63] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[64] An introduction to word of mouth marketing. Word of Mouth Marketing Association. [Online]. Available: http://www.womma.org/

[65] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. T. Archibald, and X. Liu, "Learning approaches for detecting and tracking news events," 1999.

[66] D.-V. Ziegler, "The automatic identification of languages using linguistic recognition signals," Ph.D. dissertation, Buffalo, NY, USA, 1991.

[67] G. K. Zipf, *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley, 1949.