ONLINE LEARNING UNDER ADVERSE SETTINGS

A DISSERTATION SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Hüseyin Özkan
May 26, 2015

ONLINE LEARNING UNDER ADVERSE SETTINGS

By Hüseyin Özkan

May 26, 2015

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Süleyman Serdar Kozat (Advisor)

Assoc. Prof. Dr. Sinan Gezici

Prof. Dr. Levent Arslan

Prof. Dr. A. Aydın Alatan

Prof. Dr. Lale Akarun

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

ONLINE LEARNING UNDER ADVERSE SETTINGS

Hüseyin Özkan
Ph.D. in Electrical and Electronics Engineering
Advisor: Assoc. Prof. Dr. Süleyman Serdar Kozat
May 26, 2015

We present novel solutions for contemporary real life applications that generate data at unforeseen rates in unpredictable forms including non-stationarity, corruptions, missing/mixed attributes and high dimensionality. In particular, we introduce novel algorithms for *online learning*, where the observations are received sequentially and processed only once without being stored, *under adverse settings*: i) no or limited assumptions can be made about the data source, ii) the observations can be corrupted and iii) the data is to be processed at extremely fast rates. The introduced algorithms are highly effective and efficient with strong mathematical guarantees; and are shown, through the presented comprehensive real life experiments, to significantly outperform the competitors under such adverse conditions.

We develop a novel highly dynamical ensemble method without any stochastic assumptions on the data source. The presented method is asymptotically guaranteed to perform as well as, i.e., competitive against, the best expert in the ensemble, where the competitor, i.e., the best expert, itself is also specifically designed to continuously improve over time in a completely data adaptive manner. In addition, our algorithm achieves a significantly superior modeling power (hence, a significantly superior prediction performance) through a hierarchical and self-organizing approach while mitigating over training issues by combining (taking finite unions of) low-complexity methods. On the contrary, the state-of-the-art ensemble techniques are heavily dependent on static and unstructured expert ensembles. In this regard, we rigorously solve the resulting issues such as the over sensitivity to source statistics as well as the incompatibility between the modeling power and the computational load/precision. Our results uniformly hold for every possible input stream in the deterministic sense regardless of the stationary or non-stationary source statistics. Furthermore, we directly address the data corruptions by developing novel versatile imputation methods and thoroughly demonstrate that the anomaly detection -in addition to being stand alone an important learning problem- is extremely effective for corruption

detection/imputation purposes. To that end, as the first time in the literature, we develop the online implementation of the Neyman-Pearson characterization for anomalies in stationary or non-stationary fast streaming temporal data. The introduced anomaly detection algorithm maximizes the detection power at a specified controllable constant false alarm rate with no parameter tuning in a truly online manner.

Our algorithms can process any streaming data at extremely fast rates without requiring a training phase or a priori information while bearing strong performance guarantees. Through extensive experiments over real/synthetic benchmark data sets, we also show that our algorithms significantly outperform the state-of-the-art as well as the most recently proposed techniques in the literature with remarkable adaptation capabilities to non-stationarity.

Keywords: Online Learning, Supervised Learning, Prediction, Classification, Regression, Anomaly Detection, Big Data, Adverse Conditions, Deterministic Analysis, Worst Case, Non-Stationarity, Concept Change, Self-Organizing, Decision Tree, Hidden Markov Model, HMM, Partially Observable HMM States, Label Errors, Corruption, Noise, Anomaly, Imputation, Time Series, Neyman-Pearson.

ÖZET

KARŞIT KOŞULLAR ALTINDA ÇEVRİMİÇİ ÖĞRENME

Hüseyin Özkan
Elektrik Elektronik Mühendisliği, Doktora
Tez Danışmanı: Doç. Dr. Süleyman Serdar Kozat
Mayıs 26, 2015

İstatistiki açıdan durağan olmama ve veri bozulmaları ile kayıp/karışık yüksek boyutlu veri tipleri içerme gibi tahmin edilmesi güç mahiyette ve daha evvel karşılaşılmamış derecede çok miktarlarda veri üreten günümüz uygulamaları için yenilikçi çözümler üretmekteyiz. Bu minval üzere, gözlemlerin bir bir yapıldığı ve depolanmadan sadece bir defa işlendiği *çevrimiçi öğrenme* (İngilizce: *"online learning"*) için *karşıt koşullar altında* (İngilizce: *"under adverse settings"*) yenilikçi algoritmalar takdim etmekteyiz. Bahse konu bu karşıt koşullar öyledir ki, i) veri kaynağına istinaden ancak kısıtlı yani eser miktarda istatistiki varsayım yapılabilir, ii) gözlemler bozulmuş (İngilizce: "data corruptions") olabilir ve iii) veri oldukça hızlı işlenmek durumundadır. Hesapsal açıdan son derece verimli olan ve aynı zamanda güçlü matematiksel teminatları haiz çevrimiçi algoritmalarımızın, karşıt koşullar altında rakip yöntemlerden epey üstün bir performans sergilediği sunduğumuz kapsamlı gerçek veri deneyleri ile gösterilmektedir.

Veri kaynağı hakkında herhangi bir istatistiki varsayımda bulunmaksızın, yenilikçi ve olabildiğince dinamik bir "uzman-topluluk"/"uzman tavsiyeleriyle tahmin" (İngilizce: "expert ensemble"/"predicting with expert advice") yöntemi geliştirmekteyiz. Yöntemimizin, topluluğun en iyi uzmanınki kadar bir sonuşur (İngilizce: "asymptotical") performansı sergilediğini, yani en iyi uzmanla rekabetçi olduğunu, matematiksel olarak göstermekteyiz ki; burada, en iyi uzmanın kendisi dahi tamamen veriye uyarlanan bir yordamla zamanda sürekli gelişecek şekilde çalışmaktadır. İlaveten, bu çevrimiçi algoritmamızın tasarımında, sıradüzensel (İngilizce: "hierarchical") ve veriye göre kendi kendini organize edebilen bir yaklaşım vasıtasıyla üstün bir modelleme gücü (dolayısıyla üstün bir tahmin performansı) elde ederken; fazladan eğitme (İngilizce: "over training") sorunlarından da düşük karmaşıklı yöntemleri birleştirmek suretiyle kaçınıyoruz. Öte yandan, literatürdeki mevcut teknikler hem değişmeyen/sabit yani esnek olmayan hem de yapısal olarak (hesapsal açıdan) verimli bir temsile müsade

etmeyen uzman topluluklarına ciddi seviyede bağımlıdır. Bu sebeple ortaya çıkan veri kaynağı istatistiklerine aşırı duyarlılık ve modelleme gücü ile hesapsal yük/duyarlık arasındaki uyumsuzluk (istenmeyen veya ters orantı) gibi meseleleri sunduğumuz yöntemler ile açık bir biçimde çözmüş bulunmaktayız. Matematiksel olarak ispatladığımız performans garantileri muhtemel her girdi dizisi için durağan olabilen ya da olmayabilen veri istatistiklerinden bağımsız kesin (gerekirci) bir surette her zaman geçerlidir. Ayrıca, bozulma düzeltmede mahir algoritmalar geliştirerek veri bozulması problemleri için de doğrudan çözümler üretmekteyiz. Bu çözümleri geliştirirken etraflıca göstermekteyiz ki, anormallik tespiti -başlı başına önemli bir makine eğitimi problemi olmasının yanı sıra- bozulma tespiti/düzeltmesi maksadıyla etkili bir şekilde kullanılabilir. Bu bağlamda, süratli akan, kaynak istatistiği itibariyle durağan olan yahut olmayan zaman dizilerindeki anormal gözlemler için Neyman-Pearson karakterlendirmesinin çevrimiçi gereklemesini sağlayan bir algoritmayı literatürdeki ilk çalışma olarak sunmaktayız. Sunduğumuz algoritma, parametre ayarlamaları gerektirmeksizin, çevrimiçi bir şekilde, yanlış alarm oranını arzu edilen seviyede sabit tutabilirken anormallik tespit başarısını da en büyütmektedir.

Takdim ettiğimiz çevrimiçi algoritmalarımız, herhangi bir veri dizisini kaynağı hakkında bir ön bilgi gerektirmeden eğitim aşamasız ve varsayımsız son derece hızlı bir şekilde işleyebilirken, tahmin başarısına ilişkin de matematiksel olarak ispatlanmış güçlü teminatlar sağlamaktadır. Algoritmalarımızın, durağan olmayan veri kaynağı istatistiklerine dikkate şayan bir uyarlanabilirlikle, literatürde hem rutin olarak kullanılan hem de henüz önerilmiş muadil yöntemlerden önemli ölçüde üstün bir performans sergilediğini gerçek yahut suni olabilen bilindik veri kümeleri üzerinde yaptığımız geniş kapsamlı deneylerle göstermekteyiz.

Anahtar sözcükler: Çevrimiçi Öğrenme, Gözetimli Öğrenme, Tahmin, Sınıflandırma, Bağlanım, Anormallik Tespiti, Büyük Veri, Karşıt Koşullar, Gerekirci Çözümleme, En Kötü Durum, Durağan Olmayan, Kavram Değişimi, Kendi Kendini Organize Eden, Karar Ağacı, Saklı Markov Modeli, SMM, Kısmi Gözlemlenebilir SMM Durumları, Etiket Hataları, Bozulma, Gürültü, Anormallik, Düzeltme, Zaman Dizisi, Neyman-Pearson.

Acknowledgement

I thank my advisor Assoc. Prof. Dr. Süleyman Serdar Kozat for his invaluable guidance throughout my doctoral study. I cannot possibly overestimate the value of his advices, without which this thesis would have existential issues.

I thank my employer, the administrative board of ASELSAN Inc., Ankara, Turkey, for their great institutional support in graduate studies, who not only allow the graduate students to regularly attend the lectures at Bilkent without requiring work hour compensation but even designate a bus for every single lecture for them to conveniently commute.

I thank my co-authors Mr. Soner Özgun Pelvan, Mr. Nuri Denizcan Vanlı, Mr. Mehmet Ali Dönmez, Mr. Sait Tunç, Mr. Fatih Özkan and Mr. Arda Akman, who helped me in publishing my results with their careful reviews and contributions in the experiments of my papers.

I had insightful discussions with my colleagues from Algorithm Team of Electro Optic System Design Department at MGEO, ASELSAN Inc. throughout my study and I also thank them. I would like to specially mention Dr. Özgür Yılmaz and Dr. Sait Kubilay Pakin's names here.

Finally, I thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for their support of my study under the contract (program) 2211.

To my parents Serpil and Ali Naim Özkan

Tuti-i mucize guyem, ne desem laf değil,
Çerh ile söyleşemem ayinesi saf değil.

-vesselam.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the contemporary signal processing, communications and machine learning applications, algorithms are required to process data at a fast rate, yet to learn complex models often in a non-stationary environment [1–7]. An example is the news recommender application offered by a web site [2], where millions of user clicks (statement of interest) are typically received every day in addition to the huge volume of daily news. In this example, the web server is required to continuously *classify* the upcoming news on the basis of each user (interested or not) to make real time recommendations. Meanwhile, the learned detection/classification models must also be *updated* simultaneously since the user preferences can change over time, i.e., data is non-stationary. To address this ambitious goal, we introduce versatile methods to maximally exploit the information per instance -with only a single access- in the *online* setting and update the most recently learned hypothesis.

Furthermore, in various different applications from a wide range of fields such as time series analysis and computer vision, the data can partially (or even almost completely) be affected by severe noise and irregularities in several phases, e.g., occlusions during a visual recording or packet losses during transmission in a

communication channel or a CD scratch [8–10]. Data corruptions and irregularities, i.e., *anomalies*, often severely degrade the performance of the target application; for instance, face recognition or pedestrian detection under occlusion or news recommendation with missing attributes [9]. These realistic requirements to learn complex models from possibly corrupted anomalous observations in a non-stationary environment and with computationally scalable algorithms strongly challenge the modern learning systems. In this thesis, we introduce efficient algorithms to specifically handle such *adverse* conditions with strong performance guarantees that are therefore suitable for the real life applications including big data.

To this end, we consider the general problem of estimating an unknown outcome y that is coupled with a given observation x through a joint distribution $P(x, y)$. In general, x and y can both be multi-variate and discrete or continuous. When the distribution P is known or can be accurately estimated, this problem is a detection (y is discrete) or estimation (y is continuous) problem that has been well studied in the Bayesian framework [11]. When the distribution P is unknown or cannot be estimated accurately and a training batch of data $\{(x_i, y_i)\}$ is provided, then the problem is known to be a classification/clustering (y is discrete) or a regression (y is continuous) problem in the machine learning literature [12]. From the signal processing perspective, the observations (x_t, y_t) are received sequentially over time and one learns a model in a data driven manner to make predictions about future [13]. In this thesis, we investigate online learning/prediction [13, 14] in the deterministic sense without any stochastic assumptions, which is one of the most challenging tasks. Specifically, we process each instance x_t (streamed at time t in our sequential processing framework) only once and then discard it without storing. After making a prediction $\hat{y}_t$, the true outcome y_t is revealed. Based on the induced error by the true outcome, the most recently learned model is updated for the next coming instance. Furthermore, we consider adversely conditioned environments to address realistic scenarios. For instance, the unknown joint distribution $P_t(x_t, y_t)$ can be unpredictably non-stationary and occasionally overwritten by an external factor $\bar{P}(x_t, y_t)$, i.e., x_t or

y_t or both can be drawn from another distribution that produces, e.g., a corruption or an irregularity or an anomaly. Since it is infeasible to make any stochastic assumptions on the data source under such adverse settings, we study the problem in the deterministic sense and produce results in an individual sequence manner, i.e., our results hold for every possible input sequence regardless of stationary or non-stationary source statistics.

Overview

This thesis investigates online learning under adverse settings in two main directions. We first study the non-stationarity in Chapter 2 and Chapter 3; and then study the impact of corruptions and irregularities, i.e., anomalies, in terms of the detection and classification performances in Chapter 4, Chapter 5 and Chapter 6.

In Chapter 2, we concentrate on the well known "predicting with expert advice" framework [14], which has been applied with great success in online learning [14, 15] and adaptive signal processing [13] literatures. Along this line of research, we analyze an online learning algorithm [16] that adaptively combines outputs of two constituent algorithms (or the experts) running in parallel to estimate an unknown desired signal. This online learning algorithm is shown to achieve and in some cases outperform the mean-square error (MSE) performance of the best constituent algorithm in the steady-state [16]. However, the MSE analysis of this algorithm in the literature uses approximations and relies on statistical models on the underlying signals [16, 17]. Hence, such an analysis may not be useful or valid for signals generated by various real life systems that show high degrees of non-stationarity, limit cycles and that are even chaotic in many cases [18]. On the contrary, we present results in an individual sequence manner. In particular, we relate the time-accumulated squared estimation error of this online algorithm at any time over any interval to the one of the optimal convex mixture of the constituent algorithms directly tuned to the underlying signal in a

deterministic sense without any statistical assumptions. In this sense, our analysis provides the transient, steady-state and tracking behavior [14, 19, 20] of this algorithm in a "strong" sense without any approximations in the derivations or statistical assumptions on the underlying signals such that our results are guaranteed to hold under *adverse settings*. We illustrate the introduced results through examples.

Generally, in the "predicting with expert advice" framework [14], algorithms are designed to achieve the performance of the best expert algorithm in the ensemble [21]. However, the ensemble must be chosen largely for most of the practical cases to achieve a desirable performance in the final output, which then increases the computational load that possibly hinders the online processing [2]. Moreover, the non-stationarity in the data source cannot be predicted beforehand and therefore cannot be fully handled by a fixed set of ensemble [2, 22, 23]. To this end, in Chapter 3, we introduce a sufficiently large and dynamical ensemble that is hierarchically structured by a self organizing tree. Then, we propose a novel online algorithm with strong performance guarantees, which is computationally highly efficient as it is specially designed to exploit the introduced ensemble structure. The proposed algorithm in Chapter 3 is significantly more adaptive to non-stationarity and computationally more efficient compared to the studied method in Chapter 2. Similarly, we also provide the deterministic analysis of the proposed algorithm.

More precisely, in Chapter 3, we study online supervised learning under the empirical zero-one loss and introduce a novel classification algorithm with strong theoretical results that are guaranteed to hold under *adverse settings*. The proposed method is a highly dynamical self organizing decision tree structure, which adaptively partitions the feature space into small regions and combines (takes the union of) the local simple classification models specialized in those regions. Our approach sequentially and directly minimizes the cumulative loss by jointly learning the optimal feature space partitioning and the corresponding individual partition-region classifiers. We mitigate over training issues by using basic linear classifiers at each region while providing superior modeling power through

hierarchical and data adaptive models. The computational complexity of the introduced algorithm scales linearly with the dimensionality of the feature space and the depth of the tree. Our algorithm can be applied to any streaming data without requiring a training phase or a priori information, hence processes data on-the-fly and then discards it. Therefore, the introduced algorithm is especially suitable for applications involving big data. We present a comprehensive experimental study in stationary and non-stationary environments. In these experiments, our algorithm is compared with the state-of-the-art as well as the most recently proposed methods over the well-known benchmark data sets and shown to be computationally highly superior. The proposed algorithm significantly outperforms the competing methods in the stationary settings and demonstrates remarkable adaptation capabilities to non-stationarity in the presence of drifting concepts and abrupt/sudden concept changes.

In addition to the non-stationarity, we also study the impact of corruption in observations as another attribute of adversely conditioned environments, i.e., *adverse settings*. In Chapter 4, we present an introductory example, where we assume a model (HMM [24]) for the distribution $P(x_t, y_t)$ with random flipping errors (corruption) in discrete y_t's (partially and randomly observable noisy states). The corruptions, i.e., state errors, are defined as discrete random label flips within the specified HMM. We next drop the model assumption in Chapter 5 and cover a more general set of corruptions such as an occluded digit image, where the true observation model is unknown, whereas the corruptions follow a uniform distribution. Note that in both cases, we study in the batch setting and propose batch algorithms, which successfully handle the corruptions. We conclude that based on these studies (Chapter 4 and Chapter 5), the detrimental effects on the classification/prediction performance of the target application due to the corruptions can be undo, i.e., corrected, to a significant degree when the corruptions are formulated as anomalies and specifically treated after detected. Accordingly, in Chapter 6, we also drop the batch data processing requirement and propose a computationally highly efficient anomaly detection algorithm. In the following, we summarize these studies.

In Chapter 4, we propose a new estimation algorithm for the parameters of

an HMM as to best account for the observed data. In this model, in addition to the observation sequence, we have *partial* and *noisy* access to the hidden state sequence as side information. This access can be seen as "partial labeling" of the hidden states. Furthermore, we model possible mislabeling in the side information in a joint framework and derive the corresponding EM updates accordingly. In our simulations, we observe that using this side information, we considerably improve the state recognition performance, up to 70%, with respect to the "achievable margin" defined by the baseline algorithms. Moreover, our algorithm is shown to be robust to varying training conditions.

In Chapter 5, we introduce a comprehensive and statistical framework in a model free setting for a complete treatment of localized data corruptions due to severe noise sources, e.g., an occluder in the case of a visual recording. Within this framework, we propose i) a novel algorithm to efficiently separate, i.e., detect and localize, possible corruptions from a given suspicious data instance and ii) a *Maximum A Posteriori* (MAP) estimator to impute the corrupted data. As a generalization to Euclidean distance, we also propose a novel distance measure, which is based on the ranked deviations among the data attributes and empirically shown to be superior in separating the corruptions. Our algorithm first splits the suspicious instance into parts through a binary partitioning tree in the space of data attributes and iteratively tests those parts to detect local anomalies using the nominal statistics extracted from an uncorrupted (clean) reference data set. Once each part is labeled as anomalous vs normal, the corresponding binary patterns over this tree that characterize corruptions are identified and the affected attributes are imputed. Under a certain conditional independency structure assumed for the binary patterns, we analytically show that the false alarm rate of the introduced algorithm in detecting the corruptions is independent of the data and can be directly set without any parameter tuning. The proposed framework is tested over several well-known machine learning data sets with synthetically generated corruptions; and experimentally shown to produce remarkable improvements in terms of classification purposes with strong corruption separation capabilities. Our experiments also indicate that the proposed algorithms outperform the typical approaches and are robust to varying training

phase conditions.

We emphasize that in these studies of batch framework in Chapter 4 and Chapter 5, it is unknown whether an observation is corrupted or not. Moreover, which part of an instance is corrupted (even if it is detected to be corrupted) is also unknown. The presented results in Chapter 4 and Chapter 5 demonstrate that a corruption can be modeled as a rare event and formulated/detected as an anomaly. We show that an anomaly detection approach, in addition to being a stand alone important learning problem [25], is extremely useful in terms of corruption detection, imputation or correction purposes, when it is used in conjunction with the learning framework. However, it must also be devised in a computationally efficient manner for online learning.

To this end, in Chapter 6, we study anomaly detection for fast streaming temporal data with real time Type-I error, i.e., false alarm rate, controllability; and propose a computationally highly efficient online algorithm, which closely achieves a specified false alarm rate while maximizing the detection power. Regardless of whether the source is stationary or non-stationary, the proposed algorithm sequentially receives a time series and learns the nominal attributes -in the online setting- under possibly varying Markov statistics. Then, an anomaly is declared at a time instance, if the observations are statistically sufficiently deviant. Moreover, the proposed algorithm is remarkably versatile since it does not require parameter tuning to match the desired rates even in the case of strong non-stationarity. The presented study is the first to provide the online implementation of Neyman-Pearson (NP) characterization for the problem [26] such that the NP optimality [11], i.e., maximum detection power at a specified false alarm rate, is nearly achieved in a truly online manner. In this regard, the proposed algorithm is highly novel and appropriate especially for big data applications due to its parameter-tuning free computational efficient design with the practical NP constraints under stationary or non-stationary source statistics.

As a final remark in this section, we emphasize that learning from scarce data while mitigating the "curse of dimensionality" has long been the central theme in the signal processing and machine learning literatures. Nevertheless, the situation

is rapidly changing due to the recent advancements in information technologies. The issue in the contemporary applications is now the learning from (efficient processing of) huge amount of data presented in highly unstructured forms such as non-stationarity, mixed data types, missing attributes and corruptions. As a result, the-state-of-the-art learning systems are constantly being challenged by this changing theme and the resulting *adverse* conditions, i.e., learning from unstructured fast streaming big data under strict real time processing requirements. In addressing this challenge, we propose online learning algorithms with strong theoretical guarantees that are experimentally shown to outperform the classical approaches. The thesis concludes with final remarks in Chapter 7.

Chapter 2

A Deterministic Analysis of an Online Convex Mixture of Experts Algorithm

The problem of estimating or learning an unknown desired signal is heavily investigated in online learning [14, 19, 20, 27–30] and adaptive signal processing literature [13, 16, 17, 31–33]. However, in various applications, certain difficulties arise in the estimation process due to the lack of structural and statistical information about the data model. To resolve this issue, mixture approaches that adaptively combine outputs of multiple constituent algorithms performing the same task are proposed in the online learning literature under the mixture of experts framework [14, 19, 20] and adaptive signal processing literature under the adaptive mixture methods framework [16, 17]. Along these lines, an online convexly constrained mixture of experts method that combines outputs of two learning algorithms is introduced in [16]. We point out that the "mixture of experts" framework also refers to a model [34], where the input space is divided into regions in a nested fashion to each of which an expert corresponds. The partitioning of the input space and the corresponding experts are learned jointly and combined such that a mixture of experts method is obtained. On the other hand, the mixture method in [16] adaptively combines the outputs of the constituent

algorithms that run in parallel on the same task under a convex constraint to minimize the final MSE. This adaptive mixture is shown to be universal with respect to the input algorithms in a certain stochastic sense such that this mixture achieves and in some cases outperforms the MSE performance of the best constituent algorithm in the steady-state [16]. However, the MSE analysis of this adaptive mixture in the transient and steady states uses approximations such as the separation assumptions and relies on strict statistical models on the signals, e.g., stationary data models [16,17]. In this chapter, we study this algorithm [16] from the perspective of online learning and produce results in an individual sequence manner such that our results are guaranteed to hold for any bounded arbitrary signal.

Nevertheless, signals produced by various real life systems, such as in underwater acoustic communication applications, show high degrees of nonstationarity, limit cycles and, in many cases, are even chaotic so that they hardly fit to assumed statistical models [18]. Hence an analysis based on certain statistical assumptions or approximations may not be useful or adequate under these conditions. To this end, we refrain from making any statistical assumptions on the underlying signals and present an analysis that is guaranteed to hold for any bounded arbitrary signal without any approximations. In particular, we relate the performance of this learning algorithm that adaptively combines outputs of two constituent algorithms to the performance of the optimal convex combination that is directly tuned to the underlying signal in a deterministic sense. Naturally, this optimal convex combination can only be chosen in hindsight after observing the whole signal a priori (before we even start processing the data). Since we compare the performance of this algorithm with respect to the best convex combination of the constituent filters in a deterministic sense over any time interval, our analysis provides, without any assumptions, the transient, the tracking and the steady-state behaviors together [14, 19, 20]. In particular, if the analysis window starts from $t = 1$, then we obtain the transient behavior; if the window length goes to infinity, then we obtain the steady-state behavior; and finally if the analyze window is selected arbitrary, then we get the tracking behavior as explained in detail in Section 2.2. The corresponding bounds may also hold for unbounded signals

such as with Gaussian and Laplacian distributions, if one can define reasonable bounds such that a sample stays in the defined interval with high probability.

After we provide a brief problem description in Section 2.1, we present a deterministic analysis of the convexly constrained mixture algorithm in Section 2.2, where the performance bounds are given as a theorem and a corresponding corollary. We illustrate the introduced results through examples in Section 2.3. The chapter concludes with certain remarks.

2.1 Problem Description

In this framework, we have a desired signal $\{y_t\}_{t\geq 1} \subset \mathbb{R}$, where t is the time index, and two constituent algorithms running in parallel producing $\{\hat{y}_{1,t}\}_{t\geq 1}$ and $\{\hat{y}_{2,t}\}_{t\geq 1}$, respectively, as the estimations (or predictions) of the desired signal. We assume that the desired signal $\{y_t\}_{t\geq 1}$ is finite and bounded by a known constant Y, i.e., $|y_t| \leq Y < \infty$. Here, we have no restrictions on $\hat{y}_{1,t}$ or $\hat{y}_{2,t}$, e.g., these outputs are not required to be causal, however, without loss of generality, we assume $|\hat{y}_{1,t}| \leq Y$ and $|\hat{y}_{2,t}| \leq Y$, i.e., these outputs can be clipped to the range $[-Y, Y]$ without sacrificing performance under the squared error. As an example, the desired signal and outputs of the constituent learning algorithms can be single realizations generated under the framework of [16]. At each time t, the convexly constrained algorithm receives an input vector $\boldsymbol{x}_t \stackrel{\triangle}{=} [\hat{y}_{1,t}\ \hat{y}_{2,t}]^T$ and outputs

$$\hat{y}_t = \lambda_t \hat{y}_{1,t} + (1-\lambda_t)\hat{y}_{2,t} = \boldsymbol{w}_t^T \boldsymbol{x}_t,$$

where $\boldsymbol{w}_t \stackrel{\triangle}{=} [\lambda_t\ (1-\lambda_t)]^T$, $0 \leq \lambda_t \leq 1$, as the final estimate. The final estimation error is given by $e_t = y_t - \hat{y}_t$. The combination weight λ_t is trained through an auxiliary variable ρ_t using a stochastic gradient update to minimize the squared final estimation error as

$$\lambda_t = \frac{1}{1+e^{-\rho_t}}, \tag{2.1}$$

$$\rho_{t+1} = \rho_t - \mu \nabla_{\rho_t} e_t^2 = \rho_t + \mu e_t \lambda_t (1-\lambda_t)[\hat{y}_{1,t} - \hat{y}_{2,t}], \tag{2.2}$$

The Convexly Constrained Algorithm:
Parameters: $\mu > 0$: learning rate. $\lambda^+ \in [0, \frac{1}{2}]$: clipping parameter. **Inputs:** y_t: desired signal. $\hat{y}_{1,t}, \hat{y}_{2,t}$: constituent learning algorithms. **Outputs:** $\hat{y}_t$: estimate of the desired signal.
Initialization: Set the initial weights $\lambda_1 = 1/2$ and $\rho_1 = 0$. for $t = 1 : \ldots : n$, % receive the constituent algorithm outputs $\hat{y}_{1,t}$ and $\hat{y}_{2,t}$ and % estimate the desired signal $\hat{y}_t = \lambda_t \hat{y}_{1,t} + (1 - \lambda_t)\hat{y}_{2,t}$ % Upon receiving y_t, update the weight according to the rule: $\rho_{t+1} = \rho_t + \mu e_t \lambda_t (1 - \lambda_t)[\hat{y}_{1,t} - \hat{y}_{2,t}]$ $\lambda_{t+1} = \frac{1}{1+e^{-\rho_{t+1}}}$ $\lambda_{t+1} \leftarrow \lambda^+$ and $\rho_{t+1} = \ln \frac{\lambda_{t+1}}{1-\lambda_{t+1}}$, if $\lambda_{t+1} < \lambda^+$ $\lambda_{t+1} \leftarrow 1 - \lambda^+$ and $\rho_{t+1} = \ln \frac{\lambda_{t+1}}{1-\lambda_{t+1}}$, if $\lambda_{t+1} > 1 - \lambda^+$ endfor

Table 2.1: The studied learning algorithm that adaptively combines outputs of two algorithms.

where $\mu > 0$ is the learning rate. The combination parameter λ_t in (2.1) is constrained to lie in $[\lambda^+, (1 - \lambda^+)]$, $0 < \lambda^+ < 1/2$ in [16], since the update in (2.2) may slow down when λ_t is too close to the boundaries. We follow the same restriction and analyze (2.2) under this constraint. The algorithm, presented in Table 2.1, consists of two steps: (1) the update step of the parameter ρ: $\rho_{t+1} = \rho_t + \mu e_t \lambda_t (1 - \lambda_t)[\hat{y}_{1,t} - \hat{y}_{2,t}]$ and (2) the mapping of ρ back to the corresponding combination parameter λ: $\lambda_{t+1} = \frac{1}{1+e^{-\rho_{t+1}}}$. At every time, the update step requires 6 multiplications and 3 additions (2 multiplications and 1 addition for calculating e_t); the mapping of ρ simply requires 1 division and 2 additions (taking the exponent is only a look-up). This is per time, i.e., the computational complexity does not increase with time. As for the multidimensional case, the corresponding complexity scales linearly with the input dimensionality. Hence, the complexity is $O(d)$, where d is the dimension of the input regressor.

Under the deterministic analysis framework, the performance of the algorithm

is determined by the time-accumulated squared error [14, 20, 35]. When applied to any sequence $\{y_t\}_{t\geq 1}$, the algorithm of (2.1) yields the total accumulated loss

$$L_n(\hat{y}, y) = L_n(\boldsymbol{w}_t^T \boldsymbol{x}_t, y) \stackrel{\triangle}{=} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2 \tag{2.3}$$

for any n. We emphasize that for unbounded signals such as Gaussian and Laplacian distributions, we can define a suitable Y such that the samples of y_t are inside of the interval $[-Y, Y]$ with high probability.

We next provide deterministic bounds on $L_n(\hat{y}, y)$ with respect to the best convex combination $\min_{\beta\in[\lambda^+, 1-\lambda^+]} L_n(\hat{y}_\beta, y)$, where

$$L_n(\hat{y}_\beta, y) = L_n(\boldsymbol{u}^T \boldsymbol{x}_t, y) = \sum_{t=1}^{n} (y_t - \hat{y}_{\beta,t})^2$$

and $\hat{y}_{\beta,t} \stackrel{\triangle}{=} \beta \hat{y}_{1,t} + (1-\beta)\hat{y}_{2,t} = \boldsymbol{u}^T \boldsymbol{x}_t$, $\boldsymbol{u} \stackrel{\triangle}{=} [\beta \; 1-\beta]^T$, that holds uniformly in an individual sequence manner without any stochastic assumptions on y_t, $\hat{y}_{1,t}$, $\hat{y}_{2,t}$ or n. Note that the best fixed convex combination parameter

$$\beta_o = \arg \min_{\beta\in[\lambda^+, 1-\lambda^+]} L_n(\hat{y}_\beta, y)$$

and the corresponding estimator

$$\hat{y}_{\beta_o,t} = \beta_o \hat{y}_{1,t} + (1-\beta_o)\hat{y}_{2,t},$$

which we compare the performance against, can only be determined after observing the entire sequences, i.e., $\{y_t\}, \{\hat{y}_{1,t}\}$ and $\{\hat{y}_{2,t}\}$, in advance for all n.

2.2 A Deterministic Analysis

In this section, we first relate the accumulated loss of the mixture to the accumulated loss of the best convex combination that minimizes the accumulated loss in the following theorem. Then, we discuss the implications of the our theorem in a corollary to compare the adaptive mixture [16] with the Exponentiated Gradient

algorithm [19]. The use of the Kullback-Leibler (KL) divergence as a distance measure for obtaining worst-case loss bounds was pioneered by Littlestone [36], and later adopted extensively in the online learning literature [19,20,37]. We emphasize that although the steady-state and transient MSE performances of the convexly constrained mixture algorithm are analyzed with respect to the constituent learning algorithms [16, 17], we perform the steady-state, transient and tracking analysis without any stochastic assumptions or use any approximations in the following theorem.

Theorem 1: The algorithm given in (2.2), when applied to any sequence $\{y_t\}_{t\geq 1}$, with $|y_t| \leq Y < \infty$, $0 < \lambda^+ < 1/2$, $\beta \in [\lambda^+, 1-\lambda^+]$, yields for any n and $\epsilon > 0$

$$\begin{aligned} & L_n(\hat{y}, y) - \left(\frac{2\epsilon+1}{1-z^2}\right) \min_{\beta} \{L_n(\hat{y}_\beta, y)\} \\ & \leq \frac{(2\epsilon+1)Y^2}{\epsilon(1-z^2)} \ln 2 \leq O\left(\frac{1}{\epsilon}\right), \end{aligned} \tag{2.4}$$

where $O(.)$ is the order notation, $\hat{y}_{\beta,t} = \beta \hat{y}_{1,t} + (1-\beta)\hat{y}_{2,t}$, $z \stackrel{\triangle}{=} \frac{1-4\lambda^+(1-\lambda^+)}{1+4\lambda^+(1-\lambda^+)} < 1$ and step size $\mu = \frac{4\epsilon}{2\epsilon+1}\frac{2+2z}{Y^2}$.

This theorem provides a regret bound for the algorithm (2.2) showing that the cumulative loss of the convexly constrained algorithm is close to a factor times the cumulative loss of the algorithm with the best weight chosen in hindsight. If we define the regret

$$R_n \stackrel{\triangle}{=} L_n(\hat{y}, y) - \left(\frac{2\epsilon+1}{1-z^2}\right) \min_{\beta} \{L_n(\hat{y}_\beta, y)\}, \tag{2.5}$$

then equation (2.4) implies that time-normalized regret

$$\frac{R_n}{n} \stackrel{\triangle}{=} \frac{L_n(\hat{y}, y)}{n} - \left(\frac{2\epsilon+1}{1-z^2}\right) \min_{\beta} \left\{\frac{L_n(\hat{y}_\beta, y)}{n}\right\}$$

converges to zero at a rate $O\left(\frac{1}{n\epsilon}\right)$ uniformly over the desired signal and the outputs of constituent algorithms. Moreover, (2.4) provides the exact trade-off between the transient and steady-state performances of the convex mixture in a deterministic sense without any assumptions or approximations. Note that (2.4)

is guaranteed to hold independent of the initial condition of the combination weight λ_t for any time interval in an individual sequence manner. Hence, (2.4) also provides the tracking performance of the convexly constrained algorithm in a deterministic sense.

From (2.4), we observe that the convergence rate of the right hand side, i.e., the bound, is $O\left(\frac{1}{n\epsilon}\right)$, and, as in the stochastic case [17], to get a tighter asymptotic bound with respect to the optimal convex combination of the learning algorithms, we require a smaller ϵ, i.e., smaller learning rate μ, which increases the right hand side of (2.4). Although this result is well-known in the adaptive filtering literature and appears widely in stochastic contexts, however, this trade-off is guaranteed to hold in here without any statistical assumptions or approximations. Note that the optimal convex combination in (2.4) depends on the entire signal and outputs of the constituent algorithms for all n and hence it can only be determined in hindsight.

Proof: To prove the theorem, we initially assume that clipping never happens in the course of the algorithm, i.e., it is either not required or the allowed range is never violated by λ_t. Then the extension to the case of the clipping will be straightforward. In the following, we use the approach introduced in [20] (and later used in [19]) based on measuring progress of a mixture algorithm using certain distance measures.

We first convert (2.2) to a direct update on λ_t and use this direct update in the proof. Using $e^{-\rho_t} = \frac{1-\lambda_t}{\lambda_t}$, the update in (2.2) can be written as

$$\lambda_{t+1} = \frac{\lambda_t e^{\mu e_t \lambda_t (1-\lambda_t) \hat{y}_{1,t}}}{\lambda_t e^{\mu e_t \lambda_t (1-\lambda_t) \hat{y}_{1,t}} + (1-\lambda_t) e^{\mu e_t \lambda_t (1-\lambda_t) \hat{y}_{2,t}}}. \tag{2.6}$$

Unlike [19] (Lemma 5.8), our update in (2.6) has, in a certain sense, an adaptive learning rate $\mu\lambda_t(1-\lambda_t)$ which requires different formulation, however, follows similar lines of [19] in certain parts.

Here, for a fixed β, we define an estimator

$$\hat{y}_{\beta,t} \stackrel{\triangle}{=} \beta \hat{y}_{1,t} + (1-\beta)\hat{y}_{2,t} = \boldsymbol{u}^T \boldsymbol{x}_t,$$

where $\beta \in [\lambda^+, 1-\lambda^+]$ and $\boldsymbol{u} \stackrel{\triangle}{=} [\beta \;\; 1-\beta]^T$. Defining $\zeta_t = e^{\mu e_t \lambda_t (1-\lambda_t)}$, we have from (2.6)

$$\begin{aligned} &\beta \ln\left(\frac{\lambda_{t+1}}{\lambda_t}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{t+1}}{1-\lambda_t}\right) \\ &= \hat{y}_{\beta,t} \ln \zeta_t - \ln\left(\lambda_t \zeta_t^{\hat{y}_{1,t}} + (1-\lambda_t)\zeta_t^{\hat{y}_{2,t}}\right). \end{aligned} \tag{2.7}$$

Using the inequality $\alpha^x \leq 1 - x(1-\alpha)$ for $\alpha \geq 0$ and $x \in [0,1]$ from [20], we have

$$\zeta_t^{\hat{y}_{1,t}} = (\zeta_t^{2Y})^{\frac{\hat{y}_{1,t}+Y}{2Y}} \zeta_t^{-Y} \leq \zeta_t^{-Y}\left(1 - \frac{\hat{y}_{1,t}+Y}{2Y}(1-\zeta_t^{2Y})\right),$$

which implies in (2.7)

$$\begin{aligned} &\ln\left(\lambda_t \zeta_t^{\hat{y}_{1,t}} + (1-\lambda_t)\zeta_t^{\hat{y}_{2,t}}\right) \\ &\leq -Y \ln \zeta_t + \ln\left(1 - \frac{\hat{y}_t + Y}{2Y}(1-\zeta_t^{2Y})\right), \end{aligned} \tag{2.8}$$

where $\hat{y}_t = \lambda_t \hat{y}_{1,t} + (1-\lambda_t)\hat{y}_{2,t}$. As in [19], one can further bound (2.8) using $\ln(1 - q(1-e^p)) \leq pq + \frac{p^2}{8}$ for $0 \leq q < 1$

$$\begin{aligned} &\ln\left(\lambda_t \zeta_t^{\hat{y}_{1,t}} + (1-\lambda_t)\zeta_t^{\hat{y}_{2,t}}\right) \\ &\leq -Y \ln \zeta_t + (\hat{y}_t + Y)\ln \zeta_t + \frac{Y^2 (\ln \zeta_t)^2}{2}. \end{aligned} \tag{2.9}$$

Using (2.9) in (2.7) yields

$$\begin{aligned} &\beta \ln\left(\frac{\lambda_{t+1}}{\lambda_t}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{t+1}}{1-\lambda_t}\right) \geq \\ &(\hat{y}_{\beta,t} + Y)\ln \zeta_t - (\hat{y}_t + Y)\ln \zeta_t - \frac{Y^2(\ln \zeta_t)^2}{2}. \end{aligned} \tag{2.10}$$

Now for the case of clipping let us suppose without loss of generality $\lambda_{t+1} = \lambda^+ - \alpha$, where $\lambda^+ > \alpha > 0$ so that it is set back to λ^+. We claim that the left hand side of (2.10) can only increase by clipping and hence, (2.10) stays valid after clipping. Since the derivative of $\ln x$ is monotonically decreasing with x and always positive, $\ln(\lambda_{t+1})$ must increase not less than $\frac{\alpha}{\lambda^+}$ after clipping. On the other hand, $\ln(1-\lambda_{t+1})$ can decrease not more than $\frac{\alpha}{1-\lambda^+}$ after clipping. As a result, $\beta \ln(\lambda_{t+1}) + (1-\beta)\ln(1-\lambda_{t+1})$ must increase not less than $\delta = \beta\frac{\alpha}{\lambda^+} - (1-\beta)\frac{\alpha}{1-\lambda^+}$

after clipping. Since $\beta \in [\lambda^+, 1-\lambda^+]$, $\delta \geq 0$. Hence, (2.10) is valid even after clipping.

At each adaptation, the progress made by the algorithm towards $\boldsymbol{u}$ at time t is measured as $D(\boldsymbol{u}||\boldsymbol{w}_t) - D(\boldsymbol{u}||\boldsymbol{w}_{t+1})$, where $\boldsymbol{w}_t \stackrel{\triangle}{=} [\lambda_t \ (1-\lambda_t)]^T$ and

$$D(\boldsymbol{u}||\boldsymbol{w}) \stackrel{\triangle}{=} \sum_{i=1}^{2} u_i \ln(u_i/w_i)$$

is the KL divergence [20], $\boldsymbol{u} \in [0,1]^2$, $\boldsymbol{w} \in [0,1]^2$. We require that this progress is at least $a(y_t - \hat{y}_t)^2 - b(y_t - \hat{y}_{\beta,t})^2$ for certain a, b, μ [19, 20], i.e.,

$$\begin{aligned} &a(y_t - \hat{y}_t)^2 - b(y_t - \hat{y}_{\beta,t})^2 \leq D(\boldsymbol{u}||\boldsymbol{w}_t) - D(\boldsymbol{u}||\boldsymbol{w}_{t+1}) \\ &= \beta \ln\left(\frac{\lambda_{t+1}}{\lambda_t}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{t+1}}{1-\lambda_t}\right), \end{aligned} \tag{2.11}$$

which yields the desired deterministic bound in (2.4) after telescoping. In information theory and probability theory, the KL divergence, which is also known as the relative entropy, is empirically shown to be an efficient measure of the distance between two probability vectors [19, 20]. Here, the vectors $\boldsymbol{u}$ and $\boldsymbol{w}_t$ are probability vectors, i.e., $\boldsymbol{u}, \boldsymbol{w}_t \in [0,1]^2$ and $\boldsymbol{u}^T\mathbf{1} = \boldsymbol{w}_t^T\mathbf{1} = 1$, where $\mathbf{1} \stackrel{\triangle}{=} [1 \ 1]^T$. This use of KL divergence as a distance measure between weight vectors is widespread in the online learning literature [19, 37].

We observe from (2.11) and (2.10) that to prove the theorem, it is sufficient to show that $G(y_t, \hat{y}_t, \hat{y}_{\beta,t}, \zeta_t) \leq 0$, where

$$\begin{aligned} G(y_t, \hat{y}_t, \hat{y}_{\beta,t}, \zeta_t) &\stackrel{\triangle}{=} -(\hat{y}_{\beta,t} + Y)\ln\zeta_t + (\hat{y}_t + Y)\ln\zeta_t \\ &+ \frac{Y^2(\ln\zeta_t)^2}{2} + a(y_t - \hat{y}_t)^2 - b(y_t - \hat{y}_{\beta,t})^2. \end{aligned}$$

For fixed $y_t, \hat{y}_t, \zeta_t$, $G(y_t, \hat{y}_t, \hat{y}_{\beta,t}, \zeta_t)$ is maximized when $\frac{\partial G}{\partial \hat{y}_{\beta,t}} = 0$, i.e., $\hat{y}_{\beta,t} - y_t + \frac{\ln\zeta_t}{2b} = 0$, since $\frac{\partial^2 G}{\partial \hat{y}_{\beta,t}^2} = -2b < 0$, yielding $\hat{y}^*_{\beta,t} = y_t - \frac{\ln\zeta_t}{2b}$. Note that while taking the partial derivative of $G(\cdot)$ with respect to $\hat{y}_{\beta,t}$ and finding $\hat{y}^*_{\beta,t}$, we assume that all $y_t, \hat{y}_t, \zeta_t$ are fixed. This yields an upper bound on $G(\cdot)$ in terms of $\hat{y}_{\beta,t}$. Hence,

it is sufficient to show $G(y_t, \hat{y}_t, \hat{y}^*_{\beta,t}, \zeta_t) \leq 0$, where, after some algebra, [19]

$$G(y_t, \hat{y}, \hat{y}^*_{\beta,t}, \zeta_t) = (y_t - \hat{y}_t)^2 \times \Bigg[a - \mu\lambda_t(1-\lambda_t) + \frac{\mu^2 {\lambda_t}^2 (1-\lambda_t)^2}{4b} + \frac{Y^2 \mu^2 {\lambda_t}^2 (1-\lambda_t)^2}{2} \Bigg]. \tag{2.12}$$

For (2.12) to be negative, defining $k \stackrel{\triangle}{=} \lambda_t(1-\lambda_t)$ and

$$H(k) \stackrel{\triangle}{=} k^2\mu^2(\frac{Y^2}{2} + \frac{1}{4b}) - \mu k + a,$$

it is sufficient to show that $H(k) \leq 0$ for $k \in [\lambda^+(1-\lambda^+), \frac{1}{4}]$, i.e., $k \in [\lambda^+(1-\lambda^+), \frac{1}{4}]$ when $\lambda_t \in [\lambda^+, (1-\lambda^+)]$, since $H(k)$ is a convex quadratic function of k, i.e., $\frac{\partial^2 H}{\partial k^2} > 0$. Hence, we require the interval where the function $H(\cdot)$ is negative should include $[\lambda^+(1-\lambda^+), \frac{1}{4}]$, i.e., the roots k_1 and k_2 (where $k_2 \leq k_1$) of $H(\cdot)$ should satisfy

$$k_1 \geq \frac{1}{4},\ k_2 \leq \lambda^+(1-\lambda^+),\ \text{where}$$

$$k_1 = \frac{\mu + \sqrt{\mu^2 - 4\mu^2 a\left(\frac{Y^2}{2} + \frac{1}{4b}\right)}}{2\mu^2(\frac{Y^2}{2} + \frac{1}{4b})} = \frac{1 + \sqrt{1-4as}}{2\mu s}, \tag{2.13}$$

$$k_2 = \frac{\mu - \sqrt{\mu^2 - 4\mu^2 a\left(\frac{Y^2}{2} + \frac{1}{4b}\right)}}{2\mu^2(\frac{Y^2}{2} + \frac{1}{4b})} = \frac{1 - \sqrt{1-4as}}{2\mu s}, \tag{2.14}$$

$$s \stackrel{\triangle}{=} \left(\frac{Y^2}{2} + \frac{1}{4b}\right).$$

To satisfy $k_1 \geq 1/4$, we straightforwardly require from (2.13)

$$\frac{2 + 2\sqrt{1-4as}}{s} \geq \mu.$$

To get the tightest upper bound for (2.13), we set

$$\mu = \frac{2 + 2\sqrt{1-4as}}{s},$$

i.e., the largest allowable learning rate.

To have $k_2 \leq \lambda^+(1-\lambda^+)$ with $\mu = \frac{2+2\sqrt{1-4as}}{s}$, from (2.14) we require

$$\frac{1 - \sqrt{1-4as}}{4(1+\sqrt{1-4as})} \leq \lambda^+(1-\lambda^+). \tag{2.15}$$

Equation (2.15) yields

$$as = a\left(\frac{Y^2}{2} + \frac{1}{4b}\right) \leq \frac{1-z^2}{4}, \tag{2.16}$$

where

$$z \triangleq \frac{1 - 4\lambda^+(1-\lambda^+)}{1 + 4\lambda^+(1-\lambda^+)}$$

and $z < 1$ after some algebra.

To satisfy (2.16), we set $b = \frac{\epsilon}{Y^2}$ for any (or arbitrarily small) $\epsilon > 0$ that results

$$a \leq \frac{(1-z^2)\epsilon}{Y^2(2\epsilon+1)}. \tag{2.17}$$

To get the tightest bound in (2.11), we select

$$a = \frac{(1-z^2)\epsilon}{Y^2(2\epsilon+1)}$$

in (2.17). Such selection of a, b and μ results in (2.11)

$$\begin{aligned}
&\left(\frac{(1-z^2)\epsilon}{Y^2(2\epsilon+1)}\right)(y_t - \hat{y}_t)^2 - \left(\frac{\epsilon}{Y^2}\right)(y_t - \hat{y}_{\beta,t})^2 \\
&\leq \beta \ln\left(\frac{\lambda_{t+1}}{\lambda_t}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{t+1}}{1-\lambda_t}\right).
\end{aligned} \tag{2.18}$$

After telescoping, i.e., summation over t, $\sum_{t=1}^{n}$, (2.18) yields

$$\begin{aligned}
&aL_n(\hat{y}, y) - b \min_{\beta}\{L_n(\hat{y}_\beta, y)\} \\
&\leq \beta \ln\left(\frac{\lambda_{n+1}}{\lambda_1}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{n+1}}{1-\lambda_1}\right) \leq \ln 2 \leq O(1),
\end{aligned}$$

where $\beta \ln\left(\frac{\lambda_{n+1}}{\lambda_1}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{n+1}}{1-\lambda_1}\right) \leq \ln 2$ since we initialize the algorithm with $\lambda_1 = \frac{1}{2}$. Note for a random initialization that this bound would correspond to in general $\beta \ln\left(\frac{\lambda_{n+1}}{\lambda_1}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{n+1}}{1-\lambda_1}\right) \leq -\left((1-\lambda^+)\ln\lambda^+ + \lambda^+\ln(1-\lambda^+)\right) = O(1)$. Hence,

$$\begin{aligned}
&\left(\frac{(1-z^2)\epsilon}{Y^2(2\epsilon+1)}\right) L_n(\hat{y}, y) - \left(\frac{\epsilon}{Y^2}\right)\min_{\beta}\{L_n(\hat{y}_\beta, y)\} \\
&\leq \ln 2 \leq O(1).
\end{aligned}$$

Then it follows that

$$L_n(\hat{y}, y) - \left(\frac{2\epsilon + 1}{1 - z^2}\right) \min_{\beta} \{L_n(\hat{y}_\beta, y)\}$$
$$\leq \frac{(2\epsilon + 1)Y^2}{\epsilon(1 - z^2)} \ln 2 \leq O\left(\frac{1}{\epsilon}\right), \tag{2.19}$$

which is the desired bound.

Note that using

$$b = \frac{\epsilon}{Y^2},\ a = \frac{(1 - z^2)\epsilon}{Y^2(2\epsilon + 1)},\ s = \left(\frac{Y^2}{2} + \frac{1}{4b}\right),$$

we get

$$\mu = \frac{2 + 2\sqrt{1 - 4as}}{s} = \frac{4\epsilon}{2\epsilon + 1}\frac{2 + 2z}{Y^2},$$

after some algebra, as in the statement of the theorem. □

Finally, we also define a time-normalized regret as in [19] to have a comparison between the exponentiated gradient algorithm and the adaptive mixture given in (2.2). Let us define the regret R_n^* as

$$R_n^* \stackrel{\triangle}{=} L_n(\hat{y}, y) - \min_{\beta} \{L_n(\hat{y}_\beta, y)\}, \tag{2.20}$$

then in the following corollary, we show that the time normalized regret $\frac{1}{n}R_n^*$ for the algorithm proposed originally in [16] and given in this chapter in (2.2) improves, i.e., decreases, with $O(n^{\frac{-1}{2}})$ in a similar manner to the Exponentiated Gradient algorithm [19], except that the time-normalized regret $\frac{1}{n}R_n^*$ is always above an error floor, i.e., it is a linear regret with n and hence, it does not converge to 0.

Corollary 1: The algorithm given in (2.2), when applied to any sequence $\{y_t\}_{t\geq 1}$,

with $|y_t| \leq Y < \infty$, $0 < \lambda^+ < 1/2$, $\beta \in [\lambda^+, 1-\lambda^+]$, yields for any n

$$\begin{aligned}\frac{1}{n}R_n^* &\leq \frac{4Y\epsilon}{1-z^2} + \frac{2Yz^2}{1-z^2} + \frac{Y^2\ln 2}{n(1-z^2)}(2+\frac{1}{\epsilon}) \\ &\leq O\left(n^{\frac{-1}{2}}\right) + O\left(1\right),\end{aligned}$$

where $O(.)$ is the order notation, R_n^* is defined in (2.20), $z \stackrel{\triangle}{=} \frac{1-4\lambda^+(1-\lambda^+)}{1+4\lambda^+(1-\lambda^+)} < 1$, $\epsilon = \sqrt{\frac{Y\ln 2}{4n}}$ and step size $\mu = \frac{4\epsilon}{2\epsilon+1}\frac{2+2z}{Y^2}$.

Proof: We first note that $\beta \ln\left(\frac{\lambda_{n+1}}{\lambda_1}\right) + (1-\beta)\ln\left(\frac{1-\lambda_{n+1}}{1-\lambda_1}\right) \leq \ln 2$ for $\beta, \lambda_n \in [0,1], \forall n$ since $\lambda_1 = \frac{1}{2}$ and $L_n(\hat{y}_\beta, y) \leq 2Yn, \forall\beta$. Then from (2.18) and (2.19),

$$\begin{aligned}&L_n(\hat{y}, y) - L_n(\hat{y}_\beta, y) \leq \gamma(\epsilon), \forall\beta, \\ &\text{and } \gamma(\epsilon) = \frac{4Y\epsilon n}{1-z^2} + \frac{2Yz^2 n}{1-z^2} + \frac{Y^2\ln 2}{1-z^2}(2+\frac{1}{\epsilon}),\end{aligned}$$

where

$$\gamma'(\epsilon) = \frac{4Yn}{1-z^2} - \frac{Y^2\ln 2}{1-z^2}\frac{1}{\epsilon^2} = 0 \Rightarrow \epsilon^* = \sqrt{\frac{Y\ln 2}{4n}}$$

is chosen to get the tightest bound since $\gamma''(\epsilon) = \frac{2Y^2\ln 2}{1-z^2}\frac{1}{\epsilon^3} > 0, \forall\epsilon > 0$. Hence, the statement in the corollary follows. □

We note that the algorithm given in (2.2), as shown in the corollary, has an error floor $\frac{2Yz^2}{1-z^2}$, which bounds the limit of the time-normalized regret $\lim_{n\to}\infty\frac{1}{n}R_n^*$ from below. This result is due to the non-convexity of the loss function that uses the sigmoid function in parameterization of λ_t. On the other hand, we have certain control over this error floor, which is given here as a function of $0 < z = \frac{1-4\lambda^+(1-\lambda^+)}{1+4\lambda^+(1-\lambda^+)} < 1$. Since $\lim_{\lambda^+\to\frac{1}{2}} z = 0$, and $\lim_{\lambda^+\to 0 \text{ or } 1}\ z = 1$, z controls the size of the competition class $\{\beta\}$, where $\beta \in [\lambda^+, 1-\lambda^+]$. As this class grows, the studied algorithm in this chapter is affected by a larger error floor induced on the time-normalized regret $\frac{1}{n}R_n^*$. Therefore, the algorithm given in (2.2) does not guarantee a diminishing time normalized regret and the bound it

promises is weak when compared to the, for example, Exponentiated Gradient Algorithm [19], whose time normalized regret is $O(n^{\frac{-1}{2}})$.

2.3 Experiments

In this section, we illustrate the performance of the learning algorithm (2.2) and the introduced results through examples. We demonstrate that the upper bound given in (2.4) is asymptotically tight by providing a specific sequence for the desired signal y_t and the outputs of constituent algorithms $\hat{y}_{1,t}$ and $\hat{y}_{2,t}$. We also present a performance comparison between the adaptive mixture and the corresponding best mixture component on a pair of sequences.

In the first example, we present the time-normalized regret $\frac{1}{n}R_n$ of the learning algorithm (2.2) defined in (2.5) and the corresponding upper bound given in (2.4). We first set $Y = 1$, $\lambda^+ = 0.15$ and $\epsilon = 1$. Here, for $t = 1, \ldots, 1000$, the desired signal y_t and the sequences $\hat{y}_{1,t}, \hat{y}_{2,t}$, which the parallel running constituent algorithms produce are given by

$$\hat{y}_{1,t} = Y;\ \hat{y}_{2,t} = (-1)^t Y;\ \text{and } y_t = 0.15\hat{y}_{1,t} + 0.85\hat{y}_{2,t}.$$

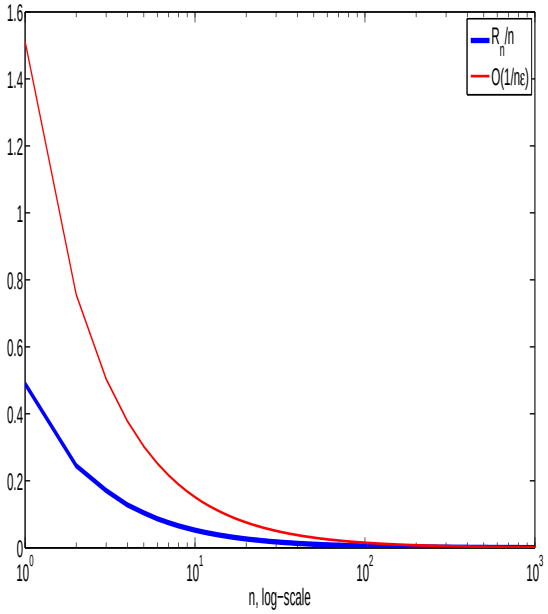
Note that, in this case, the best convex combination weight is $\beta_o = 0.15$. In Fig. 2.1a, we plot the time-normalized regret of the learning algorithm (2.2) "$\frac{1}{n}R_n$" and the upper bound given in (2.4) "$O(1/(n\epsilon))$". From Fig. 2.1a, we observe that the bound introduced in (2.4) is asymptotically tight, i.e., as n gets larger, the gap between the upper bound and the time-normalized regret gets smaller.

In the second example, we demonstrate the effectiveness of the mixture of experts algorithm (2.2) through a comparison between the time-normalized accumulated loss (2.3) of the learned mixture and the one of the best constituent expert. To this end, we design two experiments with $t = 1, \ldots, 10000$, $\lambda^+ = 0.01$, $\epsilon = 0.1$, $Y = \mathrm{e}$, where

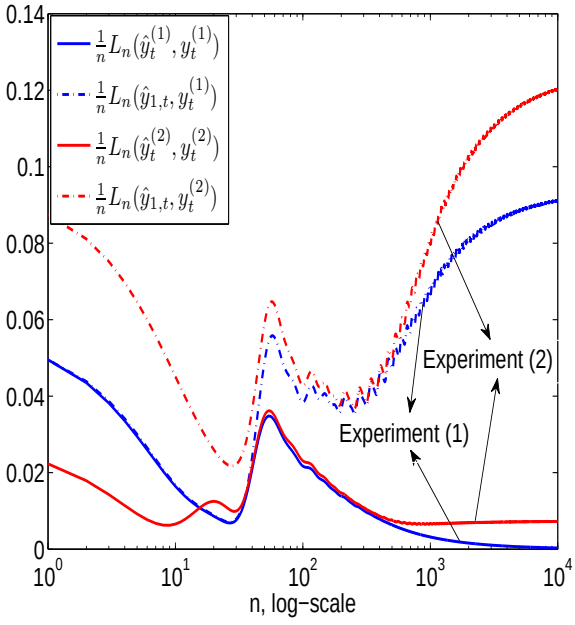
$$\hat{y}_{1,t} = 2e^{-0.005t} - 1, \hat{y}_{2,t} = \sin(0.1t)$$

are chosen as the experts in both of the experiments. In the first experiment, we choose the desired signal as the linear combination $y_t^{(1)} = 0.75\hat{y}_{1,t} + 0.25\hat{y}_{2,t}$, where $\beta_0 = 0.75$. In the second experiment, we choose the desired signal as the nonlinear function of the outputs of the both experts as $y_t^{(2)} = \sin(0.75\hat{y}_{1,t} + 0.25\hat{y}_{2,t})$. Note that the first expert provides a better time-normalized accumulated loss in both cases, i.e., $\frac{1}{n}L_n(\hat{y}_{1,t}, y_t^{(i)}) < \frac{1}{n}L_n(\hat{y}_{2,t}, y_t^{(i)})$. In Fig. 2.1b, we plot the time-normalized accumulated loss of the best (first) expert as well as the one of the mixture returned by the learning algorithm. From Fig. 2.1b, we observe that the adaptive mixture outperforms the best mixture component, i.e., expert 1 in these examples, in both of the cases. Furthermore, the adaptive mixture optimally tunes to the best linear combination in the first case, which is expected since the generation of the desired output is through a linear combination. On the other hand, the adaptive mixture suffers from an error floor, i.e., the time-normalized accumulated loss does not converge to 0, in the second case, since the generation of the desired signal is through a nonlinear transformation.

In this section, we illustrated our theoretical results and the performance of the learning algorithm (2.2) through examples. We observed through an example that the upper bound given in (2.4) is asymptotically tight. We also illustrated the effectiveness of the adaptive mixture on another example by a performance comparison between the mixture and its best component.



(a)



(b)

Figure 2.1: (a) The regret bound derived in Theorem 1. (b) Comparison of the adaptive mixture (2.2) w.r.t. the best expert.

2.4 Discussion

In this chapter, we analyze a learning algorithm [16] that adaptively combines outputs of two constituent algorithms running in parallel to model an unknown desired signal from the perspective of online learning theory and produce results in an individual sequence manner such that our results are guaranteed to hold for any bounded arbitrary signal. We relate the time-accumulated squared estimation error of this algorithm at any time to the time-accumulated squared estimation error of the optimal convex combination of the constituent algorithms that can only be chosen in hindsight. We refrain from making statistical assumptions on the underlying signals and our results are guaranteed to hold in an individual sequence manner. We provide the transient, steady state and tracking analysis of this mixture in a deterministic sense without any assumptions on the underlying signals or without any approximations in the derivations. We illustrate the introduced results through examples.

Since the online algorithm [16] we study in this chapter is based on a fixed ensemble of experts, it is not appropriate in applications, where the data is non-stationary. Moreover, its complexity scales linearly with the size of the ensemble and therefore, it is not practical to increase the ensemble size for improved modeling capabilities. To this end, we propose a novel online algorithm in Chapter 3 whose computational complexity grows in a logarithmic fashion compared to the ensemble size. The proposed algorithm is also significantly more adaptive to non-stationarity since it uses a highly dynamical ensemble, i.e., not fixed. Similarly, we also present theoretical guarantees for the introduced algorithm, which uniformly hold for every possible input sequence.

Chapter 3

Online Classification via Self-Organizing Space Partitioning

In the contemporary machine learning applications [38–41], algorithms are required to process data at an extremely fast rate, yet to learn complex models often in a non-stationary environment. An example is the news recommender application offered by a web site, where millions of user clicks (statement of interest) are typically received every day in addition to the huge volume of daily news. In this example, the web server is required to continuously *classify* the upcoming news on the basis of each user (interested or not) to make real time recommendations. Meanwhile, the learned classification models must also be *updated* simultaneously since the user preferences can change over time, i.e., *concept change* can occur. In addressing this ambitious goal, one generally aims to maximally exploit the information per instance -with only a single access- in the *online* setting and update the most recently learned hypothesis. To this end, we target such big data applications and propose an online algorithm that can learn arbitrarily complex and non-stationary structures with strong performance guarantees. In particular, we propose a novel, highly efficient and effective online classification algorithm for an infinite stream of possibly correlated (labeled)

observations from a possibly non-stationary process.

To learn complex relations while exploiting local regularities, we consider completely adaptive piecewise linear models by partitioning the observation domain, i.e., the feature space, into different regions. Specifically, we use a binary partitioning tree, where a separator (e.g., a hyperplane split or partitioner) and an online linear classifier (a "simple model" such as the perceptron) are assigned to each node/region. The sequential losses of the regional classifiers (i.e., the simple models) are combined into a global loss that is parameterized over separator/split as well as the node/region classifier parameters. We minimize this global loss using the stochastic gradient descent method and obtain the updates for the complete set of tree parameters, i.e., the separators and the region classifiers, at each newly observed instance.[1] The result is a highly dynamical self-organizing decision tree structure that jointly (and in a truly online manner) learns the region classifiers and the optimal feature space partitioning. In this respect, our strategy is highly novel and remarkably robust to drifting source statistics, i.e., non-stationarity. Since our approach is essentially based on a finite combination of linear models, it generalizes well and does not overfit or limitedly overfits [42] (as rigorously shown by our extensive set of experiments).

The introduced partitioning tree effectively defines a class of hierarchical partitions (of the feature space) and a corresponding class of a doubly exponential number ($\sim 1.5^{2^D}$, where D is the depth) of piecewise linear and online base classifiers. The proposed classifier combines the outputs of these base classifiers at each instance and generates its classification output. We prove that without any statistical assumptions, the proposed algorithm asymptotically performs as well as the best base classifier that can only be chosen in hindsight [43]. We point out that each base classifier is a specific union of hierarchically arranged regional linear classifiers. All such possible unions generate the set of the base classifiers and the final combined classifier is in fact an optimal union classifier. Our results hold for every possible input stream regardless of the underlying process that generates the data. The computational complexity of the proposed algorithm is

[1]The node classifiers are updated by their own procedures.

controllable over the depth of the tree and it linearly grows with the depth and the data dimensionality uniformly for all streamed instances.

We perform an extensive set of experiments over both stationary and non-stationary real data to illustrate the performance of our algorithm. For the stationary part, we follow the general comparison framework of [44,45] and show that our approach significantly outperforms (in terms of the average classification error) the state-of-the-art as well as the most recently proposed techniques [6,44–47] on the benchmark data sets used in [44,45]. We also provide more comprehensive and precise comparison results compared to [44, 45], as our experiments are over more trials and repeated for various normalization schemes. In the non-stationary part, we experimentally analyze the proposed approach under the continuous concept drifts and abrupt concept changes[2] [23, 42, 48], where we follow the comparison framework of [23]. We demonstrate that our approach performs comparably with [23] on linearly separable data sets only when [23] is combined with the right expert. Otherwise, our approach achieves a superior adaptation to concept changes/drifts, when one fails to choose the right expert or the class separation is strongly non-linear. Furthermore, our algorithms are significantly more efficient in terms of the computational complexity in comparison to the all compared methods [23, 42, 44–48] as detailed in Section 3.3.

In the literature of classification and regression trees [49, 50], the split criteria are typically chosen a priori and fixed such as the dyadic partitioning [51]; and a specific loss, e.g., the gini index [52], is minimized separately for each node. For instance, multivariate trees are extended to allow the simultaneous use of functional inner and leaf nodes to draw a decision in [53]. Similarly, the node specific individual decisions are combined in [6] via the context tree weighting method [54] and a piecewise linear model for sequential classification is obtained. Since the tree structures in both of these methods are fixed and chosen even before the processing starts, the resulting modeling power is very limited and significantly deteriorates in case of high dimensionality [55]. In contrast, our algorithm provides a theoretically analyzed and computationally efficient fully

[2]Concept drift -in particular- refers to the change in the true labeling of observations, e.g., the change of the user preferences in the aforementioned news recommender application.

adaptive tree for online classification, which learns the split criteria without any limitation or restriction. Thus, our method allows complex partitioning of the feature space without overfitting. This generates a dynamical self-organizing tree that is sequentially tuned to the data and adaptive to high dimensionality. Moreover, we do not use any statistical assumptions regarding the data source unlike [56] and our analyses are guaranteed to hold in a strong mathematical sense [43].

Self-organizing trees have been successfully applied to regression problems in a recent study [57]. However, the computational complexity of the algorithm [57] is exponential in the depth of the tree, whereas our approach operates with significantly smaller computational costs (i.e., linear in the depth of the tree). We also emphasize that both the problem as well as the algorithm in our study is completely different than [57]. Another successful application of the self-organizing trees is presented in [42] in the context of classification. The problem in [42] is formulated in the batch setting, thus it does not address our sequential requirements. Furthermore, the decision tree in [42] grows only from a single branch. However, the introduced algorithm in this chapter efficiently and sequentially manages a complete decision tree. Unlike [42, 58], where the final classifier is described by a single pruning of the tree (i.e., one of the partitions of the feature space defined by the decision tree), we consider the complete set of base classifiers constructed by all possible prunings.

Our approach combines simple models to obtain a strong online classifier, hence is directly related to "boosting" [15,59] and "predicting with expert advice" [14] methods of the online setting. We emphasize that the corresponding online algorithms [20–23,43–48,60–80] essentially consider a weighted linear combination of weak learners/experts that run independently in parallel with no structure. On the contrary, we consider the union of hierarchically structured simple models (or regional linear classifiers) that yields a superior classification performance with minimal computational complexity.

Several online boosting methods are proposed such as the Oza's online algorithm [46] and [60] that are asymptotically convergent to the batch solution of

Adaboost [15]. The corresponding variants [61–66] (specialized for computer vision applications) are heuristically developed with no convergence results. These studies are collected in a single theoretical framework via the stochastic gradient descent in [67]; and their robustness is investigated under noisy labels in [47]. Further analytical justifications along with a proposed smoothed boosting algorithm can be found in [44] (an online extension to [81]). Unlike these stochastic gradient descent based methods, another ensemble method formulated in a Bayesian framework is proposed in [45]. However, all these methods are appropriate only for stationary environments and based on weighted linear combinations of weak learners. In contrast, we consider unions of simple linear models that are adaptively and optimally (to minimize the final loss) located in the feature space with respect to the possible changes in the source statistics, i.e., non-stationary scenarios.

Our theoretical analyses are inline with the "predicting with expert advice" framework [14]. The weighted majority algorithm and similar aggregation strategies are presented in [21,43], which are generalized in [69,71,73] to allow switching experts at a fixed and specified rate and modified for drifting concepts in [72]. The switching rate [69] is also incorporated into the learning algorithm in [70]. A superset of loss functions that can be used in these studies are considered in [20,68]. Generally, the algorithms proposed in this framework, such as [20, 21, 43, 68–73], consider a fixed ensemble of expert algorithms and obtain an optimal weighted linear combination in terms of the regret bounds. Accordingly, we prove that the proposed algorithm asymptotically achieves the performance of the best union of the regional linear classifiers that can only be selected in hindsight. However, we do not restrict our algorithm to a fixed set of models since the non-stationarity has an unpredictable nature and cannot be confined to a fixed set of experts as in these studies [20, 21, 43, 68–73].

A dynamically weighted majority algorithm that can enhance the ensemble by addition or removal of the experts to further adapt to non-stationarity, named as "concept drift" [74], is presented in [23]. This study [23] is theoretically analyzed and further generalized to regression tasks in [22]. Similar ensemble pruning/enhancing techniques are also considered in [48, 75–79] to avoid rigidity of

the fixed set of experts/weak learners approach. In contrast, we follow a different approach: Our algorithm learns the region classifiers and collectively organizes them at every instance. Neither the union structure (partitioning) nor the simple models in our approach are fixed; yet, the proposed algorithm achieves the performance of the optimal union classifier selected in hindsight. In [48, 75–80], concept changes are generally tracked in sliding windows of the stream, which results in incremental (not sequential since batch data is used) learning.[3] In addition, a change detector in the windowed parts is used in [78, 80] to obtain increased adaptivity. On the other hand, the adaptation to concept changes in our work is due to the inherent joint learning process of the split criteria and the region classifiers in our tree. This brings the online processing capability that does not require windowed batch data. The algorithms in the literature of drifting concepts are generally heuristically developed. In this respect, in addition to the introduced error bounds in [23], the presented theoretical analysis in this study provides furthers insight into the non-stationary setting.

After we provide the problem description in Section 3.1, we introduce our sequential classifier and present the theoretical guarantees in Section 3.2. We then demonstrate the performance of our algorithms via extensive real data experiments in Section 3.3. We conclude the chapter with final remarks in Section 3.4.

3.1 Problem Description

3.1.1 Formulation

We study online binary classification, where we observe feature vectors $\{\boldsymbol{x}_t\}_{t\geq 1}$ and determine their labels $\{y_t\}_{t\geq 1}$ in an online manner.[4] In particular, the aim is

[3] A more detailed comparison can be found in our experiments in Section 3.3.

[4] All vectors are column vectors and denoted by boldface lower case letters. Matrices are represented by boldface uppercase letters. For a vector $\boldsymbol{u}$, $\boldsymbol{u}^T$ is the ordinary transpose. Throughout the chapter, the time index appears as a subscript.

to learn a classification function $f_t(\boldsymbol{x}_t)$ with $\boldsymbol{x}_t \in \mathbb{R}^p$ and $y_t \in \{-1, 1\}$ such that when applied in an online manner to any streaming data, the empirical loss of the classifier $f_t(\cdot)$, i.e.,

$$L_T(f_t) \triangleq \sum_{t=1}^{T} \mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}, \tag{3.1}$$

is asymptotically as small as the empirical loss of the best classifier $C(\boldsymbol{\phi})$ from a competition class $\mathcal{S}(\boldsymbol{\phi})$ of base classifiers for any sequence length T (where T is not a design parameter, i.e., our algorithms are truly sequential). The set of classifiers $\mathcal{S}(\boldsymbol{\phi})$ is a parameter dependent class that can be optimized over $\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is not a specific algorithm dependent parameter such as the separation hyperplane of a linear classifier, but instead it determines the "shape" of the competition class.[5] Unlike the relevant works in the literature of "predicting with expert advice" [14], the goal, in this chapter, is not only to achieve the performance of the best expert, i.e., the best base classifier, but also to optimize the competition class $\mathcal{S}(\boldsymbol{\phi})$ over the "shape" $\boldsymbol{\phi}$ to further and directly minimize the final error.

To be more precise, we measure the relative performance of f_t with respect to the performance of a base classifier[6] $f_t^{(C(\boldsymbol{\phi}))}$, where $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$, using the following regret

$$R_T(f_t; f_t^{(C(\boldsymbol{\phi}))}) \triangleq \frac{L_T(f_t) - L_T(f_t^{(C(\boldsymbol{\phi}))})}{T}, \tag{3.2}$$

for any arbitrary length T. Our aim is then *i)* to construct an online algorithm with guaranteed upper bounds on this regret for any base classifier and *ii)* to optimize over $\boldsymbol{\phi}$ in order to minimize the classification error. In this sense, the proposed algorithm f_t competes against the best base classifier that itself constantly improves.

We emphasize that a classifier $C(\boldsymbol{\phi})$ in the competition class, i.e., $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$, can be implemented in various ways. In this chapter, we consider the union classifier, which can approximate any arbitrarily nonlinear class separation

[5]The precise meaning of the parameter $\boldsymbol{\phi}$ will become clear shortly.

[6]We use the notation $f_t^{(C(\boldsymbol{\phi}))}$ to precisely denote the actual operational and online base classifier $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$.

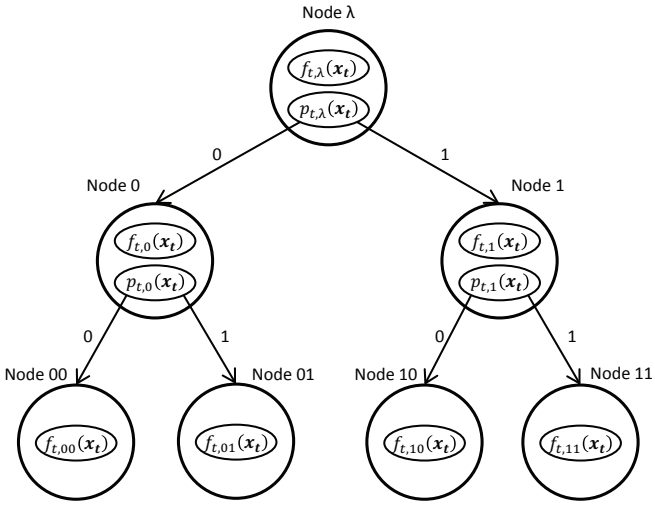


Figure 3.1: The generalized view of the complete tree structure. $f_{t,n}(\cdot)$ represents the classifier of node n and $p_{t,n}(\cdot)$ represents the separator function corresponding to node n, cf. (3.3).

by piecewise linear curves with limited (or without) overfitting, cf. the finite VC dimension discussion in [42]. To efficiently construct this set of base classifiers that is dynamically adaptive to the data through the shape $\boldsymbol{\phi}$, we introduce a self-organizing partitioning tree in the following section.

3.1.2 Adaptive Space Partitioning with Self-Organizing Trees

A tree -in our work- defines a nested partitioning of the feature space at each node of which, we have a simple region/local classifier (e.g., a linear and online classifier such as the perceptron) and a separator/partitioner (or a split) of the corresponding region. A generalized view of a depth-2 tree is given in Fig. 3.1, where $f_{t,n}$ represents the region classifier and $p_{t,n}$ represents the separator function of node n at time t. In Fig. 3.1, a depth-2 tree is used to partition the feature space as follows. The root node (or node λ) represents the entire feature space, where the separator function $p_{t,\lambda}$ bisects this region and creates node 0 and node 1. Similarly, each of these nodes are also split via $p_{t,0}$ and $p_{t,1}$ creating children nodes 00, 01 and 10, 11, respectively. We point out that the selection of the node

classifiers and separator functions are completely up to preference and can be arbitrary. However, we use perceptrons as our node classifiers and hyperplanes as our node separators. The separator $p_{t,n}$ is a function of $\boldsymbol{\phi}_{t,n}$ such as the sigmoid function

$$p_{t,n}(\boldsymbol{x}_t) = \frac{1}{1 + e^{\boldsymbol{\phi}_{t,n}^T \boldsymbol{x}_t}}, \tag{3.3}$$

where $\boldsymbol{\phi}_{t,n}$ represents the angle of the normal line to the separating hyperplane for each node n. We consider these differentiable separator functions as randomized decision blocks such that $p_{t,n}(\boldsymbol{x}_t)$ represents the probability of assigning $\boldsymbol{x}_t$ to the right child node of n.

As an example, the 2-dimensional feature space is partitioned using a depth-2 tree in Fig. 3.2. Operationally, each instance $\boldsymbol{x}_t$ is propagated from the root node to a leaf node through a certain branch such that if $\boldsymbol{\phi}_{t,n}^T \boldsymbol{x}_t \geq 0$ (cf. (3.3)) at node n, then $\boldsymbol{x}_t$ follows the 1-branch; otherwise, it follows the 0-branch. Meanwhile, at each visited node, it is classified by the local node (region) classifier. In Fig. 3.2, the dashed lines represent the partitioning of the feature space corresponding to each inner node and the two different colored regions represent the node classifier outputs at the respective regions. Each complete subtree (or pruning) generates a certain partition with a certain piecewise linear classification structure and hence, yields a complete base classifier. According to the partitioning in Fig. 3.2, 5 different base classifiers producing the competition class can be defined and these classifiers are presented in Fig. 3.3. Note that for a base classifier $C(\boldsymbol{\phi}) \in \boldsymbol{\mathcal{S}}(\boldsymbol{\phi})$, the output $f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t)$ makes its decision according to the decision of the region classifier $f_{t,n}(\boldsymbol{x}_t)$, where n is the leaf node (containing $\boldsymbol{x}_t$) of the subtree that generates $C(\boldsymbol{\phi})$.

Based on this tree partitioning, for a depth-D tree, there exist approximately 1.5^{2^D} different base classifiers [82]. The proposed classifier f_t combines the outputs of all these base classifiers at each time and generates its final output with the desired competitive classification performance. Namely, we achieve a diminishing regret in (3.2) for any base classifier such that the performance of the best base classifier is matched. Note that each base classifier is an online union classifier as it operates on a union of the regions spanning the entire feature space

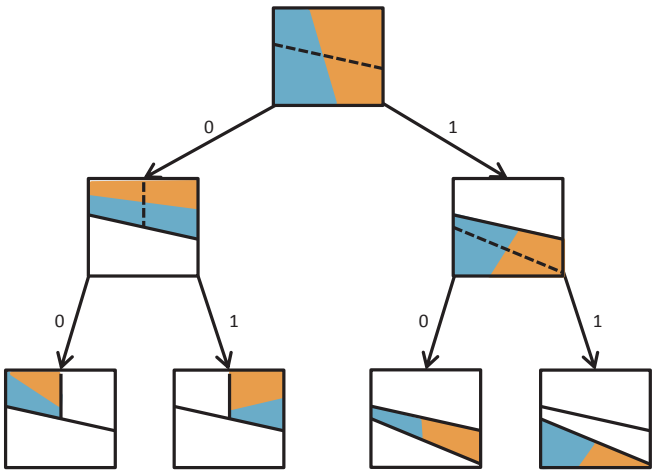


Figure 3.2: The partitioning of a 2-dimensional feature space using a depth-2 tree. The whole feature space is first bisected by $p_{t,n=\lambda}$ and split into two regions $n = 0$ and $n = 1$: if an instance $\boldsymbol{\phi}_{t,n=\lambda}^T \boldsymbol{x}_t \geq 0$ (or if $p_{t,n=\lambda}(\boldsymbol{x}_t) \geq 0.5$ in (3.3)), then it follows the 1-branch; otherwise, it follows the 0-branch. The corresponding regions are similarly bisected by $p_{t,0}$ and $p_{t,1}$, respectively. The active region at a node resulting from the previous splits is illustrated colored, where the dashed line represents the separating hyperplane (whose normal vector is $\boldsymbol{\phi}_{t,n}$) at that node and the two differently colored subregions are the corresponding local classification by $f_{t,n}$.

with simple and online linear region classifiers at each region.

In addition to the goal of obtaining the competitive performance, we also aim to constantly improve this competitive performance by directly improving the performances of the base classifiers (i.e., competitors) over time through jointly learning the optimal region classifiers as well as the optimal partitioning structure. Hence, the proposed algorithm f_t is competitive against a competition class that itself is designed to constantly improve over time. To this end, we parameterize the introduced tree over the collection of the separator function parameters $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{t,n}\}$, which also defines the aforementioned "shape" of our competition class $\mathcal{S}(\boldsymbol{\phi})$. Then, we directly minimize the resulting final classification loss of f_t in (3.1) over the complete set of tree parameters. We refer this structure as self-organizing tree.

In this framework, we emphasize two points. First, there is a $1-1$ mapping

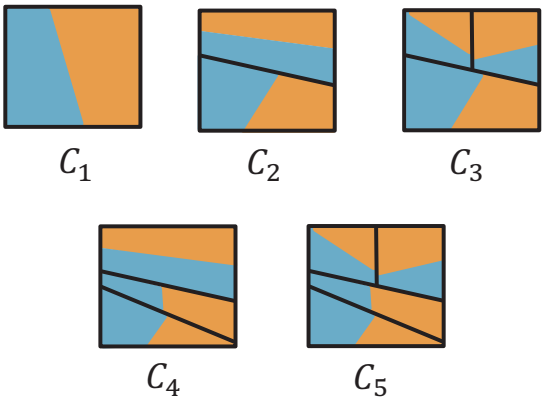


Figure 3.3: The competition class of base classifiers defined by the depth-2 tree in Fig. 3.2. Each of these base classifiers corresponds to a complete subtree in Fig. 3.2.

between the partitions and the base classifiers, and we use these phrases interchangeably: $C(\phi)$ is referred as a partition or a base classifier conveniently for clarity. However, we use the notation $f_t^{(C(\phi))}$ to precisely denote the actual operational and online base classifier. Second, these partitions, the simple region classifiers and hence the resulting base classifiers are all time varying due to our online setting. In this setting, our goal is to find a computationally efficient sequential algorithm that achieves the performance of the optimal base classifier (cf. (3.2)) and also simultaneously improves that optimal base classifier.

3.2 The Adaptive (Self-Organizing) Union Classifier

In this section, we propose two variants of our online classifier f_t. 1) AUC.Rnd: f_t randomly chooses a base classifier using a specific weighting scheme over the class $\mathcal{S}(\phi)$ and matches the output of the chosen base classifier; and 2) AUC.Avg: f_t outputs the weighted average of the base classifier decisions. Here, AUC stands for "Adaptive and Universal Union Classifier".

In our theoretical analysis, we concentrate on the algorithm AUC.Rnd; however, our results hold for AUC.Avg as well in a straightforward manner. We prove that the performance of the proposed algorithm f_t, i.e., AUC.Rnd, is asymptotically as well as the best base classifier, where the adaptation of $\boldsymbol{\phi}$ results significant performance gains as it directly improves the competitors, i.e., the best base classifier. In particular, we provide an upper bound on the regret $R_T(f_t; f_t^{(C(\boldsymbol{\phi}))})$ such that as $T \to \infty$, $R_T(f_t; f_t^{(C(\boldsymbol{\phi}))}) \to 0$ for this highly dynamical self-organizing tree. We provide the construction of the algorithm (and also the detailed construction of the base classifiers) in the proof of the following theorem, where we also present our theoretical results.

Theorem 2: *Let $\{\boldsymbol{x}_t\}_{t\geq 1}$ and $\{y_t\}_{t\geq 1}$ be arbitrary and real-valued sequence of feature vectors and their labels, respectively. The universal randomized union classifier presented in Alg. 1, i.e., AUC.Rnd, when applied to these data sequences, sequentially yields*

$$\max_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} E\left[R_T(f_t; f_t^{(C(\boldsymbol{\phi}))})\right] \leq O\left(\sqrt{\frac{2^D}{T}}\right), \tag{3.4}$$

for all T with a computational complexity $O(Dp)$, where p represents the dimensionality of the feature vectors and the expectation is with respect to the randomization parameters.

Proof of Theorem 2 and Construction of Algorithm AUC.Rnd: We start the proof by explicitly constructing the base classifiers. We next introduce a low complexity method to achieve the best classifier among doubly exponential number of different base classifiers. Then, we incorporate an adaptive method to optimize $\boldsymbol{\phi}$ in order to minimize the classification of the final algorithm.

Before proceeding, we first introduce the following notation. For ease of exposition in specifying the nodes, each node of the tree is labeled with a binary string $n = m_1 \dots m_d$, where $m_i = \{0, 1\}$ is a binary letter and d represents the depth of the node. For any inner node n, we label its left and right children as $n0$ and $n1$, respectively. We denote the empty string by λ. Moreover, we call a node

$n' = m'_1 \dots m'_{d'}$ as the prefix of node $n = m_1 \dots m_d$ if $d' \leq d$ and $m'_i = m_i$ for all $i = 1, \dots, d'$. Using this definition, we denote n_i as the depth-i prefix to node n, where $i = \{0, \dots, d\}$. This labeling operation can be observed for a depth-2 tree in Fig. 3.1.

According to the partitioning method described in Section 3.1.2, the output of a base classifier $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$ is softly constructed using the partitioning functions $p_{t,n}$ as follows. Without loss of generality, suppose that the instance $\boldsymbol{x}_t$ has fallen into the region represented by the leaf node n. Then, $\boldsymbol{x}_t$ is contained in the nodes $n_0, \dots, n_D$, where $n_D = n$ and $n_0 = \lambda$. For example, if node n_d is a leaf node of the subtree generating the base classifier $C(\boldsymbol{\phi})$, then one can simply set $f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$ as done in many prior work, cf. [6, 49, 50, 53, 54, 56, 83].

Instead of making such a hard selection, we allow an error margin for the classification output $f_{t,n_d}(\boldsymbol{x}_t)$ in order to be able to update the region boundaries later in the proof. To achieve this, for each leaf node of $C(\boldsymbol{\phi})$, we define a parameter called "path probability" to measure the contribution of each leaf node to the classification task at time t. This parameter is equal to the multiplication of the partitioning functions of the nodes from the respective leaf node to the root node, which represents the probability that $\boldsymbol{x}_t$ should be classified using the region classifier of node n_d. This path probability is defined as follows

$$P_{t,n_d}(\boldsymbol{x}_t) \triangleq \prod_{i=0}^{d-1} p_{t,n_i,m_{i+1}}(\boldsymbol{x}_t), \tag{3.5}$$

where $p_{t,n_i,m_{i+1}}(\cdot)$ represents the value of the partitioning function corresponding to node n_i towards the m_{i+1} direction, i.e.,

$$p_{t,n_i,m_{i+1}}(\boldsymbol{x}_t) \triangleq \begin{cases} p_{t,n_i}(\boldsymbol{x}_t) & \text{, if } m_{i+1} = 0 \\ 1 - p_{t,n_i}(\boldsymbol{x}_t) & \text{, if } m_{i+1} = 1 \end{cases},$$

with $p_{t,n_i}(\boldsymbol{x}) = \left[1 + \exp\left(\boldsymbol{\phi}_{t,n_i}^T \boldsymbol{x}\right)\right]^{-1}$ as in (3.3). We consider that the classification output of node n_d can be trusted with a probability of $P_{t,n_d}(\boldsymbol{x}_t)$. Intuitively, the path probability is low when the feature vector is close to the region boundaries, hence we may consider to classify that feature vector by another node classifier (e.g., the classifier of the sibling node). Using this path probabilities,

we aim to update the region boundaries by learning whether an efficient node classifier is used to classify $\boldsymbol{x}_t$, instead of directly assigning $\boldsymbol{x}_t$ to node n_d and lose a significant degree of freedom. To this end, we define the final output of each node classifier according to a Bernoulli random variable with outcomes $\{-f_{t,n_d}(\boldsymbol{x}_t), f_{t,n_d}(\boldsymbol{x}_t)\}$ where the probability of the latter outcome is $P_{t,n_d}(\boldsymbol{x}_t)$. Although the final classification output of node n_d is generated according to this Bernoulli random variable, we continue to call $f_{t,n_d}(\boldsymbol{x}_t)$ the final classification output of node n_d, with an abuse of notation. Then, the classification output of the base classifier is set to $f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$.

After constructing all base classifiers, we use a mixture-of-experts approach to achieve the performance of the best base classifier that minimizes the accumulated classification error. Before presenting this method, we first introduce certain definitions. Let the instantaneous empirical loss of the proposed classifier f_t at time t be denoted by

$$\ell_t(f_t) \triangleq \mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}.$$

Then, the expected empirical loss of this classifier over a sequence of length T can be found by

$$L_T(f_t) = E\left[\sum_{t=1}^{T} \ell_t(f_t)\right], \tag{3.6}$$

with the expectation taken with respect to the randomization parameters of the classifier f_t. We also define the effective region of each node n_d at time t as follows

$$\mathcal{R}_{t,n_d} \triangleq \left\{\boldsymbol{x} : P_{t,n_d}(\boldsymbol{x}) \geq (0.5)^d\right\}.$$

Note that according to the aforementioned structure of base classifiers, node n_d classifies an instance $\boldsymbol{x}_t$ only if $\boldsymbol{x}_t \in \mathcal{R}_{t,n_d}$. Therefore, the time accumulated empirical loss of any node n during the data stream is given by

$$L_{T,n} \triangleq \sum_{t \leq T : x_t \in \mathcal{R}_{t,n}} \ell_t(f_{t,n}). \tag{3.7}$$

Similarly, the time accumulated empirical loss of a base classifier $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$ is found as

$$L_T^{(C(\boldsymbol{\phi}))} \triangleq \sum_{n \in \mathcal{L}(C(\boldsymbol{\phi}))} L_{T,n},$$

where $\mathcal{L}(C(\boldsymbol{\phi}))$ is the set of the leaf nodes of the subtree generating $C(\boldsymbol{\phi})$.

Using these definitions, we first introduce a direct and inefficient implementation of the mixture-of-experts approach as follows. We set the final classification output of our algorithm as $f_t(\boldsymbol{x}_t) = f_t^{(C(\boldsymbol{\phi}))}$ with probability $w_t^{(C(\boldsymbol{\phi}))}$, where $w_t^{(C(\boldsymbol{\phi}))} = 2^{-J(C(\boldsymbol{\phi}))} \exp\left(-b\, L_{t-1}^{(C(\boldsymbol{\phi}))}\right) / Z_{t-1}$, and prove that we can achieve the upper bound in (3.4) with these weights. Here, $b \geq 0$ is a constant controlling the learning rate of the algorithm, $J(C(\boldsymbol{\phi})) \leq 2|\mathcal{L}(C(\boldsymbol{\phi}))| - 1$ represents the number of bits required to code the classifier $C(\boldsymbol{\phi})$ (which satisfies $\sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} J(C(\boldsymbol{\phi})) = 1$), and $Z_t = \sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} 2^{-J(C(\boldsymbol{\phi}))} \exp\left(-b\, L_t^{(C(\boldsymbol{\phi}))}\right)$ is the normalization factor.

According to the definition of Z_t, the normalization parameter at the last iteration (i.e., the iteration at time T) satisfies

$$-\frac{1}{b}\log Z_T \leq L_T^{(C(\boldsymbol{\phi}))} + \frac{J(C(\boldsymbol{\phi}))\log 2}{b}, \tag{3.8}$$

$\forall C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$. We then make the following observation

$$\begin{aligned} Z_T &= \prod_{t=1}^{T} \frac{Z_t}{Z_{t-1}} \\ &= \prod_{t=1}^{T} \Bigg\{ \sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} \frac{2^{-J(C(\boldsymbol{\phi}))} \exp\left(-b\, L_{t-1}^{(C(\boldsymbol{\phi}))}\right)}{Z_{t-1}} \\ &\qquad\qquad \times \exp\left(-b\, \ell_t(f_t^{(C(\boldsymbol{\phi}))})\right) \Bigg\} \\ &\leq \exp\left(-b\, L_T(f_t) + \frac{Tb^2}{8}\right), \end{aligned} \tag{3.9}$$

where the second line follows from the definition of Z_t and the last line follows from the Hoeffding's inequality [84] by treating the $w_t^{(C(\boldsymbol{\phi}))} \triangleq 2^{-J(C(\boldsymbol{\phi}))} \exp\left(-b\, L_{t-1}^{(C(\boldsymbol{\phi}))}\right) / Z_{t-1}$ terms as the randomization probabilities. Note that $L_T(f_t)$ represents the expected loss of the final algorithm, cf. (3.6). Combining (3.8) and (3.9), we obtain

$$\frac{L_T(f_t)}{T} \leq \frac{L_T^{(C(\boldsymbol{\phi}))}}{T} + \frac{J(C(\boldsymbol{\phi}))\log 2}{Tb} + \frac{b}{8},$$

and choosing $b = \sqrt{2^D/T}$, we find the desired upper bound in (3.4) since $J(C(\boldsymbol{\phi})) \leq 2^{D+1} - 1$, $\forall C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$.

Although we achieve the desired upper bound in (3.4) with this randomization method, the final algorithm f_t -in its current form- requires a computational complexity $O(1.5^{2^D}p)$ since the randomization $w_t^{(C(\boldsymbol{\phi}))}$ is performed over the set $\mathcal{S}(\boldsymbol{\phi})$ and $|\mathcal{S}(\boldsymbol{\phi})| \approx 1.5^{2^D}$. However, the set of all possible classification decisions has a cardinality as small as $D+1$ since $\boldsymbol{x}_t \in \mathcal{R}_{t,n_D}$ some leaf node n_D and $f_t^{(C(\boldsymbol{\phi}))} = f_{t,n_d}$ for some $d = 0, \ldots, D$, $\forall C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$. Hence, evaluating all the base classifiers in $\mathcal{S}(\boldsymbol{\phi})$ at the instance $\boldsymbol{x}_t$ to produce $f_t(\boldsymbol{x}_t)$ is unnecessary. In fact, the computational complexity for producing $f_t(\boldsymbol{x}_t)$ can be reduced from $O(1.5^{2^D}p)$ to $O(Dp)$ by performing the exact same randomization over f_{t,n_d}'s using the new set of weights w_{t,n_d}, which can be straightforwardly derived as

$$w_{t,n_d} = \sum_{C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi}) \,:\, f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)} w_t^{(C(\boldsymbol{\phi}))}. \tag{3.10}$$

To efficiently calculate (3.10) with complexity $O(Dp)$, we consider the universal coding scheme and let

$$M_{t,n} \triangleq \begin{cases} \exp\left(-bL_{t,n}\right) & \text{, if } n \text{ has depth } D \\ \frac{1}{2}\Big[M_{t,n0}M_{t,n1} + \exp\left(-bL_{t,n}\right)\Big] & \text{, otherwise} \end{cases} \tag{3.11}$$

for any node n and observe that we have $M_{t,\lambda} = Z_t$ [85]. Therefore, we can use the recursion (3.11) to obtain the denominator of the randomization probabilities $w_t^{(C(\boldsymbol{\phi}))}$. To efficiently calculate the nominator of (3.10), we introduce another intermediate parameter as follows. Letting n'_d denote the sibling of node n_d, we recursively define

$$\kappa_{t,n_d} \triangleq \begin{cases} \frac{1}{2} & \text{, if } d = 0 \\ \frac{1}{2} M_{t-1,n'_d}\, \kappa_{t,n_{d-1}} & \text{, if } 0 < d < D \\ M_{t-1,n'_d}\, \kappa_{t,n_{d-1}} & \text{, if } d = D \end{cases}, \tag{3.12}$$

$\forall d \in \{0, \ldots, D\}$, where $\boldsymbol{x}_t \in \mathcal{R}_{t,n_D}$. Using the intermediate parameters in (3.11) and (3.12), it can be shown that we have

$$w_{t,n_d} = \frac{\kappa_{t,n_d} \exp\left(-b\, L_{t,n_d}\right)}{M_{t,\lambda}}. \tag{3.13}$$

Hence, we can obtain the final output of the algorithm as $f_t(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$ with probability w_{t,n_d}, where $d \in \{0, \ldots, D\}$ (i.e., with a computational complexity $O(D)$).

We then use the final output of the introduced algorithm and update the region boundaries of the tree (i.e., organize the tree) to minimize the final classification error. To this end, we first note that $E[\mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}] = \frac{1}{4} E\left[(y_t - f_t(\boldsymbol{x}_t))^2\right]$ and then use the stochastic gradient descent method as follows

$$\begin{aligned}
\boldsymbol{\phi}_{t+1,n_d} &= \boldsymbol{\phi}_{t,n_d} - \eta \nabla E\left[\ell_t(f_t)\right] \\
&= \boldsymbol{\phi}_{t,n_d} - (-1)^{m_{d+1}} \eta \left(y_t - f_t(\boldsymbol{x}_t)\right) p_{t,n_d,m'_{d+1}}(\boldsymbol{x}_t) \\
&\qquad \times \left[\sum_{i=d+1}^{D} f_{t,n_i}(\boldsymbol{x}_t)\right] \boldsymbol{x}_t, \qquad (3.14)
\end{aligned}$$

$\forall d \in \{0, \ldots, D-1\}$, where η denotes the learning rate of the algorithm and m'_{d+1} represents the complementary letter to m_{d+1} from the binary alphabet $\{0, 1\}$. Defining a new intermediate variable

$$\pi_{t,n_d} \triangleq \begin{cases} f_{t,n_d}(\boldsymbol{x}_t) & \text{, if } d = D-1 \\ \pi_{t,n_{d+1}} + f_{t,n_d}(\boldsymbol{x}_t) & \text{, if } d < D-1 \end{cases}$$

one can perform the update in (3.14) with a computational complexity $O(p)$ for each node n_d, $d = 0, \ldots, D-1$, resulting in a overall computational complexity of $O(Dp)$ as follows

$$\begin{aligned}
\boldsymbol{\phi}_{t+1,n_d} = \boldsymbol{\phi}_{t,n_d} &- (-1)^{m_{d+1}} \eta \left(x_t - f_t(\boldsymbol{x}_t)\right) \pi_{t,n_d} \\
&\times p_{t,n_d,m'_{d+1}}(\boldsymbol{x}_t)\ \boldsymbol{x}_t. \qquad (3.15)
\end{aligned}$$

Note that in (3.15), both $p_{t,n_d}(\boldsymbol{x}_t)$ and $1 - p_{t,n_d}(\boldsymbol{x}_t)$ terms appear in the product, which can disturb the learning rate of the algorithm if $p_{t,n_d}(\boldsymbol{x}_t)$ is close to 0 or 1. Therefore, in order to avoid such a scenario, using a small positive constant $p_{lim} > 0$, the partitioning function can be restricted to $[p_{lim}, 1 - p_{lim}]$, i.e., $0 < p_{lim} \leq p_{t,n_d}(\boldsymbol{x}_t) \leq 1 - p_{lim}$, by the following

$$p_t(\boldsymbol{x}_t) = p_{lim} + (1 - 2p_{lim}) \frac{1}{1 + e^{\boldsymbol{\phi}_t^T \boldsymbol{x}_t}}.$$

Algorithm 1 Adaptive Universal Union Classifier (AUC.Rnd)

for $t = 1$, $t \leq T$ **do**
  Propagate x_t from the root to the leaf and obtain the visited nodes $n_0, \ldots, n_D$.
  Calculate $P_{t,n_d}(\boldsymbol{x}_t)$ for all $d \in 0, ..., D$ using (3.5).
  Calculate $w_{t,n_d}(\boldsymbol{x}_t)$ for all $d \in 0, ..., D$ using (3.13).
  Draw a node among $n_0, \ldots, n_D$ with probabilities $w_{t,n_0}, \ldots, w_{t,n_D}$, respectively; suppose that n_d is drawn.
  Draw a classification output $\{1, -1\}$ with probabilities $P_{t,n_d}(\boldsymbol{x}_t)$ and $1 - P_{t,n_d}(\boldsymbol{x}_t)$, respectively; $f_t(\boldsymbol{x}_t)$ is equated to the selected output.
  Update the region classifiers (perceptron) at the visited nodes [86].
  $\ell_t(f_t) \leftarrow \mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}$
  Update L_{t,n_d} for all $d \in 0, ..., D$ using (3.7).
  Apply the recursion in (3.11) to update M_{t+1,n_d} for all $d \in 0, ..., D$.
  Update the separator parameters $\boldsymbol{\phi}$ using (3.15).
end for

This concludes the proof of Theorem 2 and the pseudocode of the algorithm AUC.Rnd can be found in Algorithm 1. □

Remark: We emphasize that the sharpness of the partitioning function can be arbitrarily set by defining

$$p_t(\boldsymbol{x}_t) = \frac{1}{1 + e^{s\boldsymbol{\phi}_t^T \boldsymbol{x}_t}},$$

where $s > 0$ denotes the sharpness parameter. Therefore, we have

$$\lim_{s \to \infty} p_t(\boldsymbol{x}_t) = u(-\boldsymbol{\phi}_t^T \boldsymbol{x}_t),$$

where $u(\cdot)$ represents the unit step function. Hence, we conclude that for sufficiently large s, the probabilities of the Bernoulli random variable (which is drawn to find the final output of each node classifier) are found as $\{\delta, 1 - \delta\}$ for the outcomes $\{-f_{t,n}(\boldsymbol{x}_t), f_{t,n}(\boldsymbol{x}_t)\}$, respectively, where $\delta > 0$ is a small number. We emphasize that this approximation can be made arbitrarily accurate by increasing s. Furthermore, through our simulations, we illustrate that this approximation results in negligible overhead even for $s = 10$.

Remark: Instead of randomly selecting a base classifier and repeating its output to generate the final decision, the same randomization probabilities can be used as weighting factors. In this case, the outputs of all base classifiers are combined and our results hold in a similar expectation sense. We denote this classifier AUC.Avg, cf. Section 3.3.

3.3 Experiments

In this section, we demonstrate the performance of the proposed algorithms through several experiments in three separate parts. In the first part, we concentrate on stationary cases, where the source statistics are stationary over time. We show that in this case, our algorithm successfully combines simple classification models and significantly outperforms the most recent ensemble techniques [6, 44–47]. We then study non-stationary cases and illustrate the adaptation power of our algorithm to concept changes/drifts with respect to the state-of-the-art approaches [23,42,48]. In the final part, we present the average running times of the compared methods.

3.3.1 Stationary Data

In this part, we study our algorithms in the stationary environments when the data source statistics do not change over time, and compare the empirical performance of our methods with most recently proposed ensemble techniques and classification trees. In particular, we follow the comparison framework of [44] that is also used in [45]. However, we opt not to include a comparison with the proposed method in [45], which does not fairly fit to our comparison framework since each weak learner in [45] uses a randomized (different) subset of data attributes and that randomization is also not clearly provided. In the following, we summarize the compared methods.

Data Sets (Size/Dimension/Trials)	Perceptron	OZAB	OGB	OSB	OSB.Ocp	UC.Rnd	AUC.Rnd	AUC.Avg
First Row: Our Results with Normalized Data, i.e., each attribute is linearly mapped to $[-1, 1]$								
Second Row: Our Results with Truncated Data, i.e., $x_t \leftarrow \frac{x_t}{\max(\|x_t\|, 1)}$								
Third Row: Previously Reported Results (over 5 Trials) with Truncated Data								
Heart (270/13/100)	0.2466	0.2396	0.2328	0.2363	0.2437	0.2175	0.2186	**0.2009***
	0.2452	0.2400	0.2314	0.2317	0.2343	0.2395	0.2372	**0.2083**
	0.2489	0.2356	0.2267	0.2356	0.2311	-	-	-
Breast Cancer (683/10/100)	0.0577	0.0544	0.0571	0.0523	0.0536	0.0484	0.0486	**0.0465***
	0.0590	0.0538	0.0533	0.0490	0.0504	0.0499	0.0475	**0.0458***
	0.0592	0.0501	0.0445	0.0466	0.0515	-	-	-
Australian (693/14/100)	0.2082	0.2026	0.1970	0.2001	0.2055	0.1592	0.1577	**0.1486***
	0.2073	0.2010	0.1931	0.1900	0.1943	0.1695	0.1713	**0.1539***
	0.2099	0.2012	0.1962	0.1872	0.2078	-	-	-
Diabetes (768/8/100)	0.3225	0.3243	0.3349	0.3133	0.3155	0.2689	0.2772	**0.2575***
	0.3240	0.3258	0.3335	0.3117	0.3130	0.2875	0.2933	**0.2728**
	0.3216	0.3169	0.3313	0.3185	0.3315	-	-	-
German (1000/24/100)	0.3245	0.3186	0.3272	0.3186	0.3221	0.2813	0.2790	**0.2674***
	0.3240	0.3212	0.3241	0.3137	0.3153	0.2861	0.2837	**0.2715***
	0.3256	0.3364	0.3142	0.3148	0.3174	-	-	-
BMC (1200/2/100)	0.4709	0.4572	0.4692	0.4637	0.4658	0.2537	0.1833	**0.1703**
	0.4808	0.4808	0.4869	0.4802	0.4819	0.3451	0.2683	**0.2507**
	-	-	-	-	-	-	-	-
Splice (3175/60/100)	0.3342	0.3259	0.3279	0.3281	0.3293	0.1888	0.1886	**0.1856**
	0.2728	0.2686	0.2643	0.2567	0.2565	0.2198	0.2111	**0.2081**
	0.2717	0.2759	0.2625	0.2605	0.2590	-	-	-
Banana (5300/2/100)	0.4891	0.4796	0.4800	0.4884	0.4882	0.2798	0.1823	**0.1760**
	0.4900	0.4807	0.4827	0.4897	0.4893	0.2784	0.1904	**0.1831**
	-	-	-	-	-	-	-	-
Mushrooms (8124/112/100)	0.0174	**0.0089**	0.0180	0.0160	0.0140	0.0104	0.0108	0.0101
	0.0136	0.0064	0.0075	**0.0063**	0.0065	0.0178	0.0138	0.0129
	0.0148	0.0080	0.0068	0.0060	0.0062	-	-	-
Adult (48842/122/10)	0.2098	0.2079	0.2061	0.2062	0.2056	**0.1535**	0.1540	0.1536*
	0.2089	0.2049	0.2079	0.1986	0.1988	0.2234	**0.1560**	0.1566*
	0.2093	0.2045	0.2080	0.1994	0.1991	-	-	-
Cod-Rna (488565/8/10)	0.3527	0.3641	0.3676	0.3468	0.3458	0.0482	**0.0481**	0.0483*
	0.1899	0.2193	0.1862	0.1826	0.1831	**0.1263**	0.1265	0.1265
	0.2096	0.2170	0.2241	0.2075	0.2059	-	-	-
Cover-Type (581012/54/10)	0.3425	0.3499	0.3496	0.3361	0.3361	**0.2447**	0.2451	0.2450*
	0.3436	0.3454	0.3472	0.3326	0.3327	0.2455	0.2455	**0.2455***
	0.3437	0.3449	0.3482	0.3334	0.3338	-	-	-

Figure 3.4: The proposed algorithms significantly outperform the state-of-the-art techniques based on the average error rates on several real data sets.

1. *Online AdaBoost* - "OZAB" [46]: This method is proven to asymptotically match the AdaBoost algorithm [15] in the online setting when used with lossless weak learners, which uses Poisson sampling to obtain a weighting scheme that evolves over time. An example is presented Poisson(λ) times to a weak learner and if there is still an error after Poisson(λ) trials, it is passed to the next learner with an increased λ; otherwise (if no error in the end), it is passed with a decreased λ. Parameters are set as in [46], i.e., $\lambda = 1$ initially, uniformly in all of our experiments.

2. *Online GradientBoost* - "OGB" [47]: This method is originally proposed to make the boosting more robust to label noise. The idea is to use the functional gradient descent approach to obtain an optimal set of weak learner weights based on the choice of the loss function such as $0-1$, Exponential, Logit and Hinge loss. It uses K weak learner per M selectors, which essentially results in MK weak learners in total. As in [44], we use $K = 1$ for a fair comparison along with the logit loss that has been shown to consistently outperform other choices in [47]. Parameters are set as in [47], i.e., $w = logit'(0)$ initially, uniformly in all of our experiments.

3. *Online SmoothBoost* - "OSB" [44]: This method is the online extension to [81], where the weak learners are assumed to perform well only with respect to a smooth data distribution instead of (typically) assuming that they perform better than the random guess for every possible data distribution. Then, the example weights during the online training are obtained based on this new assumption. Parameters are set as in [44] uniformly in all of our experiments, i.e., $\gamma = 0.1$ and learner weights are uniform ($\alpha = 1/N$) throughout all iterations and examples.

4. *Online SmoothBoost with Online Convex Programming* - "OSB.Ocp" [44]: Instead of a uniform choice over the weak learners, an extra projection step into the probability simplex is added to the OSB algorithm via an online convex programming and a new algorithm OSB.Ocp is obtained in [44] using the predicting with expert advice framework [14]. As a result, a time varying weighting scheme is obtained to mitigate the problem of using too few or too many redundant weak learners. Similarly, another algorithm OSB.Exp is also proposed in [44] for the same purpose. However, since the improvements due to OSB.Ocp over OSB.Exp are for the large data sets, we only include OSB.Ocp in our comparisons that is more attractive for our framework. There is no new parameter in this case in addition to the ones from the OSB.

5. *Online Tree based Non-adaptive Competitive Classification* - "UC.Rnd" [6]: In addition, we also compare our algorithms with the method UC.Rnd [6], where the proposed technique is non-adaptive, i.e., not self organizing, in

terms of the space partitioning. We use this method in our comparisons to illustrate the gain due the self organizing structure proposed in this chapter. Depth of the tree is set to 4 uniformly in all of our experiments.

We use the perceptron algorithm [86] as the weak learners in these compared methods and as the simple local models in our algorithms (AUC.Rnd and AUC.Avg) and in UC.Rnd. We set the learning rate to $\eta = 0.05$ and the tree depth to 4 in our algorithms uniformly in all of our experiments. We use $N = 100$ weak learners for all other methods. Note that a depth-4 tree corresponds to $31 = 2^5 - 1$ local models. The proposed algorithms have linear complexity in the depth, whereas the compared methods have linear complexity in the number of weak learners.

These methods are tested on the data sets of the processed format[7] as in [44]; however, we discard the data sets "Ijccnl" and "Web Page", where a constant negative decision readily gives a reasonable performance as also pointed out in [45] since those sets are heavily imbalanced. Instead, we study the well-known "banana" data set and a binarized synthetic multi-class data "BMC" consisting of 6 identical Guassian components that are located in two dimensions such that the separation between the two classes is strongly nonlinear, cf. Fig. 3.6.

Each method is sequentially presented with the same data sequence and we calculate the error rate for the complete stream. This process is repeated for 100 random permutations (10 for the data sets of length longer than 10000) and the average error rates are reported in Fig. 3.4. For a fair comparison, we truncate each data instance to unitary norm, i.e., $\boldsymbol{x}_t \leftarrow \frac{\boldsymbol{x}_t}{\max(\|\boldsymbol{x}_t\|,1)}$, as in [44] whose original results are given in the third row and our corresponding findings are in the second row. Note that the results of [44] are only over 5 trials. In the first row, we present the results for the data normalized to the range $[-1, 1]$ without the norm truncation.

Our algorithms (including UC.Rnd [6]) consistently outperform the other methods with only one exception in case of the "Mushrooms" data set. The

[7]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

proposed approach has a slower rate of learning only in this data set. In particular, the compared methods essentially fail at the "Banana" and "BMC" data sets, which indicates that other methods are not able to extend to complex nonlinear separations starting from linear weak learners. On the contrary, our method successfully models those complex nonlinear separations with piecewise linear curves (cf. Fig. 3.6) and therefore, provides a highly superior performance especially on the "Banana" and "BMC" data sets (cf. Fig. 3.4). On these data sets, the gain due to the self adaptation capabilities (which are devised in this chapter) is also clearly demonstrated, cf. AUC.Rnd and AUC.Avg vs UC.Rnd on "Banana" and "BMC". We also observe that averaging (compared to randomization) incorporates better with our self organizing strategy as AUC.Avg almost always outperforms AUC.Rnd. We emphasize that AUC.Avg has superior performance compared to AUC.Rnd in the transient state, however, both of them are asymptotically convergent to the same performance level, cf. the comparable performance in the last three data sets of relatively long sequences in Fig. 3.4. In general, the proposed methods AUC.Avg and AUC.Rnd generalize and asymptotically cover the solution of the method UC.Rnd [6]. However, AUC.Avg has a significantly better transient characteristics and UC.Rnd occasionally performs poorly (such as "BMC", "Banana" and "Adult") depending on the complexity of the intrinsic optimal separation in the data. To validate this discussion, we present the long term behavior of these three algorithms on concatenated data sets in Fig. 3.5, where the performances improve in favor of AUC.Rnd and AUC.Avg compared to UC.Rnd with sufficiently many observations. For instance, AUC.Rnd significantly outperforms UC.Rnd on the "Heart" data set in the long run, although initially performed comparable in Fig. 3.4 (on relatively short sequences). Note that AUC.Rnd and AUC.Avg converge to the same performance level but AUC.Avg has better transient performance.

We note that: *i)* The proposed algorithms perform better with data normalized to the range $[-1, 1]$ given in the first row of Fig. 3.4; however, we continue with the norm truncated data to be aligned with the results in the corresponding studies [44, 45]. *ii)* The mismatches between our findings over 100 trials and originally reported error rates in [44] over 5 trials (second row vs third row of Fig.

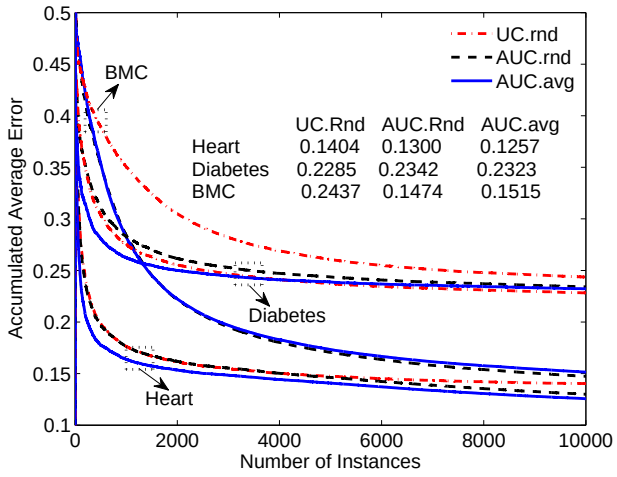


Figure 3.5: Long term behavior over 100 trials based on concatenation of random permutations of data sets: norm-truncated "BMC", "Heart" and "Diabetes".

3.4) are due to the relatively larger standard deviations in the results of [44]. *iii)* The proposed algorithms, which use perceptrons as local models, also outperform the other methods, which use the naive bayes weak learners, for most of the data sets (8/10 of applicable cases, i.e., "BMC" and "Banana" do not apply) according to the error rates reported in [44]. These cases are indicated (only for the method AUC.Avg) with a star in Fig. 3.4. *iv)* In Fig. 3.6, we present the piecewise linear separation boundaries returned by the algorithms UC.Rnd, AUC.Rnd and AUC.Avg on the "BMC" data set after one and two passes. The regions, in which the algorithms are unsure of whether to decide for the output -1 or 1, are shown in black dots. Note that these regions are mostly near the region boundaries.

3.3.2 Non-Stationary Data: Concept Change/Drift

In this part, we study the proposed algorithms with non-stationary data, where there might be continuous or sudden/abrupt changes in the source statistics, i.e., concept change. Since our algorithms process only one instance at a time without storing it, we choose the algorithm DWM [23] with perceptron or naive bayes experts for the comparison, which is also a truly online algorithm without

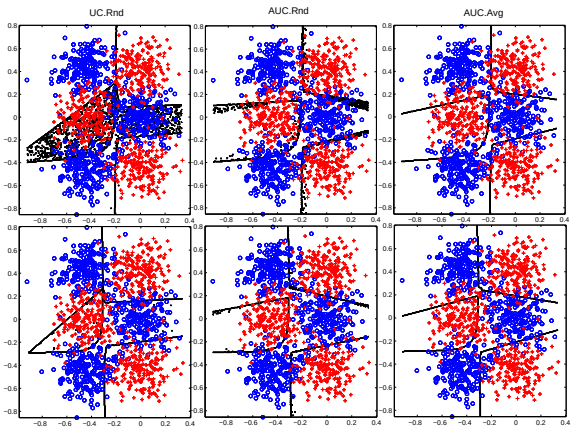


Figure 3.6: Piecewise linear separation boundaries in the "BMC" data set after one and two passes (first row: first pass, second row: second pass; first column: UC.Rnd, second column: AUC:Rnd, third column: AUC:Avg). The randomized algorithms are unsure in the black dotted regions that are mostly near the boundaries.

storage or batch process requirements. Hence, we obtain two algorithms: DWM-P, DWM-N. Most of the other algorithms [48, 60, 75–78] specialized for concept drift have sliding window approaches: A batch of data within the window of most recent examples are continuously processed to detect or learn a concept change; and then, ideally, a quicker adaptation to the change with this new information (cf. Table 1 in [48]) is achieved. The window size must be chosen large enough to fully capture the statistics of the active concept; and if it is too large, then the desired adaptation to the concept change quickly degrades at the risk of wasting the computational resources spent on the window processing. Clearly, there is no optimal window size, which introduces another parameter that has to be tuned for each experiment, i.e., window size $= ws \in \{100, 200, 500, ..., 2000\}$ in [48]. We also emphasize that such sliding window approaches are essentially batch algorithms, i.e., not truly online, and -in general- ws times slower than the truly online counterparts such as DWM or AUC.Avg. Therefore, such approaches do not fit into our framework. Nevertheless, we devise an online version of the batch classifier proposed in [42] using the sliding window approach that also learns the space partitioning and the classifier jointly using the coordinate ascent approach. In the following, we explain the details of the compared methods in this part.

1. *Dynamically Weighted Majority Algorithm* - "DWM" [23]: This method is from the "predicting with expert advice" [14] framework, which allows addition and removal of experts at each instance. The aim is to quickly match the performance of the best expert that is trained with only the active concept. Briefly, if a concept change occurs, then all the existing experts do not perform well and the DWM adds a new expert without an explicit concept change detector and anticipates to quickly match it. Parameters are set as in [23], i.e., $\beta = 0.5, \gamma = 0.01, N_i = 1$ (N_i is the initial number of experts that is bounded by $N = 100$ during the learning), uniformly in all of our experiments. We use perceptron (DWM-P) and naive bayes experts (DWM-N). For the naive bayes, we assume Gaussian with arbitrary mean and arbitrary but diagonal covariance class conditional densities and then implement the optimal Bayes classifier in an online fashion.

2. *Sliding Window based Local Space Partitioning* - "WLSP" [42]: This is a tree based batch classifier that learns the (locally) optimal space partitioning and the resulting classifier jointly: Given the partitioning, the classifier is obtained; and given the classifier, the partitioning is obtained through a certain number of iterations. The algorithm is initialized with a random partitioning many times, and the best performing one is returned. We use this procedure for an online application as follows: At each instance, we provide the algorithm with the most recent $ws = 100$ instances and the iterations are started twice with a) the most recent space partitioning and b) a random partitioning. The best performing one (among the two) is kept and the next instance is streamed. We use 10 iterations at each step; and the perceptron for a local model.

We run these methods on the "BMC" data set (1200 instances, Fig. 3.6), where a sudden/abrupt concept change is obtained such that the instances are rotated (clock-wise around the origin) 180° after the 600th instance. This effectively means a label flip and the resulting data set is denoted as "BMC-F". For a continuous concept drift, we rotate each instance 180°/1200 starting from the beginning; and the resulting data set is denoted as "BMC-C". In Fig. 3.7, we present the error plots for the compared methods over 1000 trials. At each

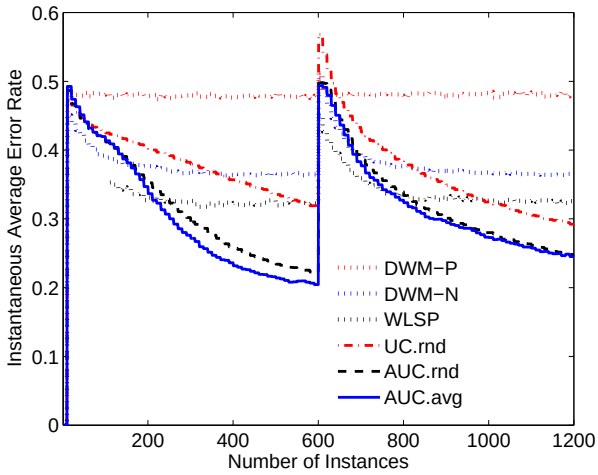


Figure 3.7: Performance of the compared methods in case of the abrupt concept change in the "BMC-F" data set. At the 600th instance, there is a 180° clock-wise rotation around the origin (derived from the "BMC data set") that is effectively a label flip. Since the data is strictly non-Gaussian, the method DWM-P or DWM-N does not perform well in this case: both methods essentially fail with an error rate that is slightly better than the random guess. In the first 100 instances, the sliding window based approach WLSP does not produce results.

10th instance, we test the models returned by the algorithms with 1200 instances drawn from the active set of statistics (active concept).

Note that since the "BMC" data is strongly Non-Gaussian with strongly non-linear class separations, the method DWM with perceptron or naive bayes do not perform well on "BMC-F". For instance, DWM-P operates with an error rate fluctuating around $0.48-0.49$ (random guess). This results since the performance of the method DWM is directly dependent on the expert success and observe that, both base learners (perceptron or the naive bayes) fail due to the high separation complexity in "BMC-F". On the other hand, the method WLSP quickly converges to its steady state, however, it is also asymptotically outperformed by our methods at both concepts with sufficient number of observations. Increasing the window size is clearly expected to boost the performance of WLSP, though, at the cost of increased computational complexity. It is already significantly slower than our techniques with even $ws = 100$, cf. Section 3.3.3. When the method WLSP is run on the "BMC-C" data set in case of the continuous concept drift, cf. Fig.

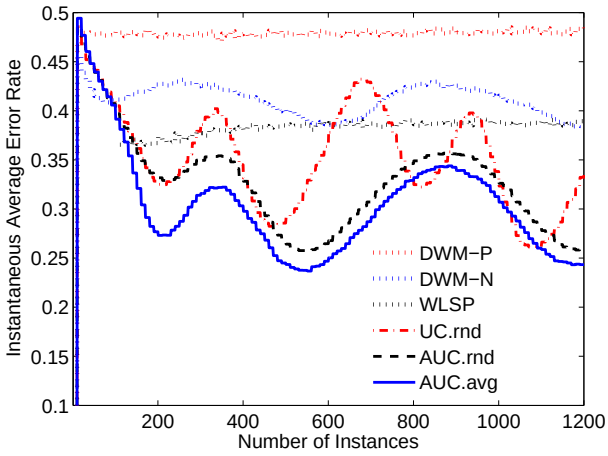


Figure 3.8: Performance of the compared methods in case of the continuous concept change in the "BMC-C" data set. At each instance, there is a 180°/1200 clock-wise rotation around the origin (derived from the "BMC data set"). Since the data is strictly non-Gaussian, the method DWM-N also -similar to DWM-P- does not perform well in this case: both methods essentially fail with an error rate that is slightly better than the random guess. Hence, only DWM-P is shown for clarity. In the first 100 instances, the sliding window based approach WLSP does not produce results.

3.8, its performance significantly degrades (compared to the one on "BMC-F") since, in this case, WLSP is trained with the batch data of a continuous mixture of concepts in the sliding windows. Under this continuous concept drift, AUC always -not only asymptotically as in the case of "BMC-F"- performs better than WLSP. Hence, the sliding window approach is sensitive to the continuous drift. Our discussion about the DWM method on the concept change data "BMC-F" remains valid in the case of the concept drift of "BMC-C". In these experiments, the power of the proposed self-organizing strategy is obvious as AUC (both .Rnd and .Avg) almost always outperforms UC.Rnd [6].

In the rest of the experiments involving concept changes, we follow the comparison framework of [23]. We conduct tests on the stagger concepts (a standard data set for concept change experiments) [23], where the data $\{1,2,3\}^{120}$ includes 2 concept switches among three concepts $1 \rightarrow 2 \rightarrow 3$ at the 41th and 81th instances. The concept definitions are as follows. Concept 1: $y = 1$, if $\mathbf{x}(1) = 3 \wedge \mathbf{x}(3) = 1$, Concept 2: $y = 1$, if $\mathbf{x}(1) = 1 \vee \mathbf{x}(2) = 2$ and Concept 3: $y = 1$, if $\mathbf{x}(3) =$

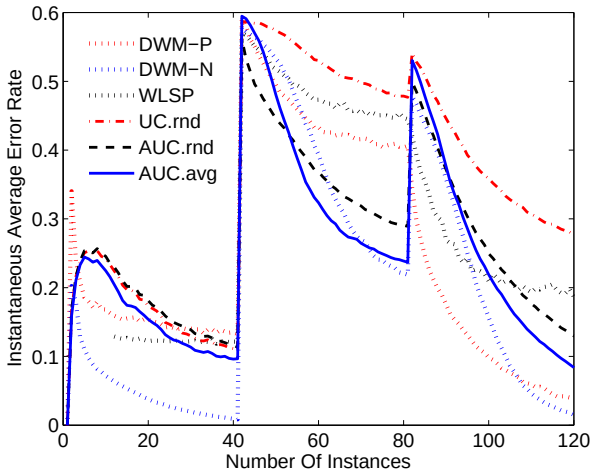


Figure 3.9: Performance of the compared methods in case of the stagger concepts.

$2 \vee \mathbf{x}(3) = 3$ (otherwise, $y = -1$). In this case, we use the window length $ws = 10$ for the method WLSP. In Fig. 3.9, we present the error curves for the compared methods. Although the method DWM-P and DWM-N do not perform well on the "BMC-F" and "BMC-C" data sets, DWM-N and our method AUC.Avg perform comparably on the second and third concepts, whereas DWM-N performs better on the first one. The method WLSP is outperformed by all other techniques since this data set requires a quick adaptation and WLSP basically could not access to sufficiently many observations in its windows as quickly as required. We observe that DWM-P is not able to adapt to the second concept (nonlinear class separation), whereas it outperforms DWM-N on the third concept (linear class separation): it is difficult to choose the right expert to be used with DWM. Note that the naive bayes expert can (limitedly) adapt to nonlinear separations at the cost of a slower convergence in the case of linear separations (compared to the percetpron), whereas the perceptron cannot adapt these nonlinear separations. On the other hand, the proposed methods AUC.Avg and AUC.Rnd can adapt to arbitrarily nonlinear separations (cf. Fig. 3.7 and Fig. 3.8) without sacrificing much from the transient state accuracy.

In particular, the proposed methods either outperform (on the second concept) all the compared algorithms or perform comparably with the best competitor (on the first and third concept) in the steady state, cf. the long term results on the

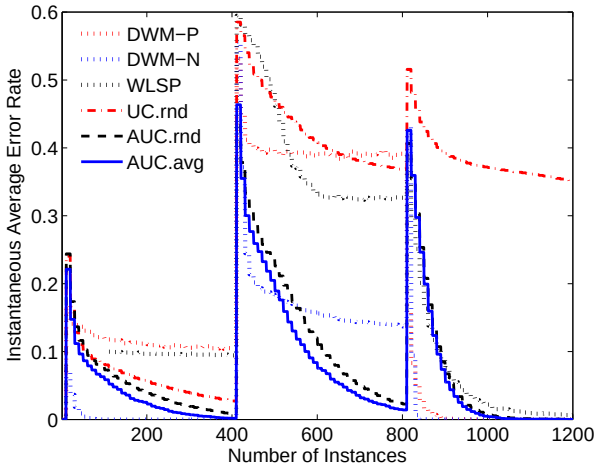


Figure 3.10: Performance of the compared methods in case of the stagger concepts in the long term.

stagger concepts presented in Fig. 3.10 (where $ws = 100$ for WLSP).

We finally run all algorithms on a larger concept change problem, where we use the synthetic drifting hyperplane (DH) data set [23], which consists of 2000 instances of dimension 10, i.e., $\mathbf{x} \in [0, 1]^{10}$. The data set includes 3 concept changes among 4 concepts: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, where the concept change from i to $i+1$ happens at the $(500i+1)$th instance during the online stream. Concept definitions are as follows: If the concept i is active, then $y = 1$, if $\mathbf{x}(j)+\mathbf{x}(j+1)+\mathbf{x}(j+2) > 0.5$ for $(i, j) \in \{(1, 1), (2, 2), (3, 4), (4, 7)\}$ ($y = -1$, otherwise). On this data set, the proposed algorithms perform as the second best with comparable performance with the best performing algorithm (DWM-N) on the second concept, Fig. 3.11.

We conclude that the DWM algorithm is significantly sensitive to the expert choice and its performance is upper-bounded by the chosen expert success. When the -within concept- target separations are relatively simple and linear, DWM-P demonstrates a quick concept adaptation; when the target separations are strongly non-linear, both the methods DWM-P and DWM-N do not perform well, e.g., BMC-F or BMC-C. Choosing more sophisticated experts is always a good option, though, at the cost of increased computational load, cf. 3.3.3. On the contrary, the proposed algorithms AUC.Avg and Auc.Rnd are able to learn

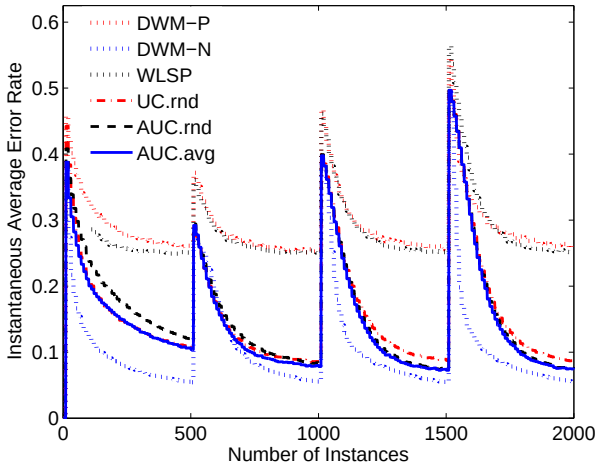


Figure 3.11: Performance of the compared methods on the hyper plane drifting data set.

to arbitrarily complex separations both in the stationary and non-stationary settings without mandating to choose the right local simple model/expert (as opposed to the method DWM) while outperforming the state-of-the-art techniques in the stationary setting with strong adaptation capabilities under the concept change/drift.

3.3.3 Computational Complexity: Running Times

We present the running times of the compared methods on the data set "BMC" (processing 1200 instances over 1 trial) for the optimized MATLAB implementations run on a daily-use machine (Intel(R) Core(TM) i5-3317U CPU @ 1.70 GHz with 4 GB memory) in Fig. 3.12. Theoretically, the presented times must scale with $N/1200$ on any data set of size N. The proposed methods are computationally the most efficient (except the base learner perceptron) among the competitors, e.g., AUC.Avg $\sim 3.5\times$ faster than DWM-P. In particular, the method DWM becomes significantly computationally demanding when used with a more sophisticated expert, e.g., DWM-N is $\sim 3\times$ slower than DWM-P.

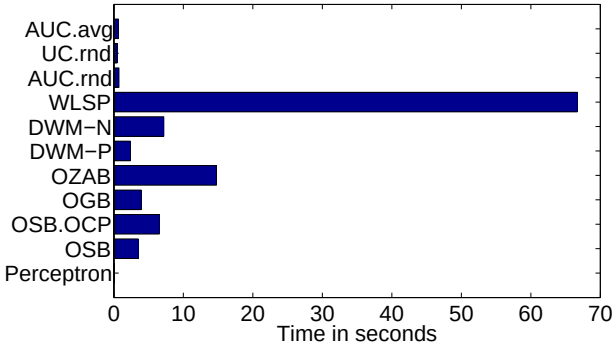


Figure 3.12: Running times of the compared methods when processing the "BMC" data set of 1200 instances.

3.4 Discussion

We propose a highly efficient (in terms of the computational complexity) online algorithm for supervised learning that is appropriate for big data applications. The introduced approach combines simple linear classifiers specialized in local regions of the feature space to generate a decision such that both the feature space partitioning and the corresponding local linear models are jointly optimized. The result is a highly dynamical decision tree structure that adaptively organizes itself to minimize the accumulated loss over time for any given data stream regardless of the underlying stationary or non-stationary statistics. We provide strong theoretical performance guarantees without any statistical assumptions on the underlying data. We present a comprehensive experimental comparison of our algorithm with respect to the state-of-the-art techniques from the literature of ensemble methods and classification trees on the commonly used benchmark data sets. We experimentally show that our algorithm significantly outperforms the state-of-the-art techniques with remarkable adaptation capabilities to non-stationarity, i.e., concept change/drift.

In Chapter 2 and Chapter 3, we concentrate on analyzing the online learning algorithms in the deterministic sense such that the produced results hold regardless of the source statistics. In addition, we also concentrate on adaptation to the non-stationarity in the data such as slow drifts or sudden/abrupt changes. In

this regard, the proposed algorithm in this chapter demonstrates highly superior performance over the method studies in Chapter 2 as well as the state-of-the-art techniques.

In the following chapters, we extend and study the impact of data corruptions on the performance of the target application. We start with an introductory example in Chapter 4, where we assume a model for the observations and introduce a batch processing method. Then we drop the model assumption in Chapter 5, where we demonstrate that the corruptions can be considered as anomalies and detected by an anomaly detection approach for corruption imputation/correction purposes. For this purpose, we propose an online anomaly detection algorithm in Chapter 6.

Chapter 4

A Novel and Robust Parameter Training Approach for HMMs Under Noisy and Partial Access to States

In a wide variety of applications in time series analysis ranging from speech processing [24, 87–93], bioinformatics [94, 95] to natural language processing [96–99], the observation sequence is represented as a stochastic process, depending on another stochastic process that generates a sequence of hidden (unobserved) states. With certain conditional independence properties regarding the observations as well as the states, this is known as Hidden Markov Model (HMM) [24, 100]. In this chapter, we particularly concentrate on discrete-time finite-state HMM with finite alphabet, which is described by two families of random variables: the hidden state sequence Z_t and the observation sequence Y_t. The random variables in the state sequence Z_t form a stochastic, discrete-time Markov chain and the observation Y_t, conditioned on the present hidden state Z_t, is independent with the past and future observations as well as the hidden states. The corresponding conditional independence structure of the model is shown as a directed acyclic graph [101] in Fig. 4.1a. Hence, an HMM is completely characterized by the set

of parameters $\lambda = (\pi, A, B)$, where A_{ij} is the state transition probabilities, B_{ij} is the observation emission probabilities and π_i is the initial state probabilities. A detailed description of the model can be found in [24]. Estimation of these model parameters $\lambda = (\pi, A, B)$ is an important problem in applications using HMM [24,91,92,94–99,102,103]. Since there is no closed form solution for the set of parameters that maximizes the probability of the observation sequence given the model, instead, iterative algorithms such as the Expectation-Maximization (EM) algorithm [104,105] (or equivalently the Baum-Welch method [106]) is used to obtain a local optimal solution [24]. In this study, we derive a new set of iterative EM equations that yield a locally optimal solution for the model parameters, when the ordinary model of the observation sequence, e.g., as in [24], is different. In our model, in addition to the observation sequence y_t (upper case letters are used to denote the random variables and the lower case letters are used to denote the corresponding realizations), we observe a part of the hidden state sequence as side information. More precisely, at every time instant t, we observe the hidden state z_t as x_t with probability τ, i.e., with $1-\tau$ probability the state stays hidden. This gives partial access to the state sequence as side information which is, in our work, incorporated in the corresponding parameter estimation problem and the associated EM algorithm. We emphasize that the state observations are not necessarily confined to a time interval but may even be sparsely and randomly distributed along the complete time span of the application. In the limiting case, if τ is 0, then there would be no state observation, and we recover the ordinary, unsupervised HMM training. Therefore, our model provides a generalized framework by letting partial access to the state sequence. Moreover, we also allow that a state observation might be corrupted with noise such that if z_t is ever observed then $P(X_t \neq z_t | Z_t = z_t) = 1 - p$. Then the corresponding conditional independence structure in this case of partially observable states is shown in Fig. 4.1b. Under these new circumstances, we explicitly provide the mathematical derivations of the new set of iterative EM equations that incorporates the side information and estimate the model parameters accordingly. In these derivations, the probability that a state observation is incorrect, $1-p$, is assumed to be known and it is provided to our algorithm as a parameter, p, which defines the confidence on the side information. Simulations show that our method is robust to

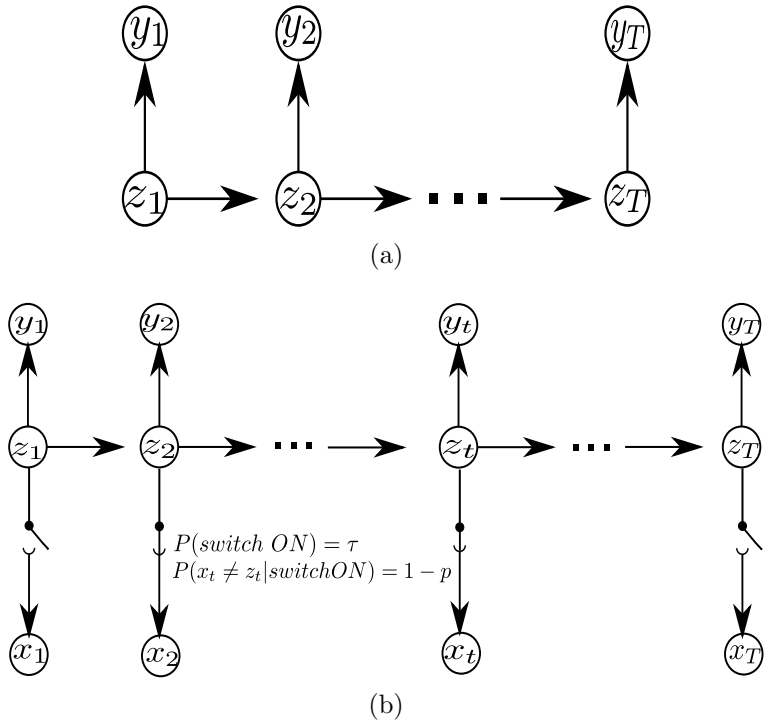


Figure 4.1: (a) The conditional independence structure of an HMM with discrete-time finite-states z_t and observations y_t of a finite alphabet. (b) The conditional independence structure of an HMM with Partial and Noisy Access x_t to the State Sequence.

the confidence parameter p, even if it does not exactly match with the underlying true quality of the side information, p_{true}. Since the hidden state sequence is partially observed, our work falls into the category of Partially Hidden Markov Model (PHMM) training (note that this term is used in [107] in a different context). Similar to semi-supervised learning, PHMMs use both "labeled" (in our context the state information) and "unlabeled" data to obtain improved model training. Such an approach is suitable, when we have access to a limited amount of labeled data along with a large amount of unlabeled data. This happens, as an example, in speech processing applications [93], where labeling, i.e., transcription, is naturally costly [93, 108], hence only limited amount is affordable, and transcriptions may contain errors. Furthermore, by allowing noisy access to the states, we model "mislabeling" event that may occur during labeling stage. PHMMs, to the best of our knowledge, date back to the studies [97–99] in the area of Natural Language Processing. In these studies, tagged text, corresponding to

the known states of a PHMM, are first analyzed through a relative frequency modeling to construct an initial model, then this model is fed into the ordinary HMM training algorithm. However, these studies do not rigorously show how the partial state information is incorporated within the ordinary HMM parameter estimation framework. The Maximum Likelihood Estimator (MLE) for the model parameters in a special case of PHMMs, where only a certain state from the state space in the underlying Markov chain is known, is theoretically (consistency and asymptotic normality of the estimator) analyzed in [96]. However, the equations for computing the MLE (using the EM algorithm or other Likelihood maximization techniques) in this special case of PHMM is not derived. In [103], iterative EM equations for a general case, where each observation can only belong to a pre-defined set of acceptable states are given, but no complete derivation is provided. On the contrary, we explicitly derive the new set of iterative EM equations for the PHMM parameter estimation problem, when there is partial access to the underlying hidden state sequence. Furthermore, the partial observation of the state sequence might be prune to noise in our model and this case is not considered in the existing literature.

After we provide the brief description of the basic HMM framework and the parameter estimation equations in Section 4.1, we derive the new set of iterative EM equations that incorporates partial and noisy access to the state sequence as side information in Section 4.2. Simulations are presented in Section 4.3 and the chapter concludes with final remarks in Section 4.4.

4.1 Problem Description

In this section, we briefly describe the basic framework for the HMM parameter estimation problem [24]. For the sake of notational simplicity, we study discrete-time finite-state HMM with finite alphabet. However, our derivations for incorporating the side information in Section 4.2 can be readily extended to the case, where the observations come from a continuous distribution and outcomes are vectors. A discrete-time HMM with finite alphabet is formally a

Markov model, for which we have a sequence of observations, y_t, drawn from a finite alphabet $V = \{v_1, v_2, ..., v_{N_v}\}$, i.e., $y_t \in V, 1 \leq t \leq T$. We also have a sequence of hidden (unobserved) states $z_t \in S = \{s_1, s_2, ..., s_{N_s}\}$, where S is the set of possible states, generated from a Markov process. Namely, $P(Z_t = z_t|\mathbf{Z}_1^{t-1} = \mathbf{z}_1^{t-1}) = P(Z_t = z_t|Z_{t-1} = z_{t-1})$, where (and in this study) the upper case (bold) letters are used to denote (a collection of) random variables and the lower case (bold) letters denote the corresponding (collection of) realizations, i.e., $\mathbf{z}_1^{t-1} = \{z_1, z_2, ..., z_{t-1}\}$, similarly for $\mathbf{Z}_1^{t-1}$. The observation sequence $y_1, y_2, ..., y_t$ is generated based on the state sequence $z_1, z_2, ..., z_t$, i.e., $P(Y_t = y_t|\mathbf{Z}_1^t = \mathbf{z}_1^t, \mathbf{Y}_1^{t-1} = \mathbf{y}_1^{t-1}) = P(Y_t = y_t|Z_t = z_t)$. We consider A as the transition matrix, where A_{ij} represents the transition probability from state s_i to s_j, $A_{ij} = P(Z_t = s_j|Z_{t-1} = s_i)$. Similarly, B is the observation probabilities at each state, i.e., $B_{ij} = P(Y_t = v_j|Z_t = s_i)$. In order to complete the HMM observation model, we also define the initial state probabilities as $\pi_i = P(Z_1 = s_i)$. Thus, $\lambda = (\pi, A, B)$ represents the parameter set that completely characterizes the HMM model shown in Fig. 4.1a.

As for the HMM parameter estimation problem, the Maximum Likelihood (ML) estimation, $\arg\max_\lambda P(\mathbf{Y} = \mathbf{y}|\lambda)$, $\mathbf{y} = \{y_1, y_2, ..., y_T\}$, is locally solved iteratively using the Expectation Maximization (EM) algorithm [24]. Then the iterative re-estimation formulas for the HMM parameters providing the ML estimate are as follows:

$$\hat{A}_{ij} = \frac{\sum_{t=1}^{T-1} \epsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \ \hat{B}_{ij} = \frac{\sum_{t=1}^{T-1} 1_{\{y_t = v_j\}} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \ \hat{\pi}_i = \gamma_1(i). \tag{4.1}$$

Here, $\epsilon_t(i,j)$ is defined as the probability of transition at time t from state s_i to s_j, given the observations $\mathbf{y}$ and the current parameters of the model, i.e.,

$$\epsilon_t(i,j) = P(Z_t = s_i, Z_{t+1} = s_j|\mathbf{Y} = \mathbf{y}, \lambda),$$

and $\gamma_t(i)$ is defined as the probability of being at state $z_t = s_i$, given the observations and the model, i.e.,

$$\gamma_t(i) = \sum_{j=1}^{N_s} \epsilon_t(i,j).$$

These iterative re-estimation formulas can be computed efficiently through the well-known forward-backward procedure [109,110], which is based on the forward and backward variables and the corresponding recursions. The forward variable, $\alpha_t(i)$, along with the recursion in [24] is given by

$$\alpha_t(i) = P(\mathbf{Y}_1^t = \mathbf{y}_1^t, Z_t = s_i|\lambda) = B_{iy_t} \sum_{j=1}^{N_s} \alpha_{t-1}(j) A_{ji},\ \alpha_1(i) = \pi_i B_{iy_1},\ 2 \leq t \leq T,$$

which is the probability of observing $\mathbf{y}_1^t$ and being at state $z_t = s_i$, given the model λ. Similarly, the backward variable is given by

$$\beta_t(i) = P(\mathbf{Y}_{t+1}^T = \mathbf{y}_{t+1}^T|Z_t = s_i, \lambda) = \sum_{j=1}^{N_s} \beta_{t+1}(j) A_{ij} B_{jy_{t+1}}, \beta_T(i) = 1, 1 \leq t \leq T-1,$$

which is the probability of observing $\mathbf{y}_{t+1}^T$, given the state $z_t = s_i$ and the model. By noting that $P(Z_t = s_i, Z_{t+1} = s_j, \mathbf{Y} = \mathbf{y}|\lambda) = \alpha_t(i) A_{ij} B_{jy_{t+1}} \beta_{t+1}(j)$ and $P(\mathbf{Y} = \mathbf{y}|\lambda) = \sum_{k=1}^{N_s} \sum_{l=1}^{N_s} \alpha_t(k) A_{kl} B_{ly_{t+1}} \beta_{t+1}(l)$, we obtain

$$\epsilon_t(i,j) = \frac{\alpha_t(i) A_{ij} B_{jy_{t+1}} \beta_{t+1}(j)}{\sum_{k=1}^{N_s} \sum_{l=1}^{N_s} \alpha_t(k) A_{kl} B_{ly_{t+1}} \beta_{t+1}(l)}.$$

Then the iterative re-estimation formulas for the HMM parameters given in (4.1) can be computed efficiently using the forward-backward recursions. As a result, given the training data, we estimate the HMM parameters λ by the iterative re-estimation procedure defined by the EM algorithm. Namely, given the HMM parameters λ_{q-1} at an iteration q, we re-estimate the model parameters as λ_q using the re-estimation formulas in (4.1). This procedure is guaranteed to improve the likelihood of the observations at every iteration and converge to a set of HMM parameters $\hat{\lambda}$, which is at least locally optimal (cf. [24] and the references therein).

In the following section, we incorporate the noisy side information into the HMM framework. To this end, we introduce the "incomplete-data problem" [104], derive the conditional expectation of the complete-data log likelihood to obtain the iterative re-estimation formulas and finally adapt the forward-backward procedure for the case of partially observable states.

4.2 HMM training with noisy and partial access to the state sequence

In this section, we derive the new set of iterative EM equations for the HMM parameter estimation, when we have noisy and side information on the hidden states. Here, we have an observation sequence $y_t \in \mathbf{y} = \{y_1, y_2, ..., y_T\}$, with *partial* and *noisy* access to the hidden states, $z_t \in \mathbf{z} = \{z_1, z_2, ..., z_T\}$, as this side information. Each hidden state $z \in \mathbf{z}$ might be observed as x with probability τ, i.e., we do not necessarily have a state observation at a given time instant. Hence, we have *partial* access to the hidden state sequence. In addition to this partial access, a state observation x might also be *noisy* such that $P(X \neq s|Z = s) = (1 - p)$. We assume that if a state observation is erroneous, then $P(X = s_2|Z = s_1) = \frac{1}{N_s - 1}$, $\forall s_1, s_2 \in S$ and $s_1 \neq s_2$. We here note that if the probability distribution of the erroneous state observations concentrated at a particular state (for each observed hidden state), then we would have more information about the underlying hidden states. For an instance, suppose we observed $x = s$ and it is erroneous, i.e., $z \neq s$. Then, z would be more likely to be the state s^* for which the corresponding erroneous state observations concentrated at $s \neq s^*$. This clearly would be a favorable case in terms of the HMM parameter estimation as well as the recognition of the underlying hidden states. In this study, we targeted the worst case, i.e., the case of most ambiguous state observations, when there is an error. Hence, we assumed the probability distribution of erroneous state observations is not concentrated and so uniform. For ease of notation, we define the state observations at every time t as $x_t \in \mathbf{x} = \{x_1, x_2, ..., x_T\}$, such that if z_t is ever observed as $s \in S$, then $x_t = s$. Otherwise, $x_t = s_0$, where s_0 is a pseudo-state. This expands our state space to $S' = S \cup \{s_0\}$. Thus, we model mislabeling and partial labeling jointly in one complete framework as shown in Fig. 4.1b.

After having described our model, in the following, we first deduce the iterative re-estimation formulas of the HMM parameters under partial and noisy access to the hidden states. In the following, we consider the HMM as an instance of the

"incomplete data problem" and derive the conditional expectation of the complete data log-likelihood to apply the EM algorithm to the Maximum Likelihood estimation. Then we present the forward and backward recursions for an efficient computation of the deduced re-estimation formulas.

4.2.1 The re-estimation formulas for the HMM parameters through likelihood maximization under partial and noisy access to the states

The ML estimation of the HMM parameters in this case of partial and noisy access to the hidden states is given by the maximization of the log-likelihood of all observations with respect to the model parameters

$$\arg\max_{\lambda} \log\left(P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)\right).$$

Clearly, this maximization would have been computationally more tractable if the hidden states $\mathbf{z}$ were completely known, i.e., if we had $x_t = z_t$ at each time t, in addition to the observation sequence $\mathbf{y}$, since then the corresponding conditional probability distribution could be factorized into a simpler form (due to Markov property). Hence, considering the unobserved hidden states as the missing data, it is more plausible to formulate this parameter estimation problem as an instance of the "incomplete data problem" [104] and consider the corresponding augmentation of the observations with the states as $\mathbf{w} = (\mathbf{y}, \mathbf{x}, \mathbf{z})$ as the "complete data". Then one can construct the following relationship between the incomplete-data log-likelihood $\log\left(P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)\right)$ and the complete-data log-likelihood $\log\left(P(\mathbf{W}=\mathbf{w}|\lambda)\right)$:

$$\begin{aligned}\log\left(P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)\right) &\geq \sum_{\mathbf{z}} Q(\mathbf{z};\lambda)\log\left(P(\mathbf{W}=\mathbf{w}|\lambda)\right) - \sum_{\mathbf{z}} Q(\mathbf{z};\lambda)\log\left(Q(\mathbf{z};\lambda)\right)\\ &= E_{\mathbf{Z}|\mathbf{Y},\mathbf{X},\lambda}\left\{\left(\log\left(P(\mathbf{W}=\mathbf{w}|\lambda)\right)\right) + C(\lambda)\right\},\end{aligned}$$

where $Q(\mathbf{z};\lambda) = P(\mathbf{Z}=\mathbf{z}|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda)$, $C(\lambda) = -\sum_{\mathbf{z}} Q(\mathbf{z};\lambda)\log\left(Q(\mathbf{z};\lambda)\right)$ and the expectation is with respect to the random variables $\mathbf{Z}$ conditioned on $(\mathbf{Y},\mathbf{X},\lambda)$. Based on this construction, we can readily maximize the conditional expectation of the complete data log-likelihood through the EM algorithm

(cf. [104] and the references therein) in order to maximize the incomplete data likelihood as originally intended. The EM algorithm works iteratively between two separate steps, known as E-steps and M-steps, such that at iteration q, the E-step calculates $Q(\mathbf{z};\lambda_{q-1})$ and the M-step maximizes the conditional expectation $E_{\mathbf{Z}|\mathbf{Y},\mathbf{X},\lambda_{q-1}}\left(\log\left(P(\mathbf{W}=\mathbf{w}|\lambda_q)\right)\right)$ with respect to the λ_q and note that $C(\lambda_{q-1})$ does not affect the maximization in the M-step of iteration q. We emphasize that since the complete data log-likelihood here is factorizable, the ML estimation of the HMM parameters becomes computationally more tractable when it is posed as an incomplete data problem. Indeed, we show that the forward-backward procedure stays applicable in this case of partial and noisy access to the states. We now deduce the re-estimation formulas for the HMM parameters.

Let $Q(\mathbf{z};\lambda_{q-1}) = P(\mathbf{Z}=\mathbf{z}|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda_{q-1})$ be the output of E-step, then M-step carries out the following maximization:

$$\begin{aligned}&\arg\max_{\lambda_q} E_{\mathbf{Z}|\mathbf{Y},\mathbf{X},\lambda_{q-1}}\left(\log\left(P(\mathbf{W}=\mathbf{w}|\lambda_q)\right)\right)\\&=\arg\max_{\lambda_q}\sum_{\mathbf{z}} Q(\mathbf{z};\lambda_{q-1})\log\left(P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\mathbf{Z}=\mathbf{z}|\lambda_q)\right),\end{aligned}$$

which, using the product of conditional probabilities, yields

$$\begin{aligned}&\arg\max_{\lambda_q} E_{\mathbf{Z}|\mathbf{Y},\mathbf{X},\lambda_{q-1}}\left(\log\left(P(\mathbf{W}=\mathbf{w}|\lambda_q)\right)\right)\\&=\arg\max_{\lambda_q}\sum_{\mathbf{z}} Q(\mathbf{z};\lambda_{q-1})\log\big(P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x},\mathbf{Z}=\mathbf{z},\lambda_q)P(\mathbf{X}=\mathbf{x}|\mathbf{Z}=\mathbf{z},\lambda_q)P(\mathbf{Z}=\mathbf{z}|\lambda_q)\big).\end{aligned}$$

Since $\mathbf{X}$ is independent with λ_q conditioned on $\mathbf{Z}$ and $\mathbf{Y}$ is independent with $\mathbf{X}$ conditioned on $(\mathbf{Z},\lambda_q)$, we obtain

$$\begin{aligned}&\arg\max_{\lambda_q} E_{\mathbf{Z}|\mathbf{Y},\mathbf{X},\lambda_{q-1}}\left(\log\left(P(\mathbf{W}=\mathbf{w}|\lambda_q)\right)\right)\\&=\arg\max_{\lambda_q}\sum_{\mathbf{z}} Q(\mathbf{z};\lambda_{q-1})\log\big(P(\mathbf{Y}=\mathbf{y}|\mathbf{Z}=\mathbf{z},\lambda_q)P(\mathbf{X}=\mathbf{x}|\mathbf{Z}=\mathbf{z})P(\mathbf{Z}=\mathbf{z}|\lambda_q)\big),\end{aligned}$$

where we can drop the factor $P(\mathbf{X}=\mathbf{x}|\mathbf{Z}=\mathbf{z})$ since it does not depend on λ_q and reach

$$\begin{aligned}&\arg\max_{\lambda_q} E_{\mathbf{Z}|\mathbf{Y},\mathbf{X},\lambda_{q-1}}\left(\log\left(P(\mathbf{W}=\mathbf{w}|\lambda_q)\right)\right)\\&=\arg\max_{\lambda_q}\sum_{\mathbf{z}} Q(\mathbf{z};\lambda_{q-1})\log\big(P(\mathbf{Y}=\mathbf{y}|\mathbf{Z}=\mathbf{z},\lambda_q)P(\mathbf{Z}=\mathbf{z}|\lambda_q)\big).\end{aligned}\tag{4.2}$$

We point out that the maximization in (4.2) does not involve the side information $\mathbf{x}$, except that $Q(\mathbf{z};\lambda_{q-1})$ is related to $\mathbf{x}$. However, since $Q(\mathbf{z};\lambda_{q-1})$ is calculated in E-step before M-step starts in the course of our algorithm, it only brings constant factors to the maximization in (4.2) and, hence, it does not affect the M-step derivations. Therefore, rest of the derivations follows the regular M-step derivations of the EM algorithm for the ordinary HMM parameter training and we estimate the transition probabilities as

$$\begin{aligned}\hat{A}_{ij} &= \frac{\sum_{\mathbf{z}} Q(\mathbf{z};\lambda_{q-1}) \sum_{t=1}^{T-1} 1_{\{z_t=s_i \wedge z_{t+1}=s_j\}}}{\sum_{\mathbf{z}} Q(\mathbf{z};\lambda_{q-1}) \sum_{t=1}^{T-1} 1_{\{z_t=s_i\}}} \\ &= \frac{\sum_{\mathbf{z}} \sum_{t=1}^{T-1} 1_{\{z_t=s_i \wedge z_{t+1}=s_j\}} P(\mathbf{Z}=\mathbf{z}|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda_{q-1})}{\sum_{\mathbf{z}} \sum_{t=1}^{T-1} 1_{\{z_t=s_i\}} P(\mathbf{Z}=\mathbf{z}|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda_{q-1})},\end{aligned}$$

where $1_{\{h\}}$ is the indicator function such that it outputs 1 if h, as a statement, is satisfied; and 0 otherwise. Here, the indicator function in the numerator and the denominator marginalizes the probability $P(\mathbf{Z}=\mathbf{z}|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda_{q-1})$, since the outer summation is over all possible hidden state sequences. Hence, we obtain

$$\begin{aligned}\hat{A}_{ij} &= \frac{\sum_{t=1}^{T-1} P(Z_t=s_i, Z_{t+1}=s_j|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda_{q-1})}{\sum_{t=1}^{T-1} P(Z_t=s_i|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda_{q-1})} \\ &= \frac{\sum_{t=1}^{T-1} \bar{\epsilon}_t(i,j)}{\sum_{t=1}^{T-1} \bar{\gamma}_t(i)}.\end{aligned}$$

Here, $\bar{\epsilon}_t(i,j)$ is the probability of transition at time t from state s_i to s_j, given the observations $\mathbf{y}$, the side information $\mathbf{x}$, and the model λ_{q-1}, i.e.,

$$\bar{\epsilon}_t(i,j) = P(Z_t=s_i, Z_{t+1}=s_j|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda), \tag{4.3}$$

and $\bar{\gamma}_t(i)$ is the probability of being at state $z_t=s_i$, given the observations, the side information and the model λ_{q-1}, i.e.,

$$\bar{\gamma}_t(i) = P(Z_t=s_i|\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x},\lambda) = \sum_{j=1}^{N_s} \bar{\epsilon}_t(i,j).$$

Note that the state transition probabilities are estimated, given the side information, as the expected number of transitions from state s_i to s_j divided by the expected number of times in the state s_i. Similarly, $\hat{B}_{ij}$ is given by

$$\hat{B}_{ij} = \frac{\sum_{t=1}^{T-1} 1_{\{y_t=v_j\}} \bar{\gamma}_t(i)}{\sum_{t=1}^{T-1} \bar{\gamma}_t(i)},$$

which is, given the side information, the expected number of times in the state s_i and observing v_j, divided by the expected number of times in the state s_i. Also, the set of initial probabilities for the hidden state z_1 is estimated as $\hat{\pi}_i = \bar{\gamma}_1(i)$. We next derive the forward and backward recursions in order to efficiently compute the EM estimate of the HMM parameters.

4.2.2 Forward and backward recursions

To derive the forward and backward recursions in this case of partially observable hidden states, we first update the variables of the forward-backward procedure, which incorporates the side information $\mathbf{x}$. The updated forward variable is defined as,

$$\bar{\alpha}_t(i) = P(\mathbf{Y}_1^t = \mathbf{y}_1^t, \mathbf{X}_1^t = \mathbf{x}_1^t, Z_t = s_i|\lambda), \tag{4.4}$$

the probability of observing $(\mathbf{y}_1^t, \mathbf{x}_1^t)$ and being at state $z_t = s_i$, given the model λ. Note that z_t is the correct and the underlying hidden state, whereas $\mathbf{x}_1^t$ are the state observations, for which we might have: (1) $x_t = s_0$ corresponding to the case that z_t is not actually observed and (2) noisy, if z_t is actually observed. Similarly, the backward variable,

$$\bar{\beta}_t(i) = P(\mathbf{Y}_{t+1}^T = \mathbf{y}_{t+1}^T, \mathbf{X}_{t+1}^T = \mathbf{x}_{t+1}^T|Z_t = s_i, \lambda), \tag{4.5}$$

is the probability of observing $(\mathbf{y}_{t+1}^T, \mathbf{x}_{t+1}^T)$, given the model and the state $z_t = s_i$. The updated forward and backward variables are the key variables of incorporating the side information. The following proposition explicitly relates these variables to the side information and provides the corresponding recursions.

Proposition 1: For the updated forward and backward variables defined in (4.4) and (4.5), we have

$$\bar{\alpha}_t(i) = \nu(x_t, s_i) B_{iy_t} \sum_{j=1}^{N_s} A_{ji} \bar{\alpha}_{t-1}(j),\ 2 \leq t \leq T,$$

$$\bar{\beta}_t(i) = \sum_{j=1}^{N_s} \nu(x_{t+1}, s_j) \bar{\beta}_{t+1}(j) A_{ij} B_{jy_{t+1}}, 1 \leq t \leq T-1,$$

where $\nu(x_t, s_i) = 1_{\{x_t = s_0\}}(1-\tau) + 1_{\{x_t = s_i\}}\tau p + 1_{\{x_t \neq s_i \wedge x_t \neq s_0\}} \frac{\tau(1-p)}{N_s - 1}$, $s_i \neq s_0$, $s_j \neq s_0$.

Proof: Using the marginalization over the random variable Z_{t-1}, we can obtain $\bar{\alpha}_t(i)$ as

$$\bar{\alpha}_t(i) = \sum_{j=1}^{N_s} P(\mathbf{Y}_1^t = \mathbf{y}_1^t, \mathbf{X}_1^t = \mathbf{x}_1^t, Z_t = s_i, Z_{t-1} = s_j | \lambda),$$

which can be expressed, using the product of conditional probabilities, as

$$\bar{\alpha}_t(i) = \sum_{j=1}^{N_s} [P(Y_t = y_t, X_t = x_t, Z_t = s_i | \mathbf{Y}_1^{t-1} = \mathbf{y}_1^{t-1}, \mathbf{X}_1^{t-1} = \mathbf{x}_1^{t-1}, Z_{t-1} = s_j, \lambda)$$
$$P(\mathbf{Y}_1^{t-1} = \mathbf{y}_1^{t-1}, \mathbf{X}_1^{t-1} = \mathbf{x}_1^{t-1}, Z_{t-1} = s_j | \lambda)].$$

By definition of the updated forward variable, we get

$$\bar{\alpha}_t(i) = \sum_{j=1}^{N_s} P(Y_t = y_t, X_t = x_t, Z_t = s_i | \mathbf{Y}_1^{t-1} = \mathbf{y}_1^{t-1}, \mathbf{X}_1^{t-1} = \mathbf{x}_1^{t-1}, Z_{t-1} = s_j, \lambda) \bar{\alpha}_{t-1}(j),$$

where Markov Property is applied to reach

$$\begin{aligned}
\bar{\alpha}_t(i) &= \sum_{j=1}^{N_s} P(Y_t = y_t, X_t = x_t, Z_t = s_i | Z_{t-1} = s_j, \lambda) \bar{\alpha}_{t-1}(j) \\
&= \sum_{j=1}^{N_s} P(Y_t = y_t, X_t = x_t | Z_t = s_i, Z_{t-1} = s_j, \lambda) P(Z_t = s_i | Z_{t-1} = s_j, \lambda) \bar{\alpha}_{t-1}(j) \\
&= \sum_{j=1}^{N_s} P(Y_t = y_t, X_t = x_t | Z_t = s_i, \lambda) P(Z_t = s_i | Z_{t-1} = s_j, \lambda) \bar{\alpha}_{t-1}(j).
\end{aligned}$$

Since X_t and Y_t are independent conditioned on (Z_t, λ), we obtain

$$\begin{aligned}\bar{\alpha}_t(i) &= \sum_{j=1}^{N_s} P(Y_t = y_t, X_t = x_t | Z_t = s_i, \lambda) A_{ji} \bar{\alpha}_{t-1}(j) \\ &= \sum_{j=1}^{N_s} P(X_t = x_t | Z_t = s_i, \lambda) P(Y_t = y_t | Z_t = s_i, \lambda) A_{ji} \bar{\alpha}_{t-1}(j).\end{aligned}$$

Then, by definition of the probability of error events in the side information, we get the proposition for the updated forward variable as

$$\bar{\alpha}_t(i) = \nu(x_t, s_i) B_{iy_t} \sum_{j=1}^{N_s} A_{ji} \bar{\alpha}_{t-1}(j),\ 2 \leq t \leq T.$$

As for the initialization, we set $\bar{\alpha}_1(i) = \nu(x_1, s_i)\pi_i B_{iy_1}$. Similarly, the corresponding recursion for the updated backward variable can be found as

$$\bar{\beta}_t(i) = \sum_{j=1}^{N_s} \nu(x_{t+1}, s_j) \bar{\beta}_{t+1}(j) A_{ij} B_{jy_{t+1}},\ 1 \leq t \leq T-1,$$

for which we have the initialization $\bar{\beta}_T(i) = 1$.■

Here, p reflects the confidence that we have on the side information and it is a parameter in our training algorithm. Ideally, when given a set of data, p (named as p_{train} in Section 4.3) should be set according to the underlying true noise level, $1 - p_{\text{true}}$, which is unknown. This brings an immediate trade-off between setting the confidence too low or too high, when an accurate guess about $1 - p_{\text{true}}$ is not present. If we have too high confidence, then our algorithm basically overfits to the noise in the side information, which degrades the state recognition performance as discussed in Section 4.3. On the other hand, if we have too low confidence, then our algorithm does not fully exploit the side information to its limit. We discuss this later in Section 4.3, when investigating the robustness of our algorithm to the confidence parameter p (p_{train} in Section 4.3).

We next present the following proposition that relates $\bar{\epsilon}_t(i, j)$ to the updated forward and backward variables in order to exploit the recursions given in Proposition 1 in the estimation of the HMM parameters in our new framework.

Proposition 2: With the definitions in (4.4) and (4.5), we have

$$\begin{aligned}\bar{\epsilon}_t(i,j) &= P(Z_t = s_i, Z_{t+1} = s_j | \mathbf{Y} = \mathbf{y}, \mathbf{X} = \mathbf{x}, \lambda) \\ &= \frac{B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} \bar{\alpha}_t(i) \bar{\beta}_{t+1}(j)}{P(\mathbf{Y} = \mathbf{y}, \mathbf{X} = \mathbf{x}|\lambda)},\end{aligned}$$

where $\nu(x_{t+1}, s_j) = 1_{\{x_{t+1}=s_0\}}(1-\tau) + 1_{\{x_{t+1}=s_j\}}\tau p + 1_{\{x_{t+1}\neq s_j \wedge x_{t+1}\neq s_0\}}\frac{\tau(1-p)}{N_s-1}$, $s_j \neq s_0$.

Proof: Splitting the observations as $\mathbf{y} = (\mathbf{y}_1^t, y_{t+1}, \mathbf{y}_{t+2}^T)$, and the side information as $\mathbf{x} = (\mathbf{x}_1^t, x_{t+1}, \mathbf{x}_{t+2}^T)$, (4.3) yields

$$\begin{aligned}\bar{\epsilon}_t(i,j) =& \frac{P(Z_{t+1} = s_j, \mathbf{X}_{t+2}^T = \mathbf{x}_{t+2}^T, \mathbf{Y}_{t+2}^T = \mathbf{y}_{t+2}^T|\lambda)}{P(\mathbf{Y} = \mathbf{y}, \mathbf{X} = \mathbf{x}|\lambda)} \times \\ & P(Z_t = s_i, \mathbf{X}_1^{t+1} = \mathbf{x}_1^{t+1}, \mathbf{Y}_1^{t+1} = \mathbf{y}_1^{t+1} | Z_{t+1} = s_j, \mathbf{X}_{t+2}^T = \mathbf{x}_{t+2}^T, \mathbf{Y}_{t+2}^T = \mathbf{y}_{t+2}^T, \lambda).\end{aligned}$$

Since $(Z_t, \mathbf{X}_1^{t+1}, \mathbf{Y}_1^{t+1})$ is independent with $(\mathbf{X}_{t+2}^T, \mathbf{Y}_{t+2}^T)$ conditioned on (Z_{t+1}, λ), we obtain

$$\begin{aligned}\bar{\epsilon}_t(i,j) =& \frac{P(Z_{t+1} = s_j, \mathbf{X}_{t+2}^T = \mathbf{x}_{t+2}^T, \mathbf{Y}_{t+2}^T = \mathbf{y}_{t+2}^T|\lambda)}{P(\mathbf{Y} = \mathbf{y}, \mathbf{X} = \mathbf{x}|\lambda)} \times \\ & P(Z_t = s_i, \mathbf{X}_1^{t+1} = \mathbf{x}_1^{t+1}, \mathbf{Y}_1^{t+1} = \mathbf{y}_1^{t+1} | Z_{t+1} = s_j, \lambda),\end{aligned}$$

which, re-arranging the conditional probabilities, yields

$$\begin{aligned}\bar{\epsilon}_t(i,j) =& \frac{P(Z_t = s_i, Z_{t+1} = s_j, \mathbf{X}_1^{t+1} = \mathbf{x}_1^{t+1}, \mathbf{Y}_1^{t+1} = \mathbf{y}_1^{t+1}|\lambda)}{P(\mathbf{Y} = \mathbf{y}, \mathbf{X} = \mathbf{x}|\lambda)} \times \\ & P(\mathbf{X}_{t+2}^T = \mathbf{x}_{t+2}^T, \mathbf{Y}_{t+2}^T = \mathbf{y}_{t+2}^T | Z_{t+1} = s_j, \lambda) \\ =& \frac{P(Z_{t+1} = s_j, X_{t+1} = x_{t+1}, Y_{t+1} = y_{t+1} | Z_t = s_i, \mathbf{X}_1^t = \mathbf{x}_1^t, \mathbf{Y}_1^t = \mathbf{y}_1^t, \lambda)}{P(\mathbf{Y} = \mathbf{y}, \mathbf{X} = \mathbf{x}|\lambda)} \times \\ & P(Z_t = s_i, \mathbf{X}_1^t = \mathbf{x}_1^t, \mathbf{Y}_1^t = \mathbf{y}_1^t|\lambda) P(\mathbf{X}_{t+2}^T = \mathbf{x}_{t+2}^T, \mathbf{Y}_{t+2}^T = \mathbf{y}_{t+2}^T | Z_{t+1} = s_j, \lambda).\end{aligned}$$

Since $(Z_{t+1}, X_{t+1}, Y_{t+1})$ and $(\mathbf{X}_1^t, \mathbf{Y}_1^t)$ are independent conditioned on (Z_t, λ), and

recognizing the terms $\bar{\alpha}_t(i)$ and $\bar{\beta}_{t+1}(j)$, we obtain

$$\begin{aligned}\bar{\epsilon}_t(i,j) &= \frac{P(Z_{t+1}=s_j, X_{t+1}=x_{t+1}, Y_{t+1}=y_{t+1}|Z_t=s_i,\lambda)\bar{\alpha}_t(i)\bar{\beta}_{t+1}(j)}{P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)} \\ &= \frac{P(X_{t+1}=x_{t+1}, Y_{t+1}=y_{t+1}|Z_{t+1}=s_j, Z_t=s_i,\lambda)}{P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)}\times \\ & P(Z_{t+1}=s_j|Z_t=s_i,\lambda)\bar{\alpha}_t(i)\bar{\beta}_{t+1}(j),\end{aligned}$$

wherein, Markov Property is used to reach

$$\bar{\epsilon}_t(i,j) = \frac{P(X_{t+1}=x_{t+1}, Y_{t+1}=y_{t+1}|Z_{t+1}=s_j,\lambda)P(Z_{t+1}=s_j|Z_t=s_i,\lambda)\bar{\alpha}_t(i)\bar{\beta}_{t+1}(j)}{P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)}.$$

Since X_{t+1} is independent with Y_{t+1} conditioned on (Z_{t+1},λ), we obtain

$$\begin{aligned}\bar{\epsilon}_t(i,j) &= \frac{P(Y_{t+1}=y_{t+1}|Z_{t+1}=s_j,\lambda)P(X_{t+1}=x_{t+1}|Z_{t+1}=s_j,\lambda)}{P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)}\times \\ & P(Z_{t+1}=s_j|Z_t=s_i,\lambda)\bar{\alpha}_t(i)\bar{\beta}_{t+1}(j).\end{aligned}$$

Then, due to the definition of the probability of error event in the side information, we get the proposition as

$$\bar{\epsilon}_t(i,j) = \frac{B_{jy_{t+1}}\nu(x_{t+1},s_j)A_{ij}\bar{\alpha}_t(i)\bar{\beta}_{t+1}(j)}{P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}|\lambda)}.\blacksquare$$

Based on the new set of equations as well as the recursions defined in Proposition 1, we incorporated possibly corrupted side information into the HMM training framework. We finally point out that the forward and backward variables tend to 0 exponentially [100, 111] as we are provided longer sequences, i.e., $\bar{\alpha}_T(i) \to 0$, and $\bar{\beta}_1(i) \to 0, \forall i$, as $T \to \infty$. This would create in practice stability issues, i.e., numeric underflow, on any computer if the recursions in Proposition 1 were directly evaluated. Hence, we propose to use the following scaling scheme of [111] in order to avoid such issues. Let us define the normalization factor c_t

$$c_t = \frac{1}{\sum_{i=1}^{N_s}\bar{\alpha}_t(i)} \text{ and } \sum_{i=1}^{N_s} c_t\bar{\alpha}_t(i) = 1,$$

then we normalize the forward variable $\bar{\alpha}_t(i)$ as $c_t\bar{\alpha}_t(i)$ at each time t after it is calculated with respect to the recursions given in Proposition 1. Similarly, we also normalize the backward variable $\bar{\beta}_t(i)$ with c_t as $c_t\bar{\beta}_t(i)$ at each time t. Then it is

easy to see that the re-estimation formulas, i.e., whether they are computed with the normalized or unnormalized forward and backward variables, remain intact. Namely, consider the re-estimation formulas for the state transition probabilities calculated with the normalized forward and backward variables as

$$\begin{aligned}\hat{A}_{ij} &= \frac{\sum_{t=1}^{T-1} \bar{\epsilon}_t(i,j)}{\sum_{t=1}^{T-1} \bar{\gamma}_t(i)} \\ &= \frac{\sum_{t=1}^{T-1} B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} C_t \bar{\alpha}_t(i) D_{t+1} \bar{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N_s} B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} C_t \bar{\alpha}_t(i) D_{t+1} \bar{\beta}_{t+1}(j)},\end{aligned}$$

where $C_t = \prod_{i=1}^{t} c_i$ and $D_{t+1} = \prod_{i=t+1}^{T} c_i$. Hence, noting that $C_t D_{t+1} = \prod_{i=1}^{T} c_i, \forall t$,

$$\begin{aligned}\hat{A}_{ij} &= \frac{\sum_{t=1}^{T-1} \bar{\epsilon}_t(i,j)}{\sum_{t=1}^{T-1} \bar{\gamma}_t(i)} \\ &= \frac{\prod_{i=1}^{T} c_i \sum_{t=1}^{T-1} B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} \bar{\alpha}_t(i) \bar{\beta}_{t+1}(j)}{\prod_{i=1}^{T} c_i \sum_{t=1}^{T-1} \sum_{j=1}^{N_s} B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} \bar{\alpha}_t(i) \bar{\beta}_{t+1}(j)} \\ &= \frac{\sum_{t=1}^{T-1} B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} \bar{\alpha}_t(i) \bar{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N_s} B_{jy_{t+1}} \nu(x_{t+1}, s_j) A_{ij} \bar{\alpha}_t(i) \bar{\beta}_{t+1}(j)}.\end{aligned}$$

Similarly, the estimates for the conditional observation probabilities $\hat{B}_{ij}$ as well as the initial state probabilities $\hat{\pi}_i$ also remain exact with this normalization scheme. Therefore, this normalization provides the numerical stabilization of the proposed estimation method. In the next section, we provide examples that demonstrate the performance of the new set of training updates under different scenarios.

4.3 Experiments

In this section, we demonstrate the performance of our method through simulations using data generated with the following HMM parameters:

$N_s = 3$, $N_v = 3$, $\pi = \begin{bmatrix} 0.3 & 0.3 & 0.4 \end{bmatrix}$, and

$$A = \begin{bmatrix} 0.8 & 0.19 & 0.01 \\ 0.01 & 0.8 & 0.19 \\ 0.19 & 0.01 & 0.8 \end{bmatrix} \quad B = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.6 & 0.3 \\ 0.3 & 0.1 & 0.6 \end{bmatrix}.$$

For these simulations, we generate a test sequence of length 500 and a training sequence of length 250 along with the side information of a relatively high noise level, $1-p_{\text{true}} = 0.4$, and a relatively low noise level, $1-p_{\text{true}} = 0.2$, with τ ranging from 0 to 0.6. We emphasize that the exact noise level may not be known by the algorithm. Hence, we provide p_{train} to the algorithm which may not be equal to the p_{true}. Here, the parameter p_{train} reflects the confidence (equivalently the expected noise level) that we have on the side information. Since this confidence on the side information might not be accurate, i.e., p_{train} does not necessarily match with p_{true}, for analyzing the sensitivity of our method to the confidence parameter, we train our algorithm with different choices for p_{train}: (1) we set confidence that is in the proximity of p_{true} ($p_{\text{train}} \sim p_{\text{true}}$), i.e., $p_{\text{train}} \in \{0.55, 0.6, 0.65\}$, if $p_{\text{true}} = 0.6$ and $p_{\text{train}} \in \{0.75, 0.8, 0.85\}$, if $p_{\text{true}} = 0.8$, (2) we set too high confidence on the side information ($p_{\text{train}} \gg p_{\text{true}}$), i.e., $p_{\text{train}} = 1$, when $p_{\text{true}} \in \{0.6, 0.8\}$ and, (3) we set too low confidence ($p_{\text{train}} \ll p_{\text{true}}$), i.e., $p_{\text{train}} = 0.5$, when $p_{\text{true}} = 1$. Using the training sequence, we first estimate the unknown model parameters, A_{ij}, B_{ij}, and π_{ij}. Then, on the test sequence, the hidden state sequence is estimated by the Viterbi algorithm [112,113] using the estimated model parameters. We apply our method on 500 different pairs of test and training sequences and we present the resulting average state recognition error rates for all the cases aforementioned. In order to show the efficacy of incorporating the side information by our method, we compare the state recognition error rates of our algorithm with: (1) Baseline Performance, the state recognition error rate if the model parameters are estimated by the ordinary HMM parameter estimation. This is the performance, which is readily achievable with no side information. (2) The Oracle, the state recognition error rate if the true model parameters are directly used in the state estimation on the test sequence. This is the performance limit if the HMM training algorithm is run on infinite amount of training data, which is only asymptotically achievable. (3) Limit of Algorithm, the state recognition error rate if the side

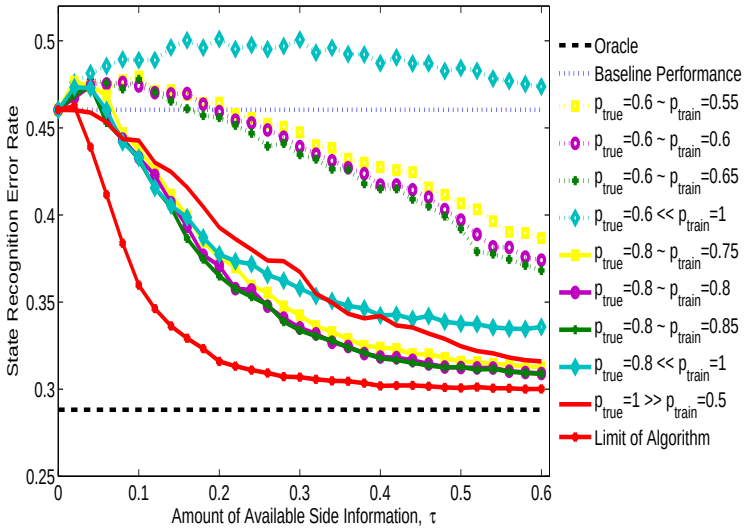


Figure 4.2: Simulation results for various scenarios. Our algorithm is trained with $p_{\text{train}} \in \{0.55, 0.60, 0.65\}$ when $p_{\text{true}} = 0.60$ and $p_{\text{train}} \in \{0.75, 0.80, 0.85\}$ when $p_{\text{true}} = 0.80$. The State Recognition Error Rates are estimated by the Viterbi algorithm. Performance of our algorithm is compared against three performance limits: (1) Baseline Performance, error rate by ordinary HMM using no side Information, (2) Oracle, error rate if the true model parameters are used in state recognition, and (3) Limit of Algorithm, $p_{\text{train}} = p_{\text{true}} = 1$.

information is completely accurate and the algorithm is trained with complete confidence on the side information, i.e., $p_{\text{true}} = 1$, $p_{\text{train}} = 1$. Finally, this is the performance limit that our algorithm can gain at most by exploiting the side information. Here, we name the difference between the Baseline Performance and the Oracle as the "achievable margin" since no algorithm can obtain state recognition improvements more than the achievable margin, provided that, as in this work, first the model parameters are estimated and then used in the Viterbi algorithm for state recognition.

Our simulations show that the performance of our method, provided that $p_{\text{train}} \sim p_{\text{true}}$, improves with the amount of side information that is indicated by τ. In particular, when we have accurate access to the hidden states, i.e.,

$p_{\text{true}} = p_{\text{train}} = 1$, the state recognition rate in the test sequence, labeled as Limit of Algorithm in Fig. 4.2, consistently approaches to the Oracle as τ increases and reaches $\sim 90\%$ gain (the performance improvement over the baseline corresponds to $\sim 90\%$ of the achievable margin) with 30% additional information on states, i.e., $\tau = 0.3$, as shown in Fig. 4.2. This proves the efficacy of our method with incorporating the side information. On the other hand, in the case of noisy access to the hidden states such that 20% of the state observations are mislabeled, i.e., $p_{\text{true}} = 0.8$, our method (when $p_{\text{train}} \sim p_{\text{true}}$) is able to provide substantial gain, 70%, at $\tau = 0.3$. In this case, as τ increases, the recognition approaches to Limit of Algorithm showing that our algorithm optimally incorporates the side information under noise asymptotically. Even if the noise level is further increased up to a level as high as 40% mislabeling, we still obtain a gain that consistently increases with τ, when $p_{\text{train}} \sim p_{\text{true}}$. Thus, our method is robust to noise. Nevertheless, the algorithm must not rely on the side information with too high confidence. Specifically, when we have the confidence $p_{\text{train}} = 1$ in case of high noise level, i.e., $p_{\text{true}} = 0.6$, we do not obtain any improvement compared to the baseline. On the contrary, the algorithm does not fully exploit the side information to its limit, if the confidence is too low. For instance, in case of $p_{\text{train}} = 0.5$ and $p_{\text{true}} = 1$, the rate of performance improvement with τ is significantly slower than that of Limit of Algorithm, i.e., $p_{\text{true}} = 1$, $p_{\text{train}} = 1$. According to our simulations, setting the confidence in the proximity of the true noise level is sufficient to obtain the maximum gain, i.e., our algorithm does not require an exact match between p_{true} and p_{train}. This demonstrates that our algorithm is also robust to the mismatches in the confidence parameter.

4.4 Discussion

We introduce a new parameter estimation algorithm for HMM, when we have partial and noisy access to the hidden state sequence as side information. This side information can be seen as partial labeling, "possibly wrong", of the hidden states. In this work, we model mislabeling and partial labeling of the hidden states jointly in one complete framework. This framework naturally recovers the

unsupervised HMM training if the partial access to the hidden states is turned off. In our simulations, we observe that, using this side information, we considerably improve the state recognition performance, up to 70%, with respect to the "achievable margin". Moreover, our method is shown to be robust to the training conditions. Finally, since this framework includes possible mislabeling events, our algorithm models realistic training conditions more accurately than the ordinary HMM training, i.e., it will provide the same performance in other examples.

In this chapter, we study discrete corruptions (state errors) under a model knowledge (HMM) in the batch setting (a training data assumed). In the following, we generalize to continuous corruptions and drop the model assumption.

Chapter 5

Data Imputation through the Identification of Local Anomalies

In many applications from a wide variety of fields, the data to be processed can partially (or even almost completely) be affected by severe noise in several phases, e.g., occlusions during a visual recording or packet losses during transmission in a communication channel. Such partial, i.e., localized, data corruptions often severely degrade the performance of the target application; for instance, face recognition or pedestrian detection under occlusion [5, 114–117]. In order to reduce the impact of this adverse effect, we develop a complete and novel framework, which efficiently detects, localizes and imputes corruptions by identifying the local anomalies in a given suspicious data instance. We emphasize that neither the existence nor, if exists, the location of a corruption is known in our framework. Moreover, the proposed algorithms do not assume a model but operate in a data driven manner.

We consider the local corruptions as statistical deviations from the nominal distribution of the uncorrupted (clean) observations. To detect and localize corruptions, i.e., such statistical deviations, we model a corruption as an anomaly due to an external factor (communication failure in a channel or occluder object in an image), which locally overwrites a data instance and moves it outside the

support of the nominal distribution. However, corruptions that we consider as examples of anomalies have further specific properties such that (a) the corruptions in an instance are confined to unknown intervals along the data attributes, i.e., localized, and (b) not only a corrupted part but also all of its subparts are anomalous. Thus, a corruption does not provide an anomaly due to an incompatible combination of normal subparts. Based on these properties that accurately model a wide variety of real life applications, we characterize the event of corruption and formulate the corresponding detection/localization as an anomaly detection problem, cf. [25, 26, 118–122].

The introduced algorithm applies a series of statistical tests with a pre-specified false alarm rate to the parts of the suspicious instance after extracting the nominal statistics from a reference (training) data set of uncorrupted (clean) observations. As a result, each part is labeled as anomalous/normal and the local anomalies are identified. These parts are generated and organized through a binary tree partitioning of the data attributes, each node of which corresponds to a part of the suspicious instance, cf. Fig. 5.1. Once the nodes (or parts) are labeled as anomalous/normal on this tree, the patterns of corruption are identified using the aforementioned characterization to detect and localize corruptions, cf. Fig. 5.2. We point out that this localization procedure transforms the nominal distribution into a multivariate Bernoulli distribution with a success probability that precisely coincides with the constant false alarm rate of the local anomaly tests. Considering the hierarchy among the binary labels implied by the tree as a directed acyclic graph, the resulting multivariate Bernoulli distribution achieves a certain dependency structure. Under this condition, we derive the false alarm rate of the proposed framework in detecting the corruptions and show that it is a constant rate, namely, no parameter tuning is required to achieve the desired/specified false alarm rate even if the data change.

If a corruption is localized, then we impute/replace the affected attributes with the estimates of the underlying unknown true attributes. For this purpose, we additionally develop a novel *Maximum A Posteriori* (MAP) estimator using the "score function" defined in [120]. Our estimator exploits the local dependencies among the data attributes, where the locality is encoded in the binary partitioning

tree. We point out that the implementation of this MAP estimator does not load extra computational cost since it utilizes the outputs of our anomaly detection approach, which are computed prior to the imputation phase. Furthermore, we also propose a novel distance measure named "ranked Euclidian distance" as a generalization to the standard Euclidean distance, which is used in the course of the labeling of each part as anomalous/normal. The proposed distance measure is compared with the standard Euclidean distance in the experiments and shown to be superior in terms of detecting and localizing corruptions.

We conduct tests over several well-known machine learning data sets [123, 124], which are exposed to severe data corruptions. Our experiments indicate that the proposed framework achieves significant improvements after imputation up to 80% in terms of the classification purposes and outperforms the typical approaches. The proposed algorithms are also empirically shown to be robust to varying training phase conditions with strong corruption separation capabilities.

In this study, the corrupted attributes are considered to be statistically independent with the underlying unobserved true data, i.e., corrupted attributes are of no use in estimation of the uncorrupted counterparts. Hence, if one knows which attributes are corrupted in an instance, then those attributes can readily be treated as missing data, cf. [105, 125–129]. For example, classification and clustering with missing data is a well-studied problem in the machine learning literature. The corresponding studies such as [105,127,128,130,131] are related to inference with incomplete data [127] and generative models [130], where Bayesian frameworks [128] are used for inference under missing data conditions. Alternatively, pseudo-likelihood [132] and dependency network [133] approaches solve the data completion problem by learning conditional distributions. In [134], the probability density of the missing data is modeled conditioned on a set of introduced latent variables and thereafter, a MAP based inference is used. However, all of the studies [105, 125–128, 130–134] either assume the knowledge of the location of the missing attributes or impose strong modeling constraints; as opposed to our model free solutions.

On the other hand, imputation is commonly used as a pre-processing tool [128].

The Mixture of Factor Analyzers [135] approach replaces the missing attributes with samples drawn from a parametric density, which models the distribution of the underlying true data. Whereas the proposed imputation techniques in [136, 137] are both non-parametric and based on the inference of the posterior densities via certain kernel expansions. On the contrary, the MAP estimator in this study does not even attempt to estimate the posterior density either in a parametric or non-parametric manner. Instead, the introduced method is only based on the sufficient rank statistics. We emphasize that unlike our approach, the incomplete data approaches generally assume the knowledge of the missing attributes, i.e., they are precisely localized and provided beforehand. For example, the occluded pixels in the event of occlusion of a target object in an image cannot be known a priori, which requires a detection and localization step. Since the existing studies do not have such a step, an exhaustive list of the occluded pixels as the result of a manual inspection of the missing attributes is required as an input to the algorithms proposed in the corresponding literature. In this regard, our study is the first to jointly handle the issues of detecting/localizing missing attributes, i.e., corruptions, as well as their imputation in a single, complete and comprehensive framework. Hence, the generic local corruption detection and imputation algorithm of our framework complements the missing data imputation approaches as an additional merit.

Data imputation and completion is also essential in image processing for handling corrupted images, e.g., [138, 139]. Generally, a corrupted image is restored by explicitly learning the image statistics [140, 141] or by using neural networks [142–144]. These denoising studies do not attempt to localize corruptions in an image, but treat them as a noise and filter it out using statistical approaches applied to the image globally. Even though this is a valid approach for image enhancement, an attempt to correct/enhance an image globally in case of only a localized corruption might be even detrimental since the uncorrupted parts are also affected by global operations. Additionally, it is not usually possible to locally impute corrupted portions using denoising approaches. There exist several studies that aim localization as well. Studies such as [114, 117] indicate

that occlusion, as an example of corruption, is a common phenomenon and detrimental in pedestrian detection as well as face recognition applications. In this regard, detection of occluded, i.e., corrupted, visual objects had been previously investigated in a number of studies [145–148]. In these studies, occlusion detection is performed using domain specific knowledge (visual cues) or external information (object geometry), which, however, are not always available in general data imputation setting. From the machine learning perspective, descriptors are extracted from various parts of the occluded object in [149] and similarly; part-based descriptors are weighted with the occlusion measure in [150] to relieve the corresponding degrading effects. Since these approaches do not directly target handling occlusions, i.e., corruptions, they only provide partial or limited solutions. Several other studies propose solutions via extracting occlusion maps, e.g., [151, 152]. In [151], HOG based classification errors; and in [152], template based reconstruction errors are used to generate such an occlusion map. However, both studies assume rigid models and significantly rely on domain specific knowledge; and in general fail to remain applicable if the data source belongs to another domain. In this study, we assume that data is generic and no domain information is available, yet detection and imputation of corruption is necessary for improving the subsequent processing stages, such as classification.

Summary of Contributions:

1. This is the first study that jointly handles localized data corruptions in a single, complete and comprehensive statistical framework that is designed completely model free for the goal of separating a corruption and imputing the affected data attributes. We also provide a false alarm rate (in detecting corruptions) analysis of the framework via directed acyclic graphs.

2. A novel MAP estimator for data imputation and a novel distance measure for corruption localization purposes are proposed.

3. The proposed framework is computationally efficient in the sense that (i) it effectively utilizes a binary search for corruption separation, and (ii) the computational load due to our MAP based imputation is insignificant.

4. We propose a novel characterization for anomalies, e.g., rarities, incompatible combinations and corruptions.

In Section 5.1, we provide the problem description. We then present our algorithm in Section 5.2 and the associated computational complexity in Section 5.3. We report the corruption detection/localization performance of the proposed algorithm as well as the improvement in classification tasks achieved by the imputation in Section 5.4. The chapter concludes with a discussion in Section 5.5.

5.1 Problem Description

We have a possibly corrupted test instance $\mathbf{x} \in \mathbb{R}^d$ along with a set of uncorrupted (clean) independent and identically distributed observations $S = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{N_s}\}$ as the nominal training (reference) data, where $\mathbf{s}_i = [s_{i1}, s_{i2}, ..., s_{id}] \in \mathbb{R}^d \sim f_0(\mathbf{s})$, d is data dimensionality and f_0 is the unknown nominal density. The test instance $\mathbf{x}$ is considered to be corrupted with probability π by severe noise in multiple non-overlapping intervals along its dimensions (attributes), which are completely unknown. Suppose that for such an interval, the corruption is localized and confined to the attributes $\mathbf{x}_c^{c+\beta-1} = \{x_c, x_{c+1}, ..., x_{c+\beta-1}\}$ for some c and β in $[1, d]$ with $c + \beta - 1 \leq d$. We assume that the corrupted attributes are uniformly and independently distributed, $z_i \in \mathbf{x}_c^{c+\beta-1} \sim U_Z(z)$, where U_Z is the uniform distribution defined in a finite support. Moreover, Z is also statistically independent with the true data and hence, the knowledge of $\mathbf{x}_c^{c+\beta-1}$ is irrelevant to the uncorrupted counterparts. Note that this corruption model implies a total erasure of data in several unknown portions due to an independent source overwriting the attributes in those portions, e.g., an occluder in computer vision applications [114,117]. Typically, since no information is provided about the independent source in such applications, we consider that the uniformity assumption draws a worst case scenario and it is realistic. On the other hand, $\mathbf{x}$ is considered to be uncorrupted with probability $1 - \pi$. Therefore, whether a test instance $\mathbf{x}$ includes a corruption is unknown; and it is generally modeled to be drawn from

the mixture $\mathbf{x} \sim (1-\pi)f_0(\mathbf{x}) + \pi f_1(\mathbf{x})$ [120], where f_1 is the probability density of the corrupted instances.

The density f_1 can be derived from the unknown nominal density f_0 using the described corruption model, if the distributions of c, β and the number of corrupted intervals are further specified; which is unnecessary in the context of this study. Hypothetically, if one can correct an instance $\mathbf{x}$ drawn from the density f_1 by replacing all the corrupted attributes, e.g., $\mathbf{x}_c^{c+\beta-1}$, with the underlying true attributes, e.g., $\bar{\mathbf{x}}_c^{c+\beta-1}$, and obtain $\hat{\mathbf{x}}$, then $\hat{\mathbf{x}}$ should follow the nominal density f_0. Similarly, if the corruptions in $\mathbf{x}$ can be localized, then the corresponding portions would follow the multivariate uniform density $U_{\mathbf{Z}}(\mathbf{z})$ of the appropriate dimensionality. On the other hand, this corruption model potentially creates significant statistical deviations from the reference data since a corrupted observation $\mathbf{x} \sim f_1$ and f_1, in general, increasingly diverges from f_0 as the corruption strength increases. Here, the corruption strength can be considered as the number of corrupted attributes and/or the variance of the corruption $U_Z(z)$ that overwrites the true data. Furthermore, our modeling of corruptions poses a missing (incomplete) data problem since the unknown true attributes $\bar{\mathbf{x}}_c^{c+\beta-1}$ in a corrupted interval are statistically irrelevant to the corrupted attributes $\mathbf{x}_c^{c+\beta-1}$. In this study, by exploiting the statistical deviations from the nominal distribution of observations, we aim to detect and localize the possible corruptions in a given instance $\mathbf{x}$ and impute the corrupted or missing attributes.

To this end, we formulate an anomaly detection approach to define this framework in Section 5.2, where we draw the distinctions among several examples of anomalous observations and separate the event of corruption. Then, we propose our algorithm and analyze the associated false alarm probability in detecting corruptions as well as the computational complexity.

5.2 A Novel Framework for Corruption Detection, Localization and Imputation

In this section, we develop a novel framework for a complete treatment of possible corruptions in the input data $\mathbf{x}$. For presentational clarity and without loss of generality, we assume that the input data $\mathbf{x}$ can be corrupted only in a single interval throughout this section. Note that the generalization to the case of corruptions spread onto several intervals is immediate and indeed, we present a corresponding detailed experiment in Section 5.4. Since the corruptions are modeled as local statistical deviations within this framework, we give a brief description of the anomaly detection approach that we work with in Section 5.2.1. Based on the characterization of corruptions through their distinctive properties in Section 5.2.2, we present Algorithm TCS (Tree-based Corruption Separation). After we derive a novel MAP estimator for imputation in Section 5.2.3, we derive the false alarm rate of the proposed framework in detecting the corruptions in Section 5.2.4.

5.2.1 Detection of Statistical Deviations: Anomalies

A localized corruption is considered to affect an instance in a certain part(s) such that the affected attributes statistically deviate from the vast majority of the data. The proposed algorithm in this study localizes the corrupted attributes by identifying the local anomalies through a series of statistical checks of the test instance with the reference data. In this section, we briefly describe the anomaly detection approach that we work with and present a novel distance measure for the corruption localization purpose.

The probability density of a possibly corrupted test instance $\mathbf{x}$ can be modeled as

$$\mathbf{x} \sim (1-\pi) f_0(\mathbf{x}) + \pi f_1(\mathbf{x}), \text{ where}$$

$H_0 : \mathbf{x} \sim f_0(\mathbf{x})$ is the null hypothesis from which the nominal data are drawn,

$H_1 : \mathbf{x} \sim f_1(\mathbf{x})$ is the hypothesis representing the corrupted observations, and $\pi \in [0, 1]$ is the corresponding mixing coefficient. Within the framework of anomaly detection approaches, the nominal distribution f_0 is usually assumed unknown or hard to estimate; and instead, a set of nominal observations is provided. Then for a given test instance $\mathbf{x}$, the task in [120] is to decide whether the null hypothesis H_0 was realized or the alternative H_1 such that the detection rate (of anomalies) is maximized with a constant false alarm rate τ. For this purpose, the score function [120]

$$\hat{p}_K(\mathbf{x}) = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{1}_{\{R_S(\mathbf{x};K) \leq R_S(\mathbf{s}_i;K)\}} \tag{5.1}$$

is proposed, where $\mathbf{1}_{\{.\}}$ is the indicator function and $R_S(\mathbf{x}; K)$ is the Euclidean distance from $\mathbf{x}$ to its nearest K'th neighbor in S, if $\mathbf{x} \notin S$; and to its nearest $K+1$'th neighbor in S, otherwise. Based on this score function, the test instance $\mathbf{x}$ is declared as anomalous [120], if

$$\hat{p}_K(\mathbf{x}) \leq \tau. \tag{5.2}$$

When the mixing distribution f_1 is assumed uniform, it is shown in [120] that $\hat{p}_K(\mathbf{x})$ is an asymptotically consistent estimator of the density level of the test instance

$$p(\mathbf{x}) = \int_{\forall \mathbf{s}} \mathbf{1}_{\{f_0(\mathbf{x}) \geq f_0(\mathbf{s})\}} f_0(\mathbf{s}) d\mathbf{s} \tag{5.3}$$

under certain smoothness conditions. Remarkably, $\{x : p(x) \geq \tau\}$ provides the minimum volume set at level τ, which is the most powerful decision region for testing H_0 vs H_1 with a constant false alarm rate τ [26]. We note that the precision of the test defined in (5.2) degrades faster with the dimensionality than it improves with the size of the training data. As a result, we here point out several practical issues about detecting the existence of a corruption with this approach.

Briefly, i) a direct test of an instance $\mathbf{x}$ does not localize a possible corruption for imputation, ii) on the contrary, a truly corrupted instance, i.e., an instance of hypothesis H_1, does not necessarily test positive due to the limited training

data, high dimensionality as well as that the corruption might not be sufficiently strong; and iii) corruptions have further specific properties in addition to that they provide anomalies, which must be incorporated to achieve a better false alarm rate compared to τ.

Ranked Euclidean Distances: To address the first issue in this list, we propose a novel distance measure (not a metric in the mathematical sense), which is sensitive to only a certain α fraction of the attributes for a given pair of instances $\mathbf{x}$ and $\mathbf{y}$. For instance, a corruption of only a single attribute in a given test instance $\mathbf{x}$ might be significantly strong such that the whole instance turns anomalous with the test in (5.2) used with the standard Euclidean distance. In this case, any part of the instance $\mathbf{x}$ including the corrupted attribute would test positive, which creates an ambiguity in terms of the localization, i.e., separation, of the corrupted attribute, and in turn requires an exhaustive search over all possible subsets in the space of the attributes.

To overcome such ambiguities, we propose a distance measure so that the test in (5.2) results positive only when the corruption has a sufficiently large support, which disregards a pre-specified fraction of the attributes that are most responsible for a possible corruption. We define this measure for an $\alpha \in [0, 1]$ as

$$h_\alpha(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{\lfloor d\alpha \rfloor} (x_{k(i)} - y_{k(i)})^2}, \tag{5.4}$$

where k is a permutation of the attributes with

$$|x_{k(1)} - y_{k(1)}| \leq ... \leq |x_{k(i)} - y_{k(i)}| \leq ... \leq |x_{k(d)} - y_{k(d)}|,$$

and $\lfloor . \rfloor$ is the floor operator. Since this distance measure depends only on the α fraction of the least deviated attributes between $\mathbf{x}$ and $\mathbf{y}$, a corruption must have a support of at least $(d - \lfloor d\alpha \rfloor)$-length to make an instance anomalous with respect to the reference data. Here, $(1 - \alpha)$ can be seen as the precision of the localization when an anomalous instance is checked with the test in (5.2) using the distance measure defined in (5.4). This precision obviously cannot be made arbitrarily large since as $1 - \alpha$ approaches 1, the distance h_α becomes more prone

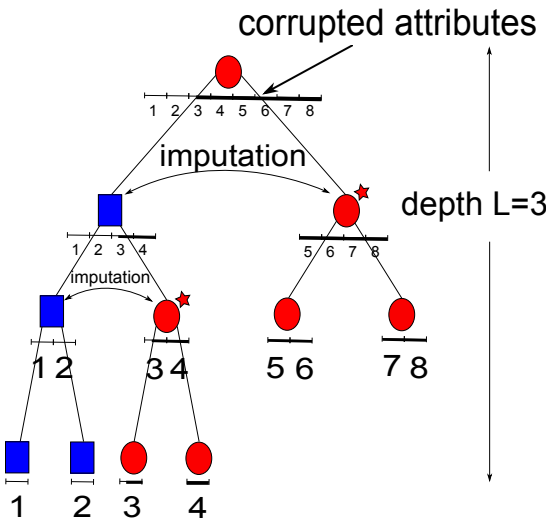


Figure 5.1: Algorithm TCS (Tree-based Corruption Separation) with $\alpha = 0.5$

to noise and the correlation structure between the attributes is less exploited. We investigate this trade-off further in our simulations. The distance measure h_α recovers the standard Euclidean distance when $\alpha = 1$ and will be named in the rest of the chapter as "ranked Euclidean distance". We note that for the cases $\alpha < 1$, h_α fails to be a metric in the mathematical sense, i.e., $h_\alpha(x, y) = 0 \Leftrightarrow x = y$ is not satisfied, which requires to specify a nominal density model on f_0 to derive the same asymptotic consistency in [120] for the score values $\hat{p}_K(x)$ in estimating the density levels $p(x)$ with h_α. However, we do not assume -in this work- any density model for f_0 or do not take any stochastic assumptions regarding the data source.

In the following section, we characterize the corruptions by presenting their specific properties and propose an algorithm to localize and impute corruptions.

5.2.2 Modeling of Localized Corruptions

If a test instance is subject to corruption in a small part only, the corruption might not be detectable when it is checked using an anomaly detection algorithm without a detailed analysis in its parts. On the other hand, an anomalous observation does not necessarily contain a corruption since it might simply be a false alarm,

in fact an uncorrupted observation. To address these two issues, we propose a statistical analysis of a test instance through its parts using a binary partitioning tree in the space of data attributes on which, we also provide a characterization to separate the event of corruption among possible anomaly scenarios.

Suppose that an instance $\mathbf{x} = [x_1, x_2, ..., x_d] \in \mathbb{R}^d$ corresponds to the root node R on a binary tree. Using half-way splits for presentational simplicity, let the set of attributes $V_{R_l} = \{x_1, x_2, ..., x_{\lfloor \frac{d}{2} \rfloor}\}$ be assigned to the left child node R_l of the root and $V_{R_r} = \{x_{\lfloor \frac{d}{2} \rfloor + 1}, x_{\lfloor \frac{d}{2} \rfloor + 2}, ..., x_d\}$ assigned to the right child node R_r, Fig. 5.1. Note that $V_R = \{x_1, x_2, ..., x_d\}$ with $V_{R_l} \cap V_{R_r} = \emptyset$ and $V_R = V_{R_l} \cup V_{R_r}$. Based on this strategy for generating subparts of an instance, we propose Algorithm TCS (Tree-based Corruption Separation) to separate and impute corruptions, which recursively expands a depth-L binary tree to partition the space of attributes. For each node ν created in the course of this expansion, the corresponding attributes/part of the test instance, e.g., $\mathbf{x}_{V_{R_l}} := \mathbf{x}_1^{\lfloor \frac{d}{2} \rfloor}$ with $\nu = R_l$, is checked whether it is consistent with the reference data restricted to those attributes, e.g., $S_{V_{R_l}} = \{s_1{}_1^{\lfloor \frac{d}{2} \rfloor}, s_2{}_1^{\lfloor \frac{d}{2} \rfloor}, ..., s_{N_s}{}_1^{\lfloor \frac{d}{2} \rfloor}\}$ with $\nu = R_l$, using the test defined in (5.2). We here use the ranked Euclidean distance h_α in this testing with a pre-specified α. Therefore, each node ν encountered in this expansion is assigned a binary label as anomalous/normal and a fully labeled (possibly unbalanced) tree is obtained for the test instance $\mathbf{x}$. We emphasize that Algorithm TCS does not completely construct this depth L-binary tree at the beginning but instead expands it by creating the nodes and the edges as needed to achieve an efficient implementation, which continues until that each data attribute is decided to be corrupted or uncorrupted.

We consider several scenarios where the observation $\mathbf{x}_{V_\nu}$ at a node ν can be anomalous. In Fig. 5.2, the nodes are illustrated as circles, if the corresponding part is found to be anomalous; and squares otherwise. An anomaly can be widespread onto the attributes and consist of anomalous subparts as illustrated in Fig. 5.2a. Since all of the subparts of a corrupted data part are also corrupted by definition, the pattern in Fig. 5.2a is regarded as a conclusive pattern. Hence, a corruption at the starred node in Fig. 5.2a is declared, unless it is the root node. Note that a global corruption at the root is disregarded in this work since it is not

localized. In another case, an anomalous observation could be non anomalous in its parts as illustrated in Fig. 5.2b, which simply happens due to an incompatible or rare combination of attributes in its subparts. This is a typical situation, where an anomalous observation is not corrupted. Hence, this case also provides a conclusive pattern in our consideration such that a corruption is rejected at the anomalous node. On the contrary, the case in Fig. 5.2c is an inconclusive pattern, which suggests a corruption at the right child, however, whether the corruption is spread in the attributes of that child or localized is unknown. Hence, the attributes of the right child is further split and explored similarly. Then, if the conclusive pattern in Fig. 5.2a (or Fig. 5.2b) is realized, then the corruption is accepted and localized (or rejected) at the starred node in Fig. 5.2d. Otherwise, the search continues. On the other hand, if a significantly small subset of the corrupted attributes are left at the left child node in Fig. 5.2c, it might not be detectable and labeled as normal. Then the corresponding attributes should further be split as illustrated in Fig. 5.2d. This process recursively defines a corruption localization with an improved false alarm rate as several anomalies are rejected as they are false alarms, i.e., non corrupted anomalies.

The introduced Algorithm TCS then searches the described binary tree in a breadth-first-search fashion for a corruption. When the conclusive (or terminating) pattern shown in Fig. 5.2a (Fig. 5.2b) is found in the course of this expansion, the search is stopped at the parent node of the found pattern, i.e., the tree is pruned on that branch, and corruption is declared (or no corruption is found and no action is necessary) for the corresponding attributes. This search of corruption at each branch starting from the root node continues to the corresponding leaf node, unless a terminating pattern is found. Finally, if a conclusive pattern is not encountered at a branch from the root to an anomalous leaf, we opt to accept the corruption at the leaf to favor a better detection at a cost of an increased corruption false alarm rate. An illustration of the progress of the algorithm is given in Fig. 5.1, where the corrupted attributes are successfully located. Note that a small set of the attributes are mislabeled as corrupted, i.e., false alarms, in the region 3, which can be corrected if the partitioning resolution is improved by increasing the depth L.

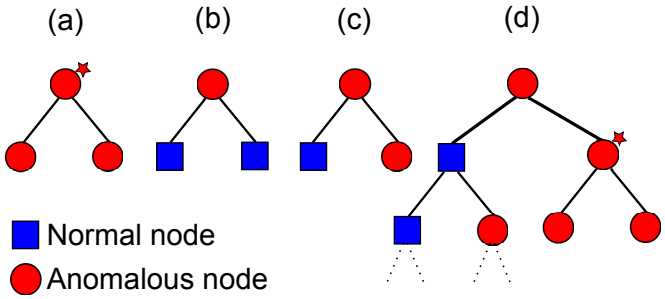


Figure 5.2: An anomalous observation with several scenarios in its parts. Note that the starred nodes indicate localized corruptions. (a) A conclusive pattern: corruption is detected. (b) A conclusive pattern: corruption is rejected. (c) An inconclusive pattern. (d) Further exploration of the test instance.

5.2.3 *Maximum A Posteriori* (MAP) Based Imputation

We emphasize that in most of the detection and estimation applications, the posterior density, e.g., $f_0(\bar{\mathbf{x}}_{V_\nu}|\mathbf{x})$ in (5.5), of the target is too complicated to assume realistic parametric models so that the nonparametric approaches are often favored in such situations [12]. In accordance, we introduce an algorithm that works under a completely model free setting regarding both the localization of the corruptions and the imputation. Furthermore, we point out that when the posterior density is multi-modal, MAP based estimators are generally known to generate more plausible results compared to MMSE based estimators or simple (possibly weighted) averaging [153], which can even generate infeasible solutions [154–156]. This is often the case especially for the computer vision and machine learning applications such as edge preserving image denoising [157]. For instance, the gradients in an occluded pedestrian image would get too smoothed in an MMSE based imputation, which might cause the gradient based feature extractors, e.g., HOG [158], to fail in case of an pedestrian detection application [12, 117]. For these reasons, we propose a novel MAP based imputation technique, which always generates feasible and likely estimates and approximates the true MAP estimator as the size of the reference data increases.

Once a corruption is localized for an instance $\mathbf{x}$ at a node ν, then our task is to estimate the original attributes $\bar{\mathbf{x}}_{V_\nu}$ using the training data set S as well as the instance $\mathbf{x}$ and impute accordingly, i.e., replace the corrupted attributes

in $\mathbf{x}$ with the estimates. Since we assume the corrupted attributes $\mathbf{x}_{V_\nu}$ to be statistically independent with the underlying true data $\bar{\mathbf{x}}_{V_\nu}$, we treat the corrupted attributes as missing data, which then should have no effect in estimation of the true attributes. Hence, we condition this estimation of the data $\bar{\mathbf{x}}_{V_\nu}$ on the remaining attributes in $\mathbf{x}$. On the other hand, we note that in most of the applications such as the image compression [159], the data attributes being in sufficiently close proximity are usually modeled to manifest high correlation. In accordance, we propose to estimate the unknown data $\bar{\mathbf{x}}_{V_\nu}$ conditioned on the attributes $\mathbf{x}_{V_{\nu_s}}$ associated with its nearest neighbor on our tree, i.e., the sibling node ν_s of ν. Note that due to the localization of corruptions by Algorithm TCS (Tree-based Corruption Separation), the attributes at the sibling node ν_s are certainly detected to be uncorrupted in case of the standard Euclidean distance; and detected to be uncorrupted with significantly high probability in case of the ranked Euclidean distance (cf. Section 5.2.4). In the following, we introduce a novel *Maximum A Posteriori* (MAP) estimator of the true data underlying the corrupted attributes based on the standard Euclidean distance (h_α with $\alpha = 1$) and then discuss the generalization over α for the ranked Euclidean distance measure. We also stress that the implementation of this estimator is only based on the outputs of our corruption localization algorithm, which are computed before the imputation phase in the course of Algorithm TCS. Therefore, computationally, the imputation phase that we develop is efficient such that it does not require further computations.

Since the only relevant part of the test instance $\mathbf{x}$ to the proposed MAP estimator is $\mathbf{x}_{V_{\nu_s}}$, we have

$$f_0(\bar{\mathbf{x}}_{V_\nu}|\mathbf{x}) = f_0(\bar{\mathbf{x}}_{V_\nu}|\mathbf{x}_{V_{\nu_s}}), \tag{5.5}$$

where $\bar{\mathbf{x}}_{V_\nu}$ represents a realization of the conditional probability density of the true data underlying the corrupted attributes V_ν. Then, the MAP estimator of $\bar{\mathbf{x}}_{V_\nu}$ maximizes the posterior distribution as

$$\mathbf{x}_{V_\nu}^{MAP} = \arg \sup_{\bar{\mathbf{x}}_{V_\nu} \in \mathbb{R}^{|V_\nu|}} f_0(\bar{\mathbf{x}}_{V_\nu}|\mathbf{x}_{V_{\nu_s}}).$$

For any $\epsilon > 0$, and under certain smoothness constraints on f_0 with $f_0(\bar{\mathbf{x}}_{V_\nu}) \neq 0$,

Algorithm 2 TCS — Tree-based Corruption Separation

Input: $\alpha, K, \tau, L;\ \ S, \mathbf{x}$

Initialize $\mathcal{C} \leftarrow \emptyset$: set of corrupted attributes
Initialize $\mathbf{y} \leftarrow \mathbf{x}$: imputed test data
Create the root node $\nu \leftarrow R$ and label
procedure RECURSE(ν)
    Create nodes ν_l and ν_r; and label
    if the pattern in Fig. 5.2a **then**
        if ν is the root **then return**
        else
            Declare corruption at ν: $\mathcal{C} \leftarrow \mathcal{C} \cup V_\nu$
            Impute attributes V_ν in $\mathbf{y}$
            return
        end if
    else if the pattern in Fig. 5.2b **then return**
    else if ν is a parent of a leaf **then**
        if ν_j ($j = l$ or $j = r$) is anomalous **then**
            Declare corruption at ν_j: $\mathcal{C} \leftarrow \mathcal{C} \cup V_{\nu_j}$
            Impute attributes V_{ν_j} in $\mathbf{y}$
        end if
        return
    else
        RECURSE(ν_l) and RECURSE(ν_r)
    end if
end procedure

Return: $\mathcal{C}$ and $\mathbf{y}$

let

$$B_\epsilon(\bar{\mathbf{x}}_{V_\nu}) \cap S_{V_\nu} \neq \emptyset$$

hold with some probability δ_{N_s}, where $B_\epsilon(\bar{\mathbf{x}}_{V_\nu})$ (w.r.t. the standard Euclidean distance) is the ϵ-ball around $\bar{\mathbf{x}}_{V_\nu}$ in $\mathbb{R}^{|V_\nu|}$ and $N_s = |S|$. Then we point out that

$$\lim_{N_s \to \infty} \delta_{N_s} = 1.$$

Hence, since ϵ can be made arbitrarily small, we obtain

$$\mathbf{x}_{V_\nu}^{MAP} = \arg \lim_{N_s \to \infty} \sup_{\bar{\mathbf{x}}_{V_\nu} \in S_{V_\nu}} f_0(\bar{\mathbf{x}}_{V_\nu} | \mathbf{x}_{V_{\nu_s}}),$$

and by the Bayes' rule

$$\begin{aligned}\mathbf{x}_{V_\nu}^{MAP} &= \arg\lim_{N_s\to\infty}\sup_{\bar{\mathbf{x}}_{V_\nu}\in S_{V_\nu}}\frac{f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{x}_{V_{\nu_s}})}{f_0(\mathbf{x}_{V_{\nu_s}})}\\ &= \arg\lim_{N_s\to\infty}\sup_{\bar{\mathbf{x}}_{V_\nu}\in S_{V_\nu}} f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{x}_{V_{\nu_s}}) \qquad (5.6)\end{aligned}$$

with probability 1, where the denominator is dropped since it does not depend on the maximizer, i.e., $\bar{\mathbf{x}}_{V_\nu}$. To approximate the MAP estimator given in (5.6), we adapt the nonparametric k-nearest neighbor (knn) based density estimation approach [160]. Let us define a small neighborhood around $\mathbf{x}_{V_{\nu_s}}$ in $\mathbb{R}^{|V_{\nu_s}|}$ as

$$\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}}) = \{\mathbf{s} : R_S(x_{V_{\nu_s}};\gamma\sqrt{N_s}) \geq h_{\alpha=1}(\mathbf{x}_{V_{\nu_s}},\mathbf{s})\}, \qquad (5.7)$$

where $h_{\alpha=1}(.,.)$ is the Euclidean distance and $R_S(x_{V_{\nu_s}};\gamma\sqrt{N_s})$ is the $h_{\alpha=1}(.,.)$ distance from $x_{V_{\nu_s}}$ to its nearest $\gamma\sqrt{N_s}$'th neighbor in $S_{V_{\nu_s}}$ for some $\gamma > 0$. Note that as $N_s \to \infty$, $\mathcal{L}(\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})) \to 0$, where $\mathcal{L}(.)$ is the Lebesgue measure. Then (5.6) yields

$$\mathbf{x}_{V_\nu}^{MAP} = \arg\lim_{N_s\to\infty}\sup_{\bar{\mathbf{x}}_{V_\nu}\in S_{V_\nu}}\frac{\int_{\mathbf{z}\in\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})} f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})\mathbf{dz}}{\mathcal{L}(\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}}))},$$

with probability 1. When N_s is sufficiently large with $N_s \geq N_s^*$ for some N_s^* or $\mathcal{L}(\mathcal{N}_{N_s})$ is sufficiently small, we assume that $f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{x}_{V_{\nu_s}})$ is subject to negligible variations only. Then, we (with probability 1) obtain the approximation:

$$\begin{aligned}\mathbf{x}_{V_\nu}^{MAP} &= \arg\lim_{N_s\to\infty}\sup_{\bar{\mathbf{x}}_{V_\nu}\in S_{V_\nu}}\frac{\int_{\mathbf{z}\in\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})} f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})\mathbf{dz}}{\mathcal{L}(\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}}))}\\ &\simeq \arg\max_{\bar{\mathbf{x}}_{V_\nu}\in S_{V_\nu},\mathbf{z}\in\mathcal{N}_{N_s^*}(\mathbf{x}_{V_{\nu_s}})} f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{z}),\end{aligned}$$

where, in order to obtain the corresponding maximum in the reference set S, knowing the rank statistics in $f_0(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})$ is enough, i.e., explicitly estimating/computing the density is unnecessary. Therefore, using the density function defined in (5.3), we obtain

$$\mathbf{x}_{V_\nu}^{MAP} \simeq \arg\max_{\bar{\mathbf{x}}_{V_\nu}\in S_{V_\nu},\mathbf{z}\in\mathcal{N}_{N_s^*}(\mathbf{x}_{V_{\nu_s}})} p(\bar{\mathbf{x}}_{V_\nu},\mathbf{z}). \qquad (5.8)$$

For sufficiently large N_s, note that $\hat{p}_K(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})$ approximates $p(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})$ [120], i.e., $\forall(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})$

$$|\hat{p}_K(\bar{\mathbf{x}}_{V_\nu},\mathbf{z}) - p(\bar{\mathbf{x}}_{V_\nu},\mathbf{z})| \simeq 0 \text{ almost surely .} \qquad (5.9)$$

Using the result in (5.8) in combination with (5.9), we propose to use MAP based estimator of the true data underlying the corrupted attributes

$$\mathbf{x}_{V_\nu}^{MAP} \simeq \hat{\mathbf{x}}_{V_\nu} = \arg \max_{\bar{\mathbf{x}}_{V_\nu} \in S_{V_\nu}, \mathbf{z} \in \mathcal{N}_{N_s^*}(\mathbf{x}_{V_{\nu_s}})} \hat{p}_K(\bar{\mathbf{x}}_{V_\nu}, \mathbf{z}),$$

based on which we replace, i.e., impute, the corrupted attributes $\mathbf{x}_{V_\nu}$ in the instance $\mathbf{x}$ with $\hat{\mathbf{x}}_{V_\nu}$ and obtain the imputed data as $\mathbf{y}$.

This estimator is implemented in Algorithm TCS (Tree-based Corruption Separation) at every node in the tree, where a corruption is detected. Namely, we i) obtain the K neighbors of the test instance in the reference data set S with respect to the attributes associated with the node ν_s; ii) for those neighbors in S, find the one, say s^*, attaining the largest score value defined in (5.1) using the attributes associated with the parent node ν_p; then iii) impute the instance $\mathbf{x}$, which is detected to be corrupted at the node ν, using s^* for the attributes V_ν. In the realistic case of high dimensional and limited data, when the standard Euclidean distance is used as in our derivations, $\mathbf{x}_{V_{\nu_s}}$ might include corrupted attributes even though it is detected as normal, which clearly adversely affects the calculation of the neighborhood $\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})$ in (5.7). In addition, $\mathbf{x}_{V_\nu}$ might only include a small support of corruption, and then we would not like to impute $\mathbf{x}_{V_\nu}$ completely. To overcome these two issues, we propose to use the ranked Euclidean distance defined in (5.4). To this end, the neighborhood $\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})$ is defined using h_α with an appropriate $\alpha \neq 1$ in (5.7). This cancels the adverse effect, up to a certain degree, of a possible corruption in $\mathbf{x}_{V_{\nu_s}}$ as desired. Nevertheless, recalling that h_α only uses the α fraction of the attributes V_{ν_s} and set the others free, h_α is not a metric in the mathematical sense and then, as $N_s \to \infty$, $\mathcal{L}(\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})) \to 0$ does not hold. As a result, the correlation structure given in (5.5) is less exploited in imputation as α decreases. Meanwhile, as α decreases, the support of the detected corruption in x_{V_ν} increases, i.e., localization improves. Therefore, we obviously have a trade-off between the imputation quality and the localization, which is sensitive to the choice of α and investigated in the experiments in greater detail. However, α should be set typically around $0.5 - 0.75$ since we use half-way splits. Finally, note that the imputation brings almost no further computational complexity, since these steps do computationally only depend on the anomaly detection results, cf. (5.2) and (5.1), at the corrupted node,

its sibling node as well as its parent node, which are all generated prior to the imputation steps.

In the following section, the proposed framework is shown to achieve a constant false alarm rate in terms of the corruption detection. Moreover, this false alarm rate is precisely calculated under a certain dependency structure among the anomalous/normal labels on the partitioning tree.

5.2.4 False Alarm Rate in Detecting Corruptions

Since the imputation is an "overwriting" operation, whether or not to impute a suspicious instance is certainly a "critical" decision. In case of a false decision if the suspicious instance is in fact uncorrupted, i.e., "a false alarm in detecting corruptions", the imputation would correspond to data loss. In this section, we study the rate of such occurrences and analyze the false alarm rate of the proposed algorithms in detecting corruptions.

The anomaly detection test applied at every node in Algorithm TCS (Tree-based Corruption Separation) operates with a constant false alarm rate τ, whereas the proposed approach is able to reject corruptions at anomalous nodes. For example, when the terminating pattern in Fig. 5.2b is encountered, all the anomalies that can be present in the tree rooted from the terminating pattern are rejected, i.e., they are not counted as corruptions. For this reason, the false alarm rate of the proposed approach must be defined in the sense of corruptions as opposed to anomalies. To analyze this false alarm rate in detecting corruptions, one also must account for the fact that the anomaly detection test at a node could be strongly correlated with the outputs of the previous tests in the course of Algorithm TCS, since the data attributes are in general correlated. In this section, we first model the labeling of the nodes, i.e., anomalous vs normal, on the partitioning tree, cf. Fig. 5.1, as a directed acyclic graph [101] achieving a certain dependency structure and then derive the false alarm rate of Algorithm TCS. Under this modeling, we also show that the constant false alarm rate in detecting the local anomalies at each node also globally maps to a constant false alarm rate

in detecting the corruptions.

Recall that Algorithm TCS expands the binary tree in Fig. 5.1 for a given uncorrupted test instance $\mathbf{s}$ and declares a corruption only if the conclusive pattern in Fig. 5.2a is encountered or a leaf node is found anomalous in the described breadth-first search. In addition to the corruption localization as well as the imputation capabilities of the proposed Algorithm TCS, let us denote the corruption detection in Algorithm TCS by $\mathcal{C}(\mathbf{s}) = 1$, if $\mathbf{s}$ is detected to be corrupted and $\mathcal{C}(\mathbf{s}) = 0$, otherwise. Then our task is to find the false alarm probability in detecting the corruptions, which is given by

$$C_\tau = \int_{\forall \mathbf{s}} \mathcal{C}(\mathbf{s}) f_0(\mathbf{s}) \mathbf{ds}, \tag{5.10}$$

where τ is the constant false alarm rate of the detection at each node and f_0 is the nominal density. Next, we observe that Algorithm TCS maps every data instance to a binary observation such that the nominal distribution f_0 is transformed into a multivariate Bernoulli distribution p_0, i.e.,

$$\begin{aligned}
&\mathbb{R}^d \to B^{2^{L+1}-1} \text{ via} \\
&\mathbf{s} \to \mathcal{L}(\mathbf{s}) = \mathbf{u} = (u_R, u_{R_l}, u_{R_r}, u_{R_{ll}}, u_{R_{lr}}, u_{R_{rl}}, u_{R_{rr}}, ...),
\end{aligned}$$

where $B = \{-1, 1\}$, L is the depth and u_R is the anomaly decision at the root node such that $u_R = 1$, if an anomaly detected; and $u_R = -1$, otherwise. Similarly for the others such as u_{R_l} is the decision at the left hand child of the root and u_{R_r} is the decision at the right hand child. Note that the proposed algorithm does not completely construct the binary tree but expands, i.e., the nodes and the edges are created as needed. Therefore, we do not completely observe the binary vector $\mathbf{u}$ that an instance $\mathbf{s}$ maps to, however, we temporarily suppose that all the labels are available for ease of exposition. Once $\mathbf{s}$ is mapped to $\mathbf{u}$, since Algorithm TCS declares a corruption based on only the vector of binary labels $\mathbf{u}$, we equivalently have

$$\begin{aligned}
C_\tau &= P\left(\mathcal{C}(\mathbf{s}) = 1 \mid \mathbf{s}, \text{ in fact, is uncorrupted}\right) \\
&= \sum_{\mathbf{u} \in \{-1,1\}^{2^{L+1}-1}} \mathcal{C}(\mathbf{u}) p_0(\mathbf{u}) \\
&= 1 - \sum_{\mathbf{u} \in \{-1,1\}^{2^{L+1}-1}} \mathcal{C}^c(\mathbf{u}) p_0(\mathbf{u}),
\end{aligned} \tag{5.11}$$

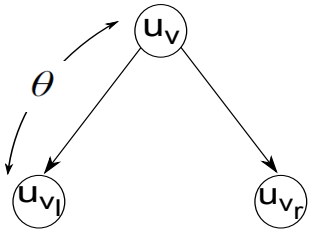


Figure 5.3: We assume the conditional independency: $p_0(u_\nu, u_{\nu_l}, u_{\nu_r}) = p_0(u_{\nu_l}|u_\nu)p_0(u_{\nu_r}|u_\nu)p_0(u_\nu)$. Moreover, $p_0(u_{\nu_l}|u_\nu) = (1-\theta)p_0(u_{\nu_l}) + \theta 1_{\{u_{\nu_l}=u_\nu\}}$, where θ defines the dependency between the parent node and its siblings such that a positive covariance is embedded. Note that $\theta = 0$ implies independency.

where $\mathcal{C}(\mathbf{u})$ is the corruption decision (with abuse of notation), $\mathcal{C}^c(\mathbf{u})$ is the complement, i.e., $\mathcal{C}^c(\mathbf{u}) = 1 - \mathcal{C}(\mathbf{u})$ and p_0 is the corresponding nominal probability mass function such that

$$p_0(\mathbf{u}) = \int_{\forall \mathbf{s}:\mathcal{L}(\mathbf{s})=\mathbf{u}} f_0(\mathbf{s})\mathbf{ds}.$$

In order to calculate the probability mass function p_0, we model the binary tree, where each node corresponds to a binary random variable, as a directed acyclic graph [101] such that the binary random variables at any two sibling nodes are independent conditioned on the knowledge of the label at the parent node. Namely, for any non leaf node ν and its children ν_l and ν_r on the binary partitioning tree, we assume the following conditional independency for the associated random labels: $p_0(u_{\nu_l}, u_{\nu_r}|u_\nu) = p_0(u_{\nu_l}|u_\nu)p_0(u_{\nu_r}|u_\nu)$, from which we obtain

$$\begin{aligned} p_0(u_\nu, u_{\nu_l}, u_{\nu_r}) &= p_0(u_{\nu_l}, u_{\nu_r}|u_\nu)p_0(u_\nu) \\ &= p_0(u_{\nu_l}|u_\nu)p_0(u_{\nu_r}|u_\nu)p_0(u_\nu), \end{aligned} \tag{5.12}$$

cf. Fig. 5.3. Here, we emphasize that $\mathbf{s}$ (or $\mathbf{u}$) is assumed to be uncorrupted in the false alarm analysis to calculate the probability given in (5.10), i.e., it does not have any localized corruptions by definition. Then, without loss of generality, if $\mathbf{s}$ is declared as anomalous at the root node, then this anomaly is not due to a corruption but simply a "rarity" as the test in (5.2) is based on density levels. On the contrary to the case of corruption, since a "rarity" at a node is not a localized phenomenon, we expect that the children inherit the parent label independently. Therefore, we assumed the conditional independency in (5.12) as a generating dependency structure for the simplest graph presented in Fig. 5.3,

which straightforwardly generalizes to the binary tree of the anomalous vs normal labels from root to the leaves. Based on this, we obtain

$$\begin{aligned} p_0(\mathbf{u}) &= p_0(\mathbf{u}_R|u_R)^* p_0(u_R) \\ &= p_0(\mathbf{u}_{R_l}\mathbf{u}_{R_r}|u_R, u_{R_l}, u_{R_r}) p_0(u_{R_l}, u_{R_r}|u_R) p_0(u_R) \\ &= p_0(\mathbf{u}_{R_l}|u_{R_l})^* p_0(\mathbf{u}_{R_r}|u_{R_r})^* p_0(u_{R_l}|u_R) p_0(u_{R_r}|u_R) p_0(u_R), \end{aligned} \tag{5.13}$$

where $\mathbf{u}_R$ is the collection of the binary variables associated with the nodes in the tree rooted from node R that excludes u_R, and the last equation follows from (5.12) and the Bayes' rule. We observe that the starred factors in the expression (5.13) are of similar forms such that the last equation can be expanded further using similar lines of derivations up until the leaves appear.

Thus, the calculation of $p_0(\mathbf{u})$ requires the calculation of the probabilities of the form $p_0(u_{\nu_l}|u_\nu)$ or $p_0(u_{\nu_r}|u_\nu)$, e.g., $p_0(u_{R_r}|u_R)$ in (5.13). Let us denote any child of the node ν by ν_s for generalization. Note that if u_ν and u_{ν_s} were independent then we would have $p_0(u_{\nu_s}|u_\nu) = p_0(u_{\nu_s}) = \tau$ when $u_{\nu_s} = 1$. However, we anticipate a statistical dependency between u_ν and u_{ν_s} generating a positive covariance. That is, conditioned on the knowledge of u_ν, we would like to impose that u_{ν_s} is more likely to attain the value u_ν compared to the prior conditions, i.e., ν_s is likely to inherit the label of its parent. On the other hand, provided that u_ν and u_{ν_s} are identically dependent, we would have that $p_0(u_{\nu_s}|u_\nu) = 1_{\{u_\nu = u_{\nu_s}\}}$, where $1_{\{.\}}$ is the indicator function. To introduce this into the derivations, we parameterize the probability mass function $p_0(u_{\nu_s}|u_\nu)$ as the weighted average between $p_0(u_{\nu_s})$ and $1_{\{u_\nu = u_{\nu_s}\}}$ as

$$\begin{aligned} p_0(u_{\nu_s}|u_\nu) &= (1-\theta) p_0(u_{\nu_s}) + \theta 1_{\{u_\nu = u_{\nu_s}\}} \\ &= (1-\theta)\left(0.5 - u_{\nu_s}(0.5 - \tau)\right) + \theta \frac{1 + u_\nu u_{\nu_s}}{2}, \end{aligned} \tag{5.14}$$

where $\theta \in [0, 1]$ is a parameter defining the degree of dependency which generates an increasing covariance as θ increases in the interval $[0, 1]$ such that $\theta = 0$ implies the statistical independency of u_ν and u_{ν_s}; and $\theta = 1$ implies identical dependency. Then, the probability mass function $p_0(\mathbf{u})$ can be calculated using this parametrization based on the recursion in (5.13). Hence, exhaustively enumerating all possible $\mathbf{u}$'s and running Algorithm TCS for each of them, one

can calculate the false alarm rate C_τ in (5.11), which is not a practical choice. Instead, through the conditional factorization in (5.12), we opt to simplify the expression (5.11) and obtain an efficient recursion. To this end, for a given node ν with depth $1 \leq i \leq L-2$, let us define the probability conditioned on u_ν that Algorithm TCS does not declare a corruption in the tree rooted from ν denoted by $\mathcal{F}(\nu; u_\nu)$ as

$$\mathcal{F}(\nu; u_\nu) = \sum_{\mathbf{u}_\nu \in B^{2^{L-i+1}-2}} \mathcal{C}^c((\mathbf{u}_\nu, u_\nu)) p_0(\mathbf{u}_\nu | u_\nu).$$

Here, $\mathcal{F}(\nu; u_\nu)$ solely depends on the depth variable i due to the symmetric factorization by the conditional independency from parents to children. Therefore, the notation simplifies to $\mathcal{F}(i;1)$ or $\mathcal{F}(i;-1)$. Using the 4 possible configurations for $(u_\nu = 1, u_{\nu_l}, u_{\nu_r})$, we can calculate $\mathcal{F}(i;1)$ as a function of $\mathcal{F}(i+1;.)$. Noting that two of those configurations are the conclusive patterns, termination and corruption patterns, we obtain

$$\mathcal{F}(i;1) = q_1^2(-1) + 2q_1(-1)q_1(1)\mathcal{F}(i+1;1)\mathcal{F}(i+1;-1),$$

where $q_i(j) = p_0(v_s = j | v = i)$ as a short hand notation; the second term corresponds to the continuation of Algorithm TCS and the first term corresponds to the terminating pattern. Unlike the second term, the first term does not have a multiplier since the search stops at such a node. Note that the corruption pattern is disregarded by definition. Similarly, we also have

$$\begin{aligned}\mathcal{F}(i;-1) = q_{-1}^2(1)\mathcal{F}^2(i+1;1) + q_{-1}^2(-1)\mathcal{F}^2(i+1;-1) \\ + 2q_{-1}(1)q_{-1}(-1)\mathcal{F}(i+1;1)\mathcal{F}(i+1;-1).\end{aligned}$$

Recalling that we declare corruptions at leaf nodes on the basis of local anomalies, we can further define

$$\mathcal{F}(L-1;1) = q_1^2(-1) \text{ and } \mathcal{F}(L-1;-1) = q_{-1}^2(-1),$$

and provide the initialization to the recursion $\mathcal{F}(i;1)$ and $\mathcal{F}(i;-1)$. On the other hand, we never declare corruptions at the root since we are focused only on localized corruptions, which is an exception and can be straightforwardly incorporated in our recursions. In terms of the recursions regarding $\mathcal{F}(i;1)$, the only change is

that the corruption pattern should not be disregarded which does not lead to a corruption detection and so does not stop the search. Then, we simply have

$$\mathcal{F}(0;1) = q_1^2(-1) + q_1^2(1)\mathcal{F}^2(1;1) + \\ 2q_1(-1)q_1(1)\mathcal{F}(i+1;1)\mathcal{F}(i+1;-1);$$

and the recursion $\mathcal{F}(i;-1)$ stays valid for $\mathcal{F}(0;-1)$. Now that we have the recursion equations defined for all depth levels on the binary tree, we can efficiently calculate the false alarm rate of Algorithm TCS as follows. Letting R represent the root node, we obtain from (5.11)

$$\begin{aligned} 1 - C_\tau &= \sum_{\mathbf{u} \in B^{2^{L+1}-1}} \mathcal{C}^c(\mathbf{u}) p_0(\mathbf{u}) \\ &= p_0(u_R = 1) \sum_{\mathbf{u}_R \in B^{2^{L+1}-2}} \mathcal{C}^c((u_R, \mathbf{u}_R)) p_0(\mathbf{u}_R | u_R = 1) + \\ &\quad p_0(u_R = -1) \sum_{\mathbf{u}_R \in B^{2^{L+1}-2}} \mathcal{C}^c((u_R, \mathbf{u}_R)) p_0(\mathbf{u}_R | u_R = -1). \end{aligned}$$

Then, recalling that $p_0(u_R = 1) = 1 - p_0(u_R = -1) = \tau$, the false alarm rate C_τ is given by

$$C_\tau = 1 - \tau\mathcal{F}(0;1) - (1-\tau)\mathcal{F}(0;-1), \tag{5.15}$$

which is equivalent to first calculating the probability that Algorithm TCS never declares a corruption and then subtracting this probability from 1.

Since the false alarm rate C_τ of Algorithm TCS (Tree-based Corruption Separation) in detecting the corruptions as found in (5.15) is independent from the data, we conclude that the false alarm rate τ of the anomaly detection at each node maps to a constant false alarm probability of our corruption detection C_τ. Secondly, even though the dependency parameter θ does not appear, i.e., hidden, in the expression (5.15), C_τ is clearly affected by θ. For example, if $\theta = 1$, i.e., if the binary label of a child node is identically dependent on the parent label and hence $p_0(u_{\nu_s}|u_\nu) = 1_{\{u_{\nu_s} = u_\nu\}}$, then it can be shown that $C_\tau = \tau$. If $\theta = 0$, i.e., if the binary label of a child node is independent with the parent label and hence $p_0(u_{\nu_s}|u_\nu) = p_0(u_{\nu_s})$, then obviously $C_\tau > \tau$. We experimentally discuss

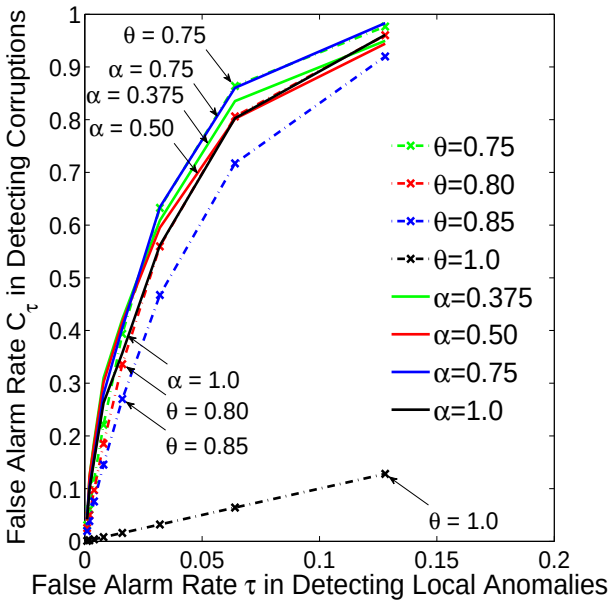


Figure 5.4: Solid (dash-dot) curves correspond to the realizations (hypothetical results). The constant false alarm rate τ in detecting the local anomalies maps to a global constant false alarm rate C_τ in detecting the corruptions with Algorithm TCS (Tree-based Corruption Separation). We observe that setting $\theta \in [0.75, 0.8]$ well approximates the relation between τ and C_τ. In case of the identical dependency, i.e., $\theta = 1$, $C_\tau = \tau$.

the quality of this hypothetical relation between τ and C_τ, cf. Fig. 5.4, and the further details in Section 5.4.

In the following section, we explain the important points of our implementation and discuss the corresponding computational complexity.

5.3 Computational Complexity

Computationally, the main building block in Algorithm TCS (Tree-based Corruption Separation) is the application of the anomaly test defined in (5.2), which computes the train-to-train distance matrix $D_S(i,j) = d(\mathbf{s}_i, \mathbf{s}_j)$ and the test-to-train distance vector $D_X(j) = d(\mathbf{x}, \mathbf{s}_j)$. Operating on these distances, the score function defined in (5.1) for the test instance must be computed, which then

requires the computation and sorting of $R_S(\mathbf{s}_i; K)$. In addition, since we label each node as anomalous or not in our tree expansion, these distances must actually be computed at each node with respect to the corresponding attributes, e.g., $D_{S_{V_\nu}}(i,j)$ and $D_{X_{V_\nu}}(j)$ at a node ν. For this purpose, we adapt the "integral image" approach for the cases, where the standard Euclidean distance is used. Namely, let us define the volume $\mathcal{D}_S(i,j,k) = \sum_{h=1}^{k}(s_{ih} - s_{jh})^2$, $\forall i,j$ with $1 \leq k \leq d$; and $\mathcal{D}_S(i,j,0) = 0$, $\forall i,j$ (similarly for $\mathcal{D}_X(j)$). Then, we simply have $D_{S_{V_\nu}}(i,j) = \sqrt{\mathcal{D}_S(i,j,k_2) - \mathcal{D}_S(i,j,k_1)}$ at a node ν, where V_ν corresponds to the set of attributes in positions between k_1+1 and k_2. The volume $\mathcal{D}_S(i,j,k)$ and the sorting of $R_S(\mathbf{s}_i; K)$ can be computed offline once the training set is provided, which defines a training phase complexity $O(2^{L+1}N_s^2 \log_2 N_s)$, where sorting is the dominant contributor. For a given test instance, we compute $D_{X_{V_\nu}}(j)$ and sort at each node ν in the expansion of our tree, which defines the test phase complexity $O(2^{L+1}N_s \log_2 N_s)$ for our algorithm, where sorting is the dominant contributor. The computational load is multiplied by constant factors in case of ranked Euclidean distance.

5.4 Experiments

In this section, we first discuss the efficacy of the false alarm rate estimation method explained in Section 5.2.4 and evaluate the performance of the critical steps in Algorithm TCS (Tree-based Corruption Separation), which are the corruption detection, localization and imputation. Then, we report the improvements achieved by the proposed framework in several classification tasks in comparison to a baseline of two state-of-the-art algorithms.

In the first set of experiments, we adapted a $0-1$ digit classification task consisting of a training set of 1500 samples and a test set of 750 samples based on the USPS data [123]. Each of these samples is a 16×16 gray scale image of either a "0" image or a "1" image, where each pixel has a real intensity value in [0,1]. We synthetically generate a corruption as described in Section 5.1 and apply to each instance in the test set with probability $\pi = \frac{1}{2}$. To be more precise,

for a test instance chosen to be corrupted, we (uniformly) randomly specify a square region of size between $(10-50)\%$ of the total area, i.e., the number of pixels in the chosen region is not less than 25 and not more than 128, overwrite each pixel in this region with a value randomly (using the uniform distribution $U_Z(z)$) drawn from the interval $[0,1]$. Then, after the training and test instances are vectorized column wise such that $\mathbf{s},\mathbf{x}\in\mathbb{R}^{256}$, the proposed Algorithm TCS is provided with the clean training data and run over the test set. We emphasize that by this vectorization scheme, the corrupted square region corresponds to multiple corrupted intervals in the vectorized observation. Hence, this example also illustrates that Algorithm TCS can handle multiple corruptions. Ideally, the neighborhood size parameter K for both imputation and corruption separation purposes should be optimized at every node of our binary tree since the data dimensionality from node to node varies. However, we opt not to optimize K for presentational clarity and set as $K=8$ near the midpoint of $[1,16]$, which is empirically found appropriate. Using the $0-1$ digit USPS data, we investigate the response of the Algorithm TCS to the local anomaly detection false alarm rate $\tau\in\Gamma=\{0.001,0.002,0.004,0.008,0.016,0.032,0.064,0.128\}$ and the ranked Euclidean distance parameter $\alpha\in\Delta=\{0.375,0.5,0.75,1\}$. As for the depth parameter, we use the deepest possible tree with $L=6$ such that the leaves are associated with 4 pixels and hence, 1 pixel at least is then used in the distance calculation with $\alpha=0.375$.

In Fig. 5.4, we compare the hypothetical false alarm rate C_τ we derive in Section 5.2.4 with the corresponding experimental realizations with respect to varying local anomaly detection false alarm $\tau\in\Gamma$. The hypothetical map from τ to C_τ is generated with several choices for the dependency parameter, $\theta\in\{0.75,0.8,0.85,1\}$, whereas the realizations correspond to several choices for the distance parameter, $\alpha\in\Delta$. Our experiments indicate that when the statistical dependency θ in (5.14) from a parent node to one of its children nodes, cf. Fig. 5.3, is chosen around 0.75, the relationship between the local anomaly false alarm rate τ and the corruption detection false alarm rate C_τ is accurately modeled. This experimentally shows that the labeling of local anomalies over a binary partitioning tree shown in Fig. 5.1 can be considered as a directed acyclic graph.

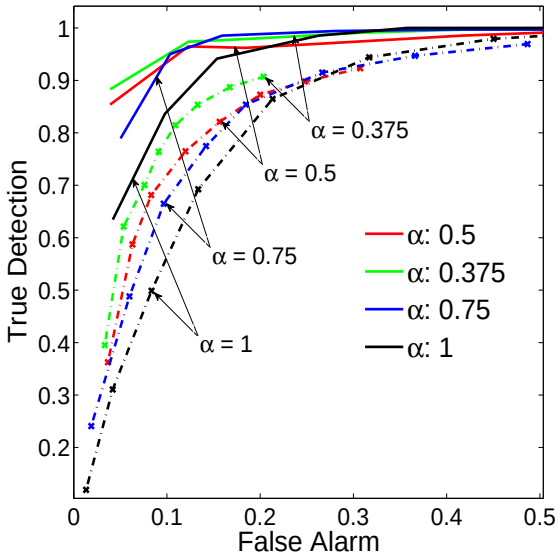


Figure 5.5: ROC curves for detection and localization of corruptions. Solid (dash-dot) curves correspond to detection (localization) performances.

We also observe that in the case of Euclidean distance, i.e., h_α with $\alpha = 1$, while $\theta \sim 0.75$ is more accurate for small τ's, θ tends to approach 0.8 as τ increases for a better modeling. This small deviation mainly happens since the conditional independency assumption explained in Fig. 5.3 does not hold in case of Euclidean distance for a certain pattern. Namely, although the labeling for a parent and its children nodes as $-1, 1, 1$ (a normal parent node with anomalous children nodes) is not possible with the standard Euclidean distance due to the test defined in (5.2), the directed acyclic graph modeling assigns it a positive probability, which then overestimates the corruption detection false alarm rate. Nevertheless, this positive probability is the smallest among the ones assigned to the all possible patterns of three nodes as desired and hence, the ordering of the patterns in terms of their probabilities is still reasonable even in the case of the Euclidean distance. On the contrary, since this pattern is also possible in the case of the ranked Euclidean distance, the accuracy of our hypothetical results improves as α decreases.

Next, we study the corruption detection and localization performance of our algorithm on the $0-1$ digit USPS data. In Fig. 5.5, we plot the empirical false

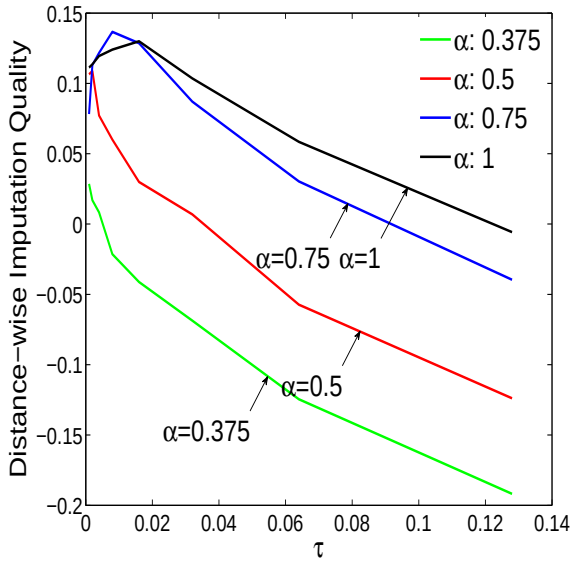


Figure 5.6: Distance-wise imputation quality with $\tau \in \Gamma$.

alarm rates versus the empirical true detection rates in terms of both corruption detection and corruption localization with respect to $\tau \in \Gamma$. Here, the true detection rate is the empirical probability, i.e., relative frequency, of a truly corrupted data instance (data attribute in case of localization) to be declared corrupted and, the false alarm rate is the empirical probability of a truly uncorrupted data instance (data attribute in case of localization) to be declared corrupted. As we discuss in Section 5.2, the ranked Euclidean distance is experimentally shown to produce a better detection as well as localization performance on the USPS data as α decreases. Recall that for a small α around 0.5, we enforce a corruption to be widely spread for Algorithm TCS to detect it at a node, which then clearly improves the localization. Similarly, the corruption detection performance also improves as α decreases. Since the ranked Euclidean distance disregards a certain fraction of largest attribute-wise deviations, Algorithm TCS behaves conservatively in declaring corruptions. This reduces the false alarms in terms of the local anomalies and in turn, reduces the false encounters of the terminating pattern shown in Fig. 5.2b. Hence, the corruption search is not stopped mistakenly, and Algorithm TCS does not miss certain corruptions, which leads to a better detection rate with the ranked Euclidean distance using a small α around 0.5. We emphasize that the local anomaly detection false alarm rate τ can be set

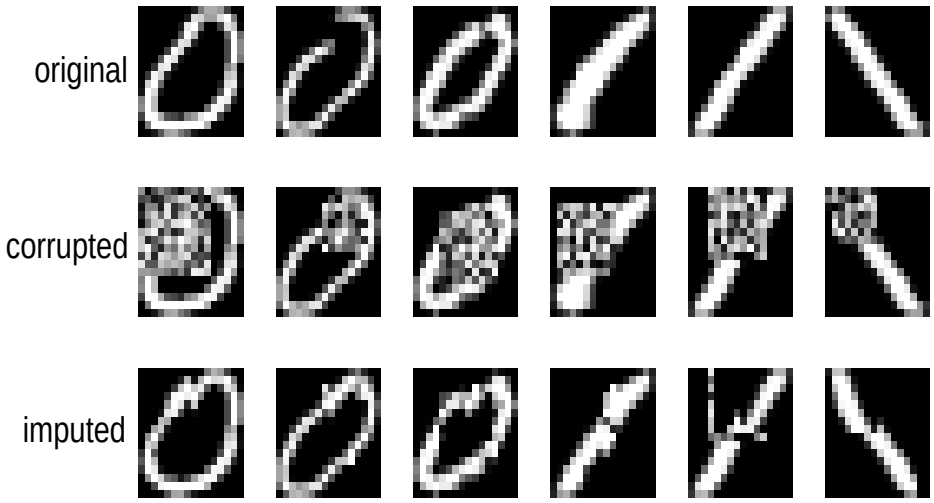


Figure 5.7: Several visual examples on USPS dataset.

independently for detection and localization to precisely determine the operating point on the ROC curves in Fig. 5.5. However, in this study, we use one single τ in all phases of Algorithm TCS. Note that when the false alarm rate is set around $0.1-0.2$, our algorithm is able to provide a detection rate around 0.9 and a localization rate around 0.8.

On the other hand, the ranked Euclidean distance parameter α cannot be made arbitrarily small. Observe that with small α, only a small fraction of attributes are used in determination of $\mathcal{N}_{N_s}(\mathbf{x}_{V_{\nu_s}})$ in (5.7) despite that the rest of the attributes might be informative through the local correlations and hence, the imputation quality degrades. We illustrate this effect in Fig. 5.6, where we use the improvements in the distance wise deviations after imputation to measure the imputation quality. For this purpose, we define

$$\frac{1}{N_c}\sum_{i=1}^{N_c}\frac{h_{\alpha=1}(\bar{\mathbf{x}}_i,\mathbf{x}_i)-h_{\alpha=1}(\bar{\mathbf{x}}_i,\hat{\mathbf{x}}_i)}{h_{\alpha=1}(\bar{\mathbf{x}}_i,\mathbf{x}_i)} \tag{5.16}$$

as the distance wise imputation quality, where N_c is the number of the corrupted test instance (which is approximately 750π), $\mathbf{x}_i$ is a corrupted test instance, $\bar{\mathbf{x}}_i$ is the uncorrupted original instance and $\hat{\mathbf{x}}_i$ is the corresponding instance after imputation. Note that this quality metric measures on average how much of the

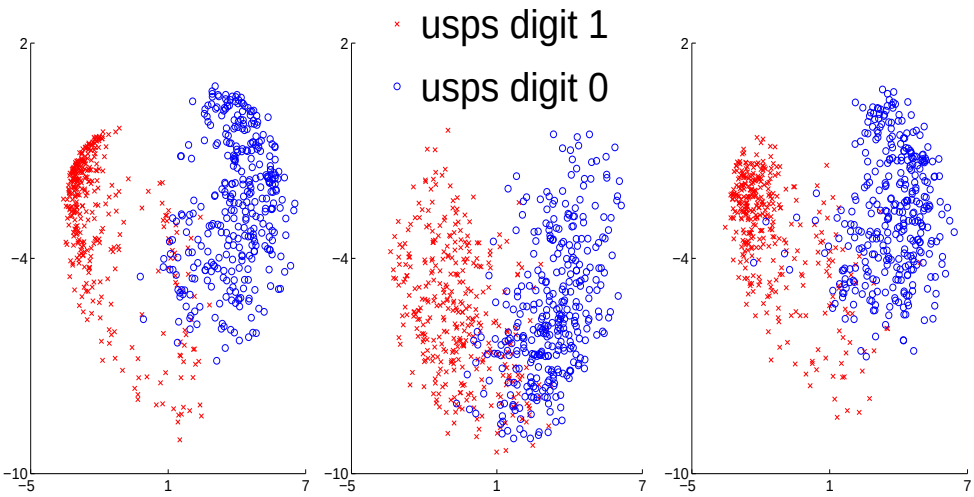


Figure 5.8: On the left is the (uncorrupted) true data scatter; mean separation between two classes: 5.95, linear SVM accuracy: 99.71%. In the middle is the corrupted data scatter; mean separation: 4.19, classification accuracy: 90.57%. On the right is the imputed data scatter; mean separation: 5.37, classification accuracy: 96.85%, which corresponds to $\sim 68.0\%$ improvement both in terms of the mean separation and classification.

distortion by the corruption is recovered after imputation. The average imputation quality defined in (5.16) is plotted versus the local anomaly detection false alarm $\tau \in \Gamma$ in Fig. 5.6. We first observe that for large τ, since the false alarm rate is also large, the imputation even further disturbs the data. Secondly, for small τ around 0.01, the proposed imputation technique is able to correct a corrupted instance up to 12% in case of $\alpha \sim 0.75$. Moreover, our experiments also indicate that for α less than 0.5, the ranked Euclidean distance is not able to produce desirable results despite its superiority in terms of detection and localization, which reinforces our discussion that α cannot be made too small.

Unlike an MMSE based approach, our MAP based imputation does not target at minimizing the reconstruction error but the most likely replacement for a corruption. Indeed, an MMSE based estimator for imputation would produce visually blurry results, for instance on the USPS data. In this regard, we present several visual examples that the proposed framework generates on the USPS data with $\tau = 0.016$, $\alpha = 0.75$, $K = 8$ in Fig. 5.7. Note that the presented visual examples tend to generate image gradients that are naturally aligned with the

Data Sets (dimensionality) Avg. acc(**X**) / acc(**X**)	Uncorrupted Training Avg. % Improvements with its Std. below TCS-MAP / TCS-NN / 4-NN (16-NN)	%5 Corrupted Training Avg. % Improvements with its Std. below TCS-MAP / TCS-NN / 4-NN (16-NN)
Image (18)	55.60 / **59.14** / 42.80 (10.34)	**40.96** / 40.95 / 27.84 (9.34)
84.62 / 57.16	1.81 / 2.58 / 3.20 (1.47)	1.32 / 1.92 / 3.32 (1.79)
Ringnorm (21)	**46.85** / 46.15 / 31.99 (19.42)	**43.36** / 39.60 / 26.36 (18.35)
76.90 / 64.40	4.23 / 3.27 / 5.40 (4.67)	4.25 / 4.89 / 4.21 (5.62)
Twonorm (21)	31.16 / **31.36** / 14.75 (16.46)	**22.60** / 20.97 / 11.64 (12.51)
96.46 / 85.84	3.51 / 1.91 / 2.09 (2.71)	3.49 / 1.96 / 1.70 (3.55)
Svmguide3 (21)	62.33 / **68.86** / 60.57 (6.86)	56.11 / **59.80** / 43.03 (12.04)
81.08 / 68.74	3.03 / 3.29 / 2.09 (5.38)	2.79 / 3.71 / 3.62 (4.15)
Breast Cancer (30)	72.19 / **72.24** / 19.07 (46.08)	**57.74** / 55.07 / 18.80 (42.33)
97.30 / 85.66	3.65 / 3.47 / 2.99 (6.07)	2.93 / 3.11 / 5.34 (5.68)
Spam Data (57)	**77.38** / 77.38 / 53.86 (64.85)	**69.46** / 68.18 / 33.78 (35.43)
88.66 / 62.56	1.69 / 1.71 / 1.43 (1.58)	2.01 / 1.88 / 2.07 (2.31)
Sonar (60)	40.25 / **48.00** / -25.83 (-38.20)	40.33 / **44.96** / -47.86 (-27.45)
75.51 / 70.43	28.42 / 26.68 / 16.87 (36.62)	24.64 / 25.79 / 25.39 (36.74)
BCI (117)	27.75 / **34.12** / 31.10 (8.19)	15.12 / **15.47** / 11.76 (8.79)
79.55 / 60.08	7.75 / 6.60 / 5.79 (9.33)	5.30 / 6.26 / 3.20 (5.16)
Digit1 (241)	76.03 / **81.96** / 39.29 (40.06)	25.29 / **27.36** / 19.49 (17.70)
95.96 / 87.80	4.61 / 3.30 / 2.44 (4.16)	3.65 / 2.92 / 3.57 (3.28)
G241c (241)	**13.04** / 12.72 / -13.41 (32.45)	**9.54** / 9.46 / 3.86 (10.95)
87.42 / 76.00	7.65 / 8.13 / 5.79 (5.77)	3.34 / 2.79 / 1.61 (3.14)
G241n (241)	**18.34** / 16.55 / -18.93 (34.64)	**4.93** / 2.85 / 1.47 (6.89)
85.60 / 77.54	7.07 / 7.54 / 8.16 (5.40)	3.25 / 3.59 / 3.16 (4.49)

Figure 5.9: The proposed algorithm TCS-MAP is compared with two baseline algorithms TCS-NN and M-NN in terms of the classification tasks on several benchmark data sets. Average improvements in classification accuracies after imputation are presented for all methods in the cases of clean data training and %5 corrupted training.

image statistics. We also observe some cases, where the corruption along a border between the cells of our partitioning tree remains after the imputation, cf. the second last column in Fig. 5.7. The residual corruptions in such cases can be handled by increasing the depth of the tree or using m-ary trees as opposed to binary splits, which is not in the scope of this study. In addition to the visual comparisons, we also evaluate the performance of the introduced framework in terms of the classification purposes. On the described $0-1$ digit USPS data, we report the data scatter plots of the test instances in Fig. 5.8, where we project the original, corrupted and imputed test data onto the two eigen vectors of the training set with the largest eigenvalues for visualization. We clearly observe a better class separation between two classes after the imputation, when compared to the class separation in the corrupted data.

We emphasize that since the proposed tree based corruption separation framework is a comprehensive one such that it operationally covers the partial solutions in the corresponding literature, it is not possible to provide a perfectly fair comparative analysis. Nevertheless, we compare the proposed framework with a baseline of algorithms constructed using the methods [122, 129] in terms of the classification tasks over the several well-known machine learning data sets [123, 124]. One of these methods, "TCS-NN", consists of the same tree-based corruption separation (TCS) procedure that we propose but utilizes -instead of our MAP imputation- the nearest neighbor (NN) imputation technique [129], which finds the nearest neighbor of a corrupted instance with respect to the sibling attributes of the corrupted node and imputes. The other method, "M-NN", also utilizes the nearest neighbor imputation but does not have a fine/detailed corruption separation step. Instead, it splits an instance into M different segments [122], applies anomaly detection to each segment and imputes an anomalous segment by its nearest neighbor that is found with respect to the neighboring segment.

In these experiments, we use a depth-4 tree for our Algorithm TCS leading to 16 leaves/segments in the finest level with $K = 8$. For each data set, after scaling each data attribute into the interval [0,1], we randomly choose 11 splits of the scaled data such that 2/3 of each split is reserved as the training (reference) data set (at most 1000 instances), and the rest is reserved as the test data set (at most 500 instances) in each split. Moreover, every instance in the test set of each split is randomly corrupted/overwritten from the uniform distribution with the support $[0, 1]$ in a random interval of attributes, which includes at least 10% of the attributes (dimensionality) and at most 50% of them. Since $30 = \frac{50+10}{2}\%$ of each test instance is corrupted on average, choosing $M = 4$ is appropriate for the method "M-NN". We also present results for the case $M = 16$. The first split is used for parameter selection purposes[1], e.g., C parameter of a linear

[1]The same values for the parameters α, τ is used for both "TCS-MAP", and "TCS-NN" to fairly and clearly observe the effect of using NN imputation instead of MAP imputation since using the same rate τ for both methods leads to the same CFAR in corruption detection. In another separate experiment cf. Table 5.1, we directly and explicitly compare the two imputation methods with the standard Euclidean distance only. The same τ is also used for "M-NN", which definitely favors "M-NN" since it corresponds to a lower CFAR for "M-NN". Euclidean distance, $\alpha = 1$, is always used for "M-NN". Depth is always 4 with $K = 8$. C is always common to all methods.

SVM, α, and the remaining 10 splits are used for performance analysis. Then, in each of the remaining 10 splits, we train a linear SVM classifier on the training set and compute the classification accuracy on the uncorrupted, corrupted and the imputed test data. In Fig. 5.9, we report the average of improvements and the corresponding standard deviation of the average improvement (std. of the improvements divided by $\sqrt{10}$ is reported, i.e., std. of the mean value estimator) in terms of the classification accuracy on the imputed test data over 10 splits. The improvement in classification accuracy is calculated as

$$\% \text{ Improvement } = 100 \times \frac{\mathbf{acc}(\hat{\mathbf{X}}) - \mathbf{acc}(\mathbf{X})}{\mathbf{acc}(\bar{\mathbf{X}}) - \mathbf{acc}(\mathbf{X})},$$

where $\mathbf{acc}(\bar{\mathbf{X}})$ ($\mathbf{acc}(\mathbf{X})$, $\mathbf{acc}(\hat{\mathbf{X}})$) is the classification accuracy -after clean data training- on the original uncorrupted test data $\bar{\mathbf{X}}$ (on the corrupted test data $\mathbf{X}$, on the imputed data $\hat{\mathbf{X}}$). Furthermore, in order to evaluate the robustness to corruptions in the training sets and to address the cases where clean data is not available, we randomly choose the 5% of each training data of each split, corrupt with the same corruption model and then repeat[2] the same experiments. The results are summarized in Fig. 5.9.

We observe that the proposed framework is successful at undoing the adverse effects of corruption and always provides (positive) improvements up to $\sim 80\%$ after imputation in terms of the classification performance. Additionally, when the drop in accuracy due to the corruptions is relatively low, e.g., 5 units of drop from 80% to 75%, even a 1 unit gain after imputation corresponds to $20\% = \frac{1}{5}$ improvements, which naturally results in relatively high standard deviations, cf. Sonar data set in Fig. 5.9. Moreover, the proposed framework is shown to be robust to corruptions in the training data by these experiments.

Our algorithms, "TCS-MAP" (Tree-based Corruption Separation with MAP imputation), perform significantly better than the method "M-NN" and comparably with the method "TCS-NN". We point out that the method "TCS-NN"

[2]Exactly the same splits with the same parameters are used. The only difference is that 5% of each training set is corrupted. Note that in these experiments, we first train a linear SVM using clean training data and then test the trained model on uncorrupted (clean) test data, corrupted test data and corrupted test data imputed by the clean training data and corrupted test data imputed by %5 corrupted training data.

strongly relies on the proposed tree-based corruption separation framework, which is the main reason underlying the success of "TCS-NN". (Note that both of the methods "M-NN" and "TCS-NN" uses the NN imputation but "M-NN" does not have the proposed tree-based corruption separation step and the method "M-NN" is outperformed by "TCS-NN".) The method "M-NN" occasionally even further corrupts the data, cf. the negative improvements for Sonar or G241c or G241n. Moreover, it is difficult to choose between the methods "4-NN" and "16-NN" since there is no clear superiority. Namely, there is definitely a scaling issue for the method "M-NN". If the corruption in an instance covers a small portion, e.g., 10%, then choosing M too large would leave several undetectable corruptions in addition to imputing large chunks of clean data due to the false alarms (negative improvements). Similarly, if the corruption in an instance covers a large portion, e.g., 50%, then choosing M too small would not only again harden the anomaly detection (due to the use of insufficient data in detecting corruptions) but also complicate the imputation since on what to condition the imputation becomes ambiguous. For a good imputation in this case, one would need to identify (with significant imperfections due the use of insufficient data because of a small M) all the corrupted and uncorrupted segments; and then condition the imputation on uncorrupted ones, which leads to a non-homogenous different evaluation for every instance that requires computationally a very high load. Therefore, choosing an appropriate M is in general hard since it must depend on the amount of the corruption, which might be unknown and random. The proposed framework resolves this scaling issue in a computationally efficient way via the binary searches and fast imputations. Moreover, our algorithm "TCS-MAP" is experimentally shown to be also robust to corruptions in training data in the sense that it strongly preserves its corruption separation/imputation capabilities even after including 5% corruptions in training.

In our experiments with uncorrupted (clean) data training, the MAP imputation and the NN imputation (only when combined with the proposed tree based corruption separation) performs comparably, which is due to the sparsity of the data compared to the dimensionality. However, we first note that the MAP estimator yields the NN estimator, if the neighborhood size is set $K = 1$ in the

Table 5.1: Performance (MSE and Classification) of the proposed MAP Imputation on the data sampled from a 2 component Gaussian Mixture Model. The MAP Imputation is consistently better than the NN ($K = 1$) Imputation for all K's.

K	1	4	8	12	16
MSE	2.90	2.31	2.19	2.14	2.12
% Classification Accuracy	79.49	80.13	80.45	80.74	80.86

imputation phase, which can easily be achieved via cross validation for K. Clearly, with such a cross validation, the MAP imputation can only perform better than the NN imputation and we opt not to optimize K for presentational clarity. Additionally, the NN imputation is definitely sensitive to corruptions in the training data since the attributes of the nearest neighbor that are used for imputation can also be corrupted with a certain probability, e.g., with probability 0.05 in our experiments. On the other hand, that possibly (with probability 0.05) corrupted nearest neighbor would achieve a lower score value $\hat{p}_K(x)$ if it was truly corrupted and it would not be picked by our MAP estimator for imputation. Thus, the proposed MAP estimator can handle such situations and is robust to corruptions, where the NN imputation performance potentially degrades more. In fact, in our experiments, the MAP imputation either enhances its superiority or becomes superior or approaches NN imputation for most of the data sets after including corruptions in training, e.g., Image or Ringnorm data sets in Fig. 5.9.

To further demonstrate the power of the proposed estimator, we devise a separate experiment, where we use a data set consisting of two Gaussian components with unitary covariances. We use the means $[1, -1]$ and $[-1, 1]$ for positive and negative classes, respectively. We generate 1000 samples as the training data, (500 for each class), and next suppose that the second attribute of each sample is corrupted/missing; and therefore imputed by our MAP estimator with varying neighborhood size K and the NN estimator. Note that in this part, we use standard Euclidean distance only. After repeating this 100 times, the resulting imputed data is compared to the original training data in the MSE sense and in terms of the classification accuracy. We summarize our findings in Table 5.1,

where the MAP imputation with $K = 1$ coincides with the NN imputation. The MAP imputation is consistently better than the NN imputation as expected. For instance, when $K = 12$, we obtain 26% improvements in the MSE sense; and 10% improvements in terms of the classification (original classification accuracy is around 92%).

5.5 Discussion

We introduce a comprehensive framework for handling localized and severe data corruptions. The novel contributions of the proposed framework includes (i) a first algorithm to jointly detect and localize such corruptions by identifying the local anomalies, (ii) a *Maximum A Posteriori* based estimator for imputation; and a distance measure for corruption separation purposes, (iii) computational efficiency via the binary searches and the fast imputations, and (iv) a characterization for anomalous observations, e.g., rarities, incompatible combinations and corruptions. We point out that our algorithm does not assume prior information or a model for the input data and instead, works in a completely data driven way. Furthermore, we present a false alarm rate analysis of the introduced framework and demonstrate that the desired false alarm rate in detecting corruptions can be set independent of the input data without parameter tuning. Through extensive experiments on real life benchmark data sets, we show that our algorithm provides signicant improvements in terms of classication purposes with strong corruption separation. The proposed algorithm significantly outperforms the typical approaches and is robust to varying training phase conditions.

We show that the anomaly detection -in addition to being a stand alone important learning problem- can be successfully incorporated into the online learning framework for corruption handling purposes. Hence, an anomaly detection must be designed efficiently for online learning from sequential data with corruptions. To this end, we also drop the batch processing requirements in Chapter 4 and Chapter 5 and propose an online anomaly detection algorithm for time series data in Chapter 6.

Chapter 6

Online Anomaly Detection under Markov Statistics with Controllable Type-I Error

Detection of anomalous patterns is of great interest in signal processing [161–164] and machine learning [25] since the irregular data due to an anomaly often detrimentally affects the target application and might even require special actions in certain scenarios [25, 165, 166]. For instance, a hacked computer/mobile device produces suspicious network traffic [167]; or an illegal U-turn in an intersection scene creates anomalous video data [163]; or an anomalous pattern in the electricity consumption data of a factory should definitely raise concerns [122]. Similarly, visual occlusions generate unpredictably irregular video data and reduce the object recognition rates [9]. In this chapter, we study the anomaly detection problem for the temporal data in the online setting and propose a novel and computationally highly efficient online algorithm. The proposed algorithm sequentially receives a time series, learns -in an online manner- the nominal attributes in the data and detects the anomalous subsequences. We use the Neyman-Pearson (NP) characterization [11] for the anomalies and nearly achieve a constant controllable false alarm rate with maximum possible detection power regardless of whether the source is stationary or non-stationary. The proposed algorithm is able to

process data at extremely high rates with linear complexity in the size of the streamed data and therefore, it is especially suitable for big data applications. Moreover, we do not require parameter tuning to match the desired rates even if the source statistics change.

There exists an extensive literature on anomaly detection [25,161,163,165,166] and the problem is studied under different nomenclature depending on the anomaly types such as novelty detection [162, 168, 169], outlier detection [164, 170,171], one class learning [118,172], intrusion detection or fault detection [173]. However, as the first time in the literature, we focus on online anomaly detection in stationary or non-stationary fast streaming temporal data with online controllability of the Type-I error that maximizes the detection power without requiring parameter tuning. Thus, the presented study considerably differs from the literature. We consider that the controllability of the false alarm rate is a crucial capability especially in the context of anomaly detection since anomalies in general draws attention and provides actionable information. In this regard, a number of false alarms more than a bearable rate is clearly frustrating and hence potentially risks the practicality of the algorithm [26, 120]. For this reason, we study the problem in the binary hypothesis framework by using the NP formulation [11], where we explicitly bound the false alarm rate (i.e. minimizing Type-I error) while achieving the maximum detection power (i.e. minimizing Type-II error). Although the NP approach is successfully applied to anomaly detection [26, 120, 164], the online implementation of the NP solution has been left unexplored. Furthermore, the existing batch NP solutions are typically based on the assumption of independent and identically distributed (i.i.d.) observations, which hardly holds in the case of temporal observations, where the data is typically highly correlated in time and non-stationary. On the contrary, we model such intrinsic correlations by using Markov models without assuming stationary source statistics, i.e., the source might be strictly non-stationary in our work.

Our method falls in the category of statistical anomaly detection with modeled nominal densities [25,174], where an anomaly is an observation that is (suspected to be) not generated by the assumed nominal stochastic model, i.e., not consistent with the majority of the typical observations. One may argue that assuming a

certain shape (which is also unknown) for the nominal distribution is too restrictive [120] and it is difficult to estimate its attributes when the data is limited. However, in our case, the issue is not scarcity of data but its availability in huge amounts and the resulting strict real time processing requirements. Moreover, the assumed stochastic nominal density model (Markov model) in this chapter is not restrictive since we exploit the generality of the well known Wold decomposition theorem [175] for stationary intervals in a non-stationary processes.

A popular approach in statistical anomaly detection is to consider the distance from a suspicious instance to the nominal set of data. In [176], if the k'th minimum distance is sufficiently large, then an anomaly is declared. This approach has deep theoretical roots in terms of the false alarm controllability. The rankings of these k'th distances are used in [177] to bound the number of false alarms in a computationally relatively more efficient manner (via an approximate nearest neighbor calculations). Such rankings have shown in [120] to be an asymptotically consistent estimator for inclusion in the Minimum Volume (MV) set, which is the optimal decision region (after taking the complement) for anomalies in the NP formulation when the anomalies are assumed to be uniformly distributed, cf. [26]. The method in [120] impressively avoids the explicit calculation of the MV set but rather calculates the sufficient membership indicator via the k'th rankings. Another method is the Geometric Entropy Minimization (GEM) [178], which compares a test instance with only the most concentrated subset of the nominal training data that is asymptotically convergent to the MV set. A computationally tractable algorithm for GEM is also provided in [178], which however does not maintain the original statistical guarantees.

We also use the MV set approach to detect anomalies in temporal data. However, our goal is to obtain a computationally scalable algorithm while maintaining the NP optimality, i.e., Type-I error controllability with maximum detection power. These methods [26, 120, 176–178] are computationally too demanding for real time processing in fast streaming data applications. For instance, the method [120] requires pair-wise distance calculations and sorting to obtain the rankings and essentially to order the likelihoods, which results in quadratic complexity in data size. Thus, one can hardly use such methods in our framework,

although they are impressive batch processing techniques. On the contrary, instead of such distance calculations and orderings, we directly approximate the distribution of the likelihoods under the general Markov models for temporal data and analytically calculate the desired quantile for the MV set. This allows a computationally highly efficient and online implementation of the NP approach with controllable Type-I error and maximum detection rate. Moreover, we do not require (like [120] and unlike others) parameter tuning to match the desired false alarm rate and also, we do not assume (unlike these methods) stationary data source.

Another statistical technique that discriminates a concentrated subset of nominal data as the decision region is the one class SVM [118], where the nominal data is lifted to a high dimensional space such that it is separated from the origin with the maximum margin. Similar approaches via the minimax probability machine in [179] and a linear programming solver (instead of quadratic programming of [118]) in [180] have also been proposed to obtain efficient and robust implementations. Nevertheless, none of these methods directly address our online processing needs as they are basically batch techniques (without update procedures) requiring at least quadratic processing complexity. Also, the false alarm controllability is not explicitly considered in these studies [118, 179, 180].

The goal in this chapter is to detect anomalous subsequences in a time series. This instance of the problem for temporal data (not in the i.i.d. batch setting like [25, 26, 120]) has also been considered, cf. [181–192] and the references therein. However, most studies do not address the problem in the online setting and they assume stationary source statistics [165, 166]. An anomaly score is assigned to each fixed length subsequence using the pair-wise distances among all possible such subsequences in [181] and the detection is based on the magnitude of the anomaly score. Several approaches [183–186] are then proposed to relieve the computational burden of the pair-wise distance calculations such as tree representations and prunings [182], local hashing [183] and Haar transform [185, 190]. Instead of the standard Euclidean distance, a compression based similarity measure is also investigated in [187, 188]. Several other methods exists, which consider unevenly sampled stochastic processes [189] as well as differently

defined anomalies [166, 191, 192]. Despite the impressively efficient implementations (for instance [182, 186, 188, 192]), the model free setting along this line of research requires to investigate pair-wise relations or extract/apply complex features/transformations. Hence, their solutions are essentially not appropriate to process large scale data in real time due to their computationally heavy requirements. In contrast, we exploit the availability of the data in huge amounts and therefore the ability to precisely estimate a general Markov model, which conveniently avoids such complex computations. Additionally, unlike these discussed studies, we ensure the false alarm controllability without parameter tuning even in the case of the non-stationary sources.

Markov models are also frequently used for anomaly detection in temporal data [8, 25, 163, 165, 166, 193–196]. Given a Markov model with unknown parameters, the parameters are first estimated in the training phase and thereafter, if the probability of a test instance is sufficiently small compared to a threshold, then an anomaly is declared in these studies. In [193], this approach is successfully demonstrated for first order Markov models, where the extension to the desired generality with higher order is straightforward. Efficient representations for large alphabet sizes is considered in via a suffix tree used in conjunction with a finite state automata in [196]; and also an extended finite state automata in [194, 195]. In [8], anomalies are defined as labeling errors in case of the hidden Markov models and their effects on state recognitions are investigated. Another successful application is the video anomaly identification [163], where pixel based motion patterns are modelled under first order Markov statistics and the sufficiently unlikely patterns are labeled anomalous. Markov models are generally efficient and can be extended to online implementations. However, it is difficult to related the threshold parameters in these studies to the false alarm rates, and even once tuned correctly; re-adjustments is necessary, if the source statistics change. For instance, in [163], extensive Monte Carlo simulations are required to find an appropriate threshold. In contrast, our algorithm does not require parameter tuning, i.e., it is parameter-tuning free, regardless of the stationary or non-stationary source statistics and nearly guarantees the specified bearable false alarm rate (it operates at a constant false alarm rate). Although there

exist other efficient online anomaly detection methods [197–201], some of which also handle non-stationary sources with successful update procedures [197, 198], they are either heuristically developed; or not sufficiently efficient to handle big data applications or based on complicated parameter tunings. Moreover, they do not consider the false alarm rate controllability. On the other hand, the proposed algorithm operates in an online manner on a fast streaming stationary or non-stationary temporal data at a desired false alarm rate while maximizing the detection power without a parameter tuning to achieve the desired rate. In this regard, we are the first to study the problem with these practical statistical and computational constraints.

We provide the problem formulation in Section 6.1. After the proposed method is described in Section 6.2, we present our online algorithm in Section 6.3. We demonstrate the performance of our algorithm in Section 6.4 through several numerical examples and simulations. The chapter concludes with final remarks in Section 6.5.

6.1 Problem Description

We first concentrate on the stationary sources, then proceed to non-stationary settings in Section 6.3 and present examples in Section 6.4.

We consider a real valued, bounded, stationary and ergodic discrete time stochastic process X_t, i.e., $X_t \in \mathbb{R}$, and $|X_t| \leq A \in \mathbb{R}^+$. Our goal is to detect anomalous subsequences, i.e., windows, in a given realization x_t from the process X_t. For this purpose, we define a (sliding) window sequence $\{w_n\}_{n=1}^{L}$ of window length L at a time t with $w_n = x_{t-L+n}$. Note that the definition of the window sequence depends on the time t, which is not shown in "w_n" for notational simplicity. Then, we decide whether the sequence w_n is statistically consistent with the underlying process X_t, i.e., anomalous, or not.

An anomaly often occurs due to an external factor overwriting the actual data

or an abrupt change in the source statistics [9, 25]. Since the characteristics of such an external factor or a sudden change cannot be predicted beforehand, it is reasonable to assume that an anomaly can be at any point in the observation domain, i.e., it is uniformly distributed [9, 26, 120]. To detect anomalies of this kind, we formulate the problem in the Neyman-Pearson (NP) testing framework, where one minimizes the miss rate at the cost of a pre-specified false alarm rate $\tau \in [0, 1]$. The NP test in this regard declares an anomaly when

$$f(w_n) \leq \delta(\tau, L), \tag{6.1}$$

such that[1] $\delta(\tau, L) =$

$$\max \left\{ \nu : \sum_{\forall s_n : \mathcal{L}(s_n) = L} 1_{\{f(s_n) \leq \nu\}} f(s_n) \leq \tau \right\}, \tag{6.2}$$

where $f(w_n)$ is the probability mass function (p.m.f.) for a nominal window W_n from X_t and $\mathcal{L}(s_n)$ is the length of the sequence s_n.

Note that the summation in (6.2) is over all possible windows s_n of length L, i.e., $\mathcal{L}(s_n) = L$; therefore, the threshold δ depends on L. Since the process X_t is stationary, the p.m.f. $f(w_n)$ accepts a common form across all windows of X_t but its form depends on the window length. Although "$f_L(w_n)$" would be a proper notation, we drop the subscript for simplification in notation.

Since w_n is random, the log-likelihood $z \stackrel{\triangle}{=} \log f(w_n)$ is also random with the corresponding density[2] $f_Z(z)$. Then, the same test in (6.1) can be also be written in the log-likelihood domain as

$$z \leq \delta(\tau, L), \tag{6.3}$$

[1] $1_{\{h\}}$ is the indicator function such that $1_{\{h\}} = 1$, if h is TRUE; and 0, otherwise. $|.|$ is the size of a set or determinant of a matrix or an absolute value depending on the argument. All listings of the form $[.]_i$ generates a column vector and $\{.\}_i$ generates a set. $\mathbf{X}'$ is the transpose of a matrix $\mathbf{X}$. $\lfloor . \rfloor$ is the floor operation for a scalar x, i.e., $\lfloor x \rfloor = \max\{y \in \mathbb{Z} : y \leq x\}$. We use the "$\pm$" notation to refer to corresponding cases respectively, i.e., $x^{\pm} = y^{\pm} : x^{+} = y^{+}$ and $x^{-} = y^{-}$. All logarithms are natural logarithms.

[2] We simply use "density" or "distribution" interchangeably while referring to the probability density or mass function of a random variable. The log-likelihood $z = \log f(w_n)$ is also time varying, however, we drop the subscript for notational simplicity.

such that

$$\delta(\tau, L) = \max\left\{\nu : \sum_{\forall z} 1_{\{z \leq \nu\}} f_Z(z) \leq \tau\right\}.$$

The log-likelihood variable z still implicitly depends on the length L since the summation is actually over $\forall z : \exists s_n$ with $z = \log f(s_n)$ and $\mathcal{L}(s_n) = L$. This test (the Neyman-Pearson (NP) test) is the "most powerful" in the sense that it yields the highest detection rate at the false alarm level τ when the anomalies are uniformly distributed [9, 26, 120].

Our aim is to devise computationally efficient online (with linear complexity in the number of processed instance) NP tests for anomaly detection for time series.

6.2 Anomaly Detection under Markov Statistics

The problem described in Section 6.1 requires one to determine not only the log-likelihood $z = \log f(w_n)$ but also the density $f_Z(\cdot)$ of the log-likelihood in addition to the threshold δ. For this purpose, we propose to model the underlying stochastic process X_t by a Discrete Time Markov Chain (DTMC) in Section 6.2.1 and obtain the density $f_Z(\cdot)$ in Section 6.2.2. Then, we propose our anomaly detection methods in Section 6.2.3, for which an efficient algorithm is presented in Section 6.3 that sequentially operates in a truly online manner.

6.2.1 Observation Model

We model the unknown density $f(\cdot)$ by a Discrete Time Markov Chain (DTMC) that is assumed to govern the actual process X_t. Suppose that the range $[-A, A]$ bounding X_t is split into N intervals defining the states $1 \leq i \leq N$ of the underlying DTMC with the amplitude intervals $I_i = [-A + (i-1)\Delta, -A + i\Delta)$, where $\Delta = 2A/N$ is the length of each interval. We assume that at each time step, the process X_t preserves its state i, i.e., it stays in the corresponding amplitude

interval, with probability λ_i and makes an up/down or right/left transition with probability $p_{i(i+1)}$ or $p_{i(i-1)}$: $\lambda_i + p_{i(i+1)} + p_{i(i-1)} = 1$. Note that since the process X_t is bounded, no transitions are allowed out of the boundaries. The resulting birth-death type process is illustrated in Fig. 6.1. The number N of amplitude levels in this model is chosen to ensure for any realization x_t and at any time t, we have the condition $|x_t - x_{t-1}| \leq \Delta$. We emphasize that any continuous time continuous source can be forced to satisfy this if it is sampled at a sufficient rate and any discrete time source can be forced to satisfy this if it is sufficiently up-sampled [202, 203]. We also emphasize that in this study, we concentrate on discrete time sources, however, our work can be straightforwardly extended to continuous time processes. For instance, an immediate corresponding model would be obtained via a Continuous Time Markov Chain (CTMC) with exponentially distributed state holding/waiting times [204].

By the introduced DTMC model, we consider the small variations of a signal at any state i as insignificant variations, i.e., as contamination/noise, and hence discard them. Instead, we concentrate on the transitions (large variations)/waiting times (persistency) between/at states. In fact, within state variations become increasingly minute as the number N of the states/regions of the amplitude increases. For example, two different sequences $w_n^{(1)} \to y_n$ and $w_n^{(2)} \to y_n$ mapping to the same state sequence y_n, where $y_i \in \{1, 2, ..., N\}$ is the state of the i'th observation in $w_n^{(j)}$, are considered "same" up to small variations. In this sense, we can continue our derivations with only the state sequence y_n without further need to refer to the actual sequence w_n. However, to emphasize the effect of the introduced DTMC as a mapping from w_n to y_n and the corresponding equivalence in between, we opt to use "w_n" to refer to both the state sequence y_n and the actual sequence w_n simultaneously unless it is necessary to explicitly state the distinction. For example, $f(w_n)$ is -to be more precise- the probability of the state sequence y_n under the introduced DTMC model.

Based on this DTMC model, the density $f(w_n)$ of a window w_n from x_t of

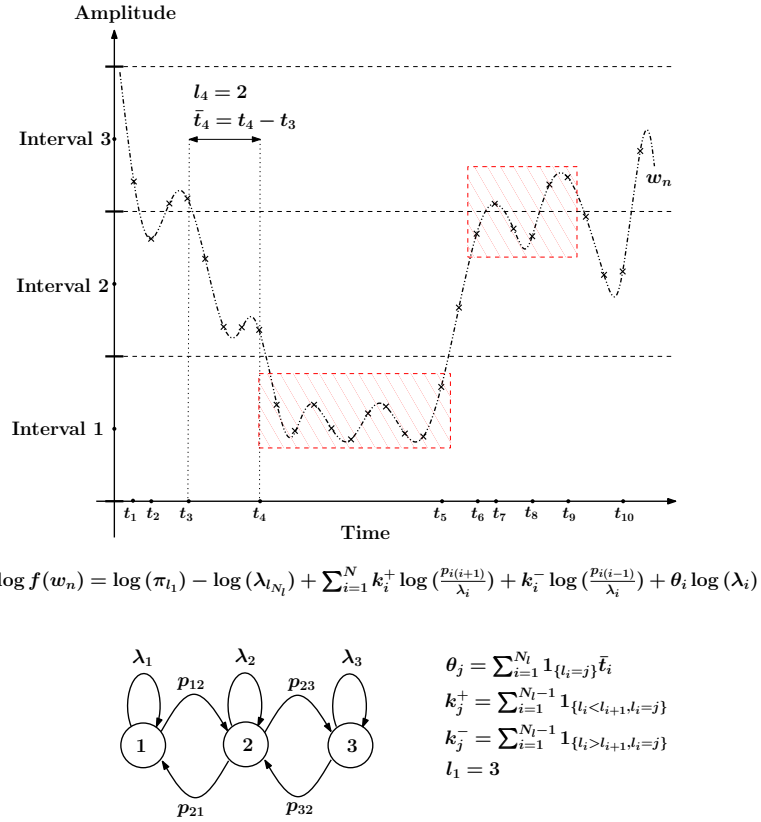


Figure 6.1: A 3-state ($N = 3$) Discrete Time Markov Chain (DTMC) modeling of the process X_t and a corresponding realization w_n, i.e., a window from x_t, is presented. The time series w_n is anomalous due to the two short-term irregularities it includes: In the boxed region on the left, it has an abnormally too long waiting time at state 1; and in the boxed region on the right, it has abnormally too many state transitions.

finite length L is given by

$$f(w_n) = \pi_{l_1} \lambda_{l_{N_l}}^{\bar{t}_{N_l}-1} \prod_{i=1}^{N_l-1} \lambda_{l_i}^{\bar{t}_i-1} p_{l_i l_{i+1}},$$

where π_{l_1} is the prior probability for the initial state l_1 that accounts for the initial conditions, $\bar{t}_i$ is the waiting time for the window w_n at the state observed right before the i'th transition, i.e.,

$$\bar{t}_i = t_i - t_{i-1}, \text{ for } 1 \leq i \leq N_l,$$

where t_i is the time of the last observation before the i'th transition with $t_0 = 0$. Also, $\{l_i\}_{i=1}^{N_l}$ with $l_i \in \{1, 2, ..., N\}$ is the corresponding sequence of states

observed before each transition. Note that the last transition is hypothetically assumed to be $t_{N_l} = L$ and N_l is the total number of transitions, which can be as small as only 1 transition. The log-likelihood is then given by

$$\begin{aligned}\log f(w_n) = &\log \pi_{l_1} + (\bar{t}_{N_l} - 1) \log \lambda_{l_{N_l}} \\ &+ \sum_{i=1}^{N_l - 1} (\bar{t}_i - 1) \log \lambda_{l_i} + \log p_{l_i l_{i+1}}. \end{aligned} \tag{6.4}$$

If we accumulate a total waiting time θ_j at state j as

$$\theta_j = \sum_{i=1}^{N_l} 1_{\{l_i = j\}} \bar{t}_i,$$

then by re-arranging (6.4) in terms of the observed unique states, we obtain

$$\begin{aligned}&\log f(w_n) = \log \pi_{l_1} - \log \lambda_{l_{N_l}} \\ &+ \sum_{i=1}^{N} k_i^+ \log \frac{p_{i(i+1)}}{\lambda_i} + k_i^- \log \frac{p_{i(i-1)}}{\lambda_i} + \theta_i \log \lambda_i, \end{aligned} \tag{6.5}$$

where $k_i = k_i^+ + k_i^-$ is the total number of state i observations in l_i with the exception that $k_{l_{N_l}} = k_{l_{N_l}}^+ + k_{l_{N_l}}^- + 1$ and k_i^+ (k_i^-) is the number of "up/down" ("right/left") transitions from state i in w_n, i.e.,

$$k_j^+ = \sum_{i=1}^{N_l - 1} 1_{\{l_i < l_{i+1}, l_i = j\}}, k_j^- = \sum_{i=1}^{N_l - 1} 1_{\{l_i > l_{i+1}, l_i = j\}}.$$

Note that the log-likelihood expression in (6.5) is exact with the convention $p_{i(i+1)} = p_{j(j-1)} = 0 \log 0 = 0$ when $i = N$ or $j = 1$ to handle the boundary conditions.

Remark: We observe that a significant reduction via the proposed observation model is possible with no or limited information loss. Since our DTMC model is a birth-death type Markov chain, the number of up and down transitions between two states must be equal due to the Global Balance (GB) equations, i.e., $k_i^+ = k_{i+1}^-$. To be more precise, $k_i^+ = k_{i+1}^- \pm 1$ is also possible depending on the first and the last states l_1 and l_{N_l}; however, we ignore those conditions for sufficiently long observations. Therefore, the accumulated total waiting times

θ_i's at each unique state as well as the corresponding number k_i^+ of transition occurrences provide sufficient statistics and are the only signal attributes to be necessarily stored, i.e., storing the actual observation w_n is unnecessary. In this sense, $\mathbf{d_w} = \{\theta_i, k_i^+\}_{i=1}^N$ is a complete set of descriptors of dimensionality $2N-1$ that is independent of the length of the sequence w_n. Note here that k_N^+=0 and the initial state l_1 are also not counted since their contribution via π_{l_1} are negligible.

We point out that the signal descriptors $\{\theta_i, k_i^+, k_i^-\}_{i=1}^N$ as well as the model parameters $\{\lambda_i, p_{i+1}, p_{i-1}\}_{i=1}^N$ and therefore the log-likelihood expression in (6.5) can be calculated/estimated sequentially in an efficient manner, cf. Section 6.3. We derive the log-likelihood density in the following Section 6.2.2 in order to later devise our anomaly detection methods in Section 6.2.3.

6.2.2 The Log-likelihood Density f_Z

In this part, we approximate the probability density of the log-likelihood $\log f(w_n)$ in order to efficiently find the threshold of the anomaly detection test formulated in Section 6.1. We start with concentrating on the random variables $k_i^\pm$, i.e., k_i^+ and k_i^-, in $\log f(w_n)$; and then continue with θ_i for this derivation.

Let us define $\hat{p}_{i(i+1)} = \frac{k_i^+}{\theta_i} = p_{i(i+1)} + \epsilon_i^+$ and $\hat{p}_{i(i-1)} = \frac{k_i^-}{\theta_i} = p_{i(i-1)} + \epsilon_i^-$, in which ϵ_i^+ and ϵ_i^- are the corresponding estimation errors. Then, we can write (6.5) as

$$\begin{aligned}
\log f(w_n) &= \log \pi_{l_1} - \log \lambda_{l_{N_l}} + \sum_{i=1}^{N} \theta_i (p_{i(i+1)} + \epsilon_i^+) \log \frac{p_{i(i+1)}}{\lambda_i} \\
&+ \theta_i (p_{i(i-1)} + \epsilon_i^-) \log \frac{p_{i(i-1)}}{\lambda_i} + \theta_i \log \lambda_i \\
&= \log \pi_{l_1} - \log \lambda_{l_{N_l}} + \sum_{i=1}^{N} \theta_i h_i + \sum_{i=1}^{N} \theta_i h_i^\epsilon,
\end{aligned}$$

where h_i is a constant with

$$h_i = p_{i(i+1)} \log \frac{p_{i(i+1)}}{\lambda_i} + p_{i(i-1)} \log \frac{p_{i(i-1)}}{\lambda_i} + \log \lambda_i$$

and h_i^ϵ is an approximation term with

$$h_i^\epsilon = \epsilon_i^+ \log \frac{p_{i(i+1)}}{\lambda_i} + \epsilon_i^- \log \frac{p_{i(i-1)}}{\lambda_i}.$$

The singularities due to the boundary conditions in our derivations are all of the form $x \log x$ evaluated at 0, which can easily be removed by the zero replacement by a limiting argument, i.e., $\lim_{x \to 0} x \log x = 0$. Hence, we have the convention $0 \log 0 = 0$. Note that these singularities can also be removed if the boundary conditions are specifically treated, which would significantly complicate the presentation without the convention $0 \log 0 = 0$.

By ergodicity of the process X_t and by weak law of large numbers (WLLN), $\epsilon_i^+, \epsilon_i^- \to 0$, as the length L of the sequence w_n goes to infinity, i.e., $h_i^\epsilon \to 0$ as $L \to \infty, \forall i$ in probability. Furthermore, we point out that as w_n is random, the log-likelihood $z = \log f(w_n)$ is also random with the density $f_Z(z)$, to which the initial condition $\log \pi_{l_1}$ has a negligible contribution for large L. Conditioned on the knowledge of θ_i, since $\hat{p}_{i(i\pm1)}$ ($\hat{p}_{i(i+1)}$ and $\hat{p}_{i(i-1)}$) are Maximum Likelihood (ML) estimators of the two parameters of a multinomial random variable (with three outcomes), they are unbiased with covariance $\frac{1}{\theta_i}\mathbf{\Sigma}_i$ and normally distributed for large θ_i, i.e., $[\epsilon_i^+, \epsilon_i^-] = \boldsymbol{\epsilon}_i | \theta_i \sim N(\mathbf{0}, \frac{1}{\theta_i}\mathbf{\Sigma}_i)$, where

$$\mathbf{\Sigma}_i = \begin{pmatrix} p_{i(i+1)}(1 - p_{i(i+1)}) & -p_{i(i+1)}p_{i(i-1)} \\ -p_{i(i+1)}p_{i(i-1)} & p_{i(i-1)}(1 - p_{i(i-1)}) \end{pmatrix}.$$

We point out that $\boldsymbol{\epsilon}_i$'s are correlated, which can be observed from the Global Balance (GB) equations from our birth-death type Markov chain, i.e., $k_i^+ = k_{i+1}^-$. Nevertheless, conditioned on the complete knowledge of $\mathbf{\Theta} = [\Theta_i]_{i=1}^N$; we know that $\boldsymbol{\epsilon}_i$'s uniformly approach 0 and the GB equations can be recovered with negligible error for sufficiently large L even by ignoring the correlations between $\boldsymbol{\epsilon}_i$'s. Therefore, we naively suppose that $Z|\mathbf{\Theta}$ is also normally distributed for large L with mean

$$\mu_\theta = \mathbf{h}_\mu \boldsymbol{\theta} = \sum_{i=1}^{N} \theta_i h_i,$$

where $\mathbf{h}'_\mu = [h_i]_{i=1}^N$ (contribution of the initial and termination conditions $\log \pi_{l_1} - \log \lambda_{l_{N_l}}$ is negligible for sufficiently long observations), $\mathbf{h}'_\mu$ is the transpose of $\mathbf{h}_\mu$; and variance

$$\sigma_\theta^2 = \mathbf{h}_\sigma \boldsymbol{\theta},$$

where

$$\mathbf{h}'_\sigma = \left[[\log \frac{p_{i(i+1)}}{\lambda_i}, \log \frac{p_{i(i-1)}}{\lambda_i}] \boldsymbol{\Sigma}_i [\log \frac{p_{i(i+1)}}{\lambda_i}, \log \frac{p_{i(i-1)}}{\lambda_i}]' \right]_{i=1}^N$$

(contribution of $\log \pi_{l_1} - \log \lambda_{l_{N_l}}$ is negligible), i.e.,

$$Z|\boldsymbol{\Theta} \sim f_{Z|\boldsymbol{\Theta}} = N(\mu_\theta, \sigma_\theta^2).$$

Similarly, the steady state probability probability π_i can be approximated [205] by the estimator $\hat{\pi}_i = \frac{\theta_i}{L} = \pi_i + \gamma_i$, which is unbiased with covariance $\frac{1}{L}\boldsymbol{\Sigma}_\gamma$, where

$$\boldsymbol{\Sigma}_\gamma = \frac{1}{L} \left\{ \sum_{i=1}^{L-1} (N-i)(\mathbf{D}_{\boldsymbol{\pi}} \mathbf{P}^i + \mathbf{P}'^i \mathbf{D}_{\boldsymbol{\pi}}) + L\mathbf{D}_{\boldsymbol{\pi}} \right\} - L\boldsymbol{\pi}\boldsymbol{\pi}',$$

$\mathbf{P}$ is the matrix of state transition probabilities consisting of the terms $p_{i(i\pm1)}$, and $\boldsymbol{\pi} = [\pi_i]_{i=1}^N$ with $\mathbf{D}_{\boldsymbol{\pi}}$ being the diagonal matrix of $\boldsymbol{\pi}$ of appropriate size. For sufficiently large L, $\boldsymbol{\gamma} = [\gamma_i]_{i=1}^N$ is normally distributed, i.e., $\boldsymbol{\gamma} \sim N(\mathbf{0}, \frac{1}{L}\boldsymbol{\Sigma}_\gamma)$ and hence,

$$\boldsymbol{\Theta} \sim f_{\boldsymbol{\Theta}} = N(L\boldsymbol{\pi}, L\boldsymbol{\Sigma}_\gamma).$$

Note that the density $f_{\boldsymbol{\Theta}}$ effectively defines a Gaussian prior on the mean μ_θ and the variance σ_θ^2 of the conditional density $f_{Z|\boldsymbol{\Theta}}$. We can further obtain a reduced set of random parameters $\boldsymbol{\Theta}^r$

$$\boldsymbol{\theta}^r = \begin{bmatrix} \mu_\theta \\ \sigma_\theta^2 \end{bmatrix} = \mathbf{H}\boldsymbol{\theta},$$

where[3] $\mathbf{H} = [\mathbf{h}'_\mu, \mathbf{h}'_\sigma]'$ with the corresponding density

$$f_{\boldsymbol{\Theta}^r} = N(L\mathbf{H}\boldsymbol{\pi}, L\mathbf{H}\boldsymbol{\Sigma}_\gamma\ \mathbf{H}').$$

[3] $\mathbf{X}'$ is the transpose of a matrix $\mathbf{X}$.

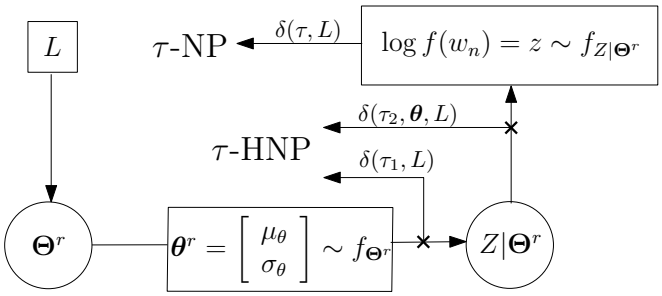


Figure 6.2: The log-likelihood density model f_Z for sufficiently large window length L based on the mixture of Gaussians in (6.6). Note that with this model, first the parameter $\boldsymbol{\theta}^r$ is sampled from $f_{\boldsymbol{\Theta}^r}$; and based on the sampled value $\boldsymbol{\theta}^r$, the log-likelihood z is sampled from $f_{Z|\boldsymbol{\Theta}^r}$.

As a result of this, we obtain

$$Z|\boldsymbol{\Theta}^r \sim f_{Z|\boldsymbol{\Theta}^r} = N(\mu_\theta, \sigma_\theta^2)$$

yielding to the (unconditional) log-likelihood density as

$$f_Z(z) = \int_{\forall \boldsymbol{\theta}^r} f_{Z|\boldsymbol{\Theta}^r}(z|\boldsymbol{\Theta}^r = \boldsymbol{\theta}^r) f_{\boldsymbol{\Theta}^r}(\boldsymbol{\theta}^r) d\boldsymbol{\theta}^r \tag{6.6}$$

is a mixture of Gaussians.

Although there is no closed form expression for the log-likelihood density f_Z, its mixture of Gaussians form suggests a generative model. In this alternative model, for a given window length L, one first samples $\boldsymbol{\theta}^r$ from the density $f_{\boldsymbol{\Theta}^r}$ and then samples, conditioned on the sampled $\boldsymbol{\theta}^r$, the log-likelihood z from the density $f_{Z|\boldsymbol{\Theta}^r}$, cf. Fig. 6.2.

6.2.3 Anomaly Detection Methods

In this section, we propose two Neyman-Pearson (NP) test based anomaly detection methods that we compare in our experiments. A) An NP test with the threshold $\delta(\tau, L)$ in (6.3) that is based on extensive Monte Carlo simulations named as "MCNP". B) A hierarchical NP test based on the derived mixture of Gaussians form of the log-likelihood density, named as "HNP", for which we

also present a computationally highly efficient online algorithm without requiring Monte Carlo simulations or parameter tuning ("Sequential HNP") in Section 6.3. Regardless of whether the source is stationary or non-stationary, the method HNP successfully achieves the desired false alarm rate while maximizing the detection power.

The proposed anomaly detection method HNP is a hierarchical application of two successive NP tests: the first one uses the marginal density of $\boldsymbol{\theta^r}$ and the second one uses the log-likelihood density conditioned on $\boldsymbol{\theta^r}$, cf. Section 6.2.3.2. During this hierarchical application, HNP allows one to analytically calculate the thresholds (of its two individual NP tests) as simple functional evaluations without requiring Monte Carlo simulations and complicated parameter tunings; and hence, yields a computationally highly efficient and online algorithm (Sequential HNP in Section 6.3). On the contrary, the NP test in (6.3) is not analytically tractable. Namely, the threshold of the NP test cannot be determined analytically (i.e. Monte Carlo simulations are necessarily used in the MCNP method) even after our simplifications in deriving the mixture of Gaussians form of the log-likelihood density. Secondly, the overall detection rate of anomalies for the test method HNP is not the best achievable when the anomalies are uniformly distributed; however, it is preferable in our study due to its efficiency. Nevertheless, the individual tests in the HNP test are all separately the most powerful as they are NP tests and the resulting HNP test is uniformly the most powerful over the variable $\boldsymbol{\theta}^r$ while achieving the desired false alarm rate τ. Moreover, when the anomalies are not uniformly distributed but appear as a change in the statistics of the underlying process X_t, the optimality in the NP test is certainly lost and the proposed method HNP outperforms the method MCNP, cf. our experiments in Section 6.4.

6.2.3.1 An NP test based on extensive Monte Carlo simulations (MCNP)

In order to avoid the effects of the imperfect Gaussian approximation in deriving the log-likelihood density f_Z when the window length L is not sufficiently large,

the first method we propose is an NP test, which is based on the exact form and the exact density of the likelihood $\log f(w_n)$ in (6.5); however, it heavily relies on extensive Monte Carlo simulations. We emphasize that this test is a generalization of the anomaly identification method in [163] to multi state birth-death type Markov chains and serves as a comparison basis in our experiments. Instead of approximating the density of $z = \log f(w_n)$ and calculating an approximate threshold for a specified false alarm rate τ, we estimate this threshold via extensive Monte Carlo simulations. In these simulations, we first randomly generate N_{mc} many samples of Z and obtain a set $\{z_k\}_{k=1}^{N_{mc}}$ of realizations. Note that while generating a sample z_k, we actually sample a sequence w_n of a fixed and specified window length L using the DTMC model and calculate $z_k = \log f(w_n)$ via (6.5). Suppose that the set $\{z_k\}_{k=1}^{N_{mc}}$ is sorted in the ascending order, i.e., $z_i \leq z_j$ for any $i \leq j$, then we estimate the true threshold in (6.3) as

$$\hat{\delta}(\tau, L) = z_{\lfloor \tau N_{mc} \rfloor},$$

which precisely provides the anomaly detection method described in Section 6.1.

6.2.3.2 A Hierarchical NP Test (HNP)

Based on the conditional independency structure that is observed as a mixture of Gaussians in the final form of the log-likelihood density in (6.6), one can intuitively separate the anomaly detection problem into two pieces. Accordingly, we finally propose a hierarchical anomaly detection test method HNP that first applies an NP test for an anomaly against $\boldsymbol{\Theta}^r$ and -if not found an anomaly- secondly applies an NP test for an anomaly against $Z|\Theta^r$. We formulate this hierarchical test as

$$\begin{aligned} &\text{(a) } \log f_{\boldsymbol{\Theta}^r}(\boldsymbol{\theta}^r) \leq \delta(\tau_1, L), \\ &\text{(b) } z \leq \delta(\tau_2, \boldsymbol{\theta}^r, L), \ \boldsymbol{\theta}^r \text{ is given,} \end{aligned}$$

such that

$$\delta(\tau_1, L) = \max \left\{ \nu : \sum_{\forall \boldsymbol{\theta}^r} 1_{\{\log f_{\boldsymbol{\Theta}^r}(\boldsymbol{\theta}^r) \leq \nu\}} f_{\boldsymbol{\Theta}^r}(\boldsymbol{\theta}^r) \leq \tau_1 \right\},$$

$$\delta(\tau_2, \boldsymbol{\theta}^r, L)$$
$$= \max\left\{\nu : \sum_{\forall z} 1_{\{z\leq\nu\}} f_{Z|\boldsymbol{\Theta}^r}(z|\boldsymbol{\Theta}^r = \boldsymbol{\theta}^r) \leq \tau_2\right\}, \forall \boldsymbol{\theta}^r,$$
$$\tau_1 + (1-\tau_1)\tau_2 \leq \tau.$$

In order to ensure an overall false alarm rate τ, we also require the condition $\tau_1 + (1-\tau_1)\tau_2 \leq \tau$, where τ_1 and τ_2 are the false alarm rates of the tests for $\boldsymbol{\theta}^r$ and the conditional observation z, respectively.

Unlike the MCNP method, the HNP method requires $\boldsymbol{\theta^r}$ in addition to the log-likelihood z that is directly observable through $\mathbf{H}\boldsymbol{\theta}$ from a window sequence w_n of length L to be tested. Then, we determine the threshold $\delta(\tau_1, L)$ from

$$\begin{aligned}\tau_1 &= Pr\{\log f_{\boldsymbol{\Theta}^r}(\boldsymbol{\theta}^r) \leq \hat{\delta}(\tau_1, L)\} \\ &= Pr\{g \geq -2\hat{\delta}(\tau_1, L) + 2C(L)\} = e^{\hat{\delta}(\tau_1,L)-C(L)},\end{aligned}$$

where g is exponentially distributed with mean 2 since the exponent of a bivariate Gaussian is chi-squared distributed with degree of freedom 2, and $C(L) = -\frac{1}{2}\log\left(4\pi^2 L^2 |\mathbf{H}\boldsymbol{\Sigma}_\gamma\ \mathbf{H}'|\right)$ is related to the normalization constant of a bivariate Gaussian. Hence, we obtain

$$\hat{\delta}(\tau_1, L) = C(L) + \log\tau_1. \tag{6.7}$$

Similarly, we determine the threshold $\delta(\tau_2, \boldsymbol{\theta}^r, L)$ from

$$\tau_2 = Pr\{z \leq \hat{\delta}(\tau_2, \boldsymbol{\theta}^r, L)\} = Q(\frac{\hat{\delta}(\tau_2, \boldsymbol{\theta}^r, L) - \mu_\theta}{\sigma_\theta}),$$

where $Q(.)$ is the cumulative distribution function for normal distribution and $\boldsymbol{\theta}^r = [\mu_\theta, \sigma_\theta^2]'$. Hence, we obtain

$$\hat{\delta}(\tau_2, \boldsymbol{\theta}^r, L) = \mu_\theta + \sigma_\theta Q^{-1}(\tau_2). \tag{6.8}$$

Finally, the overall false alarm budget τ is to be shared between τ_1 and τ_2 such that $\tau_1 + (1-\tau_1)\tau_2 = \tau$ is satisfied to maximize the detection. This is a design issue and in this work, we set $\tau_1 = \tau_2 = 1 - \sqrt{1-\tau}$.

Algorithm 3 Sequential HNP

Input: x_t, L, L_e, τ

$\tau_1 = \tau_2 = 1 - \sqrt{1-\tau}$
while x_t is sequentially streamed; and for $t \geq L_e$ **do**
    $z \leftarrow z^{(t)}$ via the update rule in (6.11)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(t)}$ via the rule in (6.10) using the window w_n
    Update the model parameters via the rule in (6.10)
    $\boldsymbol{\theta}^r = [\mu_\theta, \sigma_\theta^2]' \leftarrow \mathbf{H}\boldsymbol{\theta}$
    $\hat{\delta} \leftarrow C(L) + \log \tau_1$
    if $\log N(\boldsymbol{\theta}^r; L\mathbf{H}\boldsymbol{\pi}, L\mathbf{H}\boldsymbol{\Sigma}_\gamma\ \mathbf{H}') \leq \hat{\delta}$ **then**
        Declare anomaly at t
    else
        $\hat{\delta} \leftarrow \mu_\theta + \sigma_\theta Q^{-1}(\tau_2)$
        if $z \leq \hat{\delta}$ **then**
            Declare Anomaly at t
        end if
    end if
end while

Return: All found anomalies

We point out that the method HNP does not require Monte Carlo simulations even for varying window lengths and varying source statistics since the thresholds are derived analytically. This allows a sequential and computationally highly efficient anomaly detection algorithm, cf. Section 6.3.

6.2.3.3 Estimation of the Model Parameters

Both the proposed methods MCNP and HNP require the knowledge of the model parameters $\{(\pi_i, p_{i(i\pm1)}, \lambda_i)\}_{i=1}^N$ and recall that in this study, we are presented a sequence x_t and we would like to detect anomalous (sliding) windows in x_t. To estimate these model parameters, we use another sliding estimation windows e_n with length $L_e \gg L$, where note that $w_L = e_{L_e} = x_t$ and e_n includes w_n. Then, our model parameter estimators using the signal attributes $\{\theta_i^{e_n}, k_i^{+e_n}, k_i^{-e_n}\}_{i=1}^N$

extracted from e_n are as follows:

$$\hat{p}_{i(i\pm1)} = \frac{k_i^{\pm e_n}}{\theta_i^{e_n}}, \hat{\lambda}_i = 1 - \hat{p}_{i(i+1)} - \hat{p}_{i(i-1)}, \hat{\pi}_i = \frac{\theta_i^{e_n}}{L_e}, \tag{6.9}$$

where we use the "$\pm$" notation to refer to both cases respectively, i.e., we have two equations in (6.9) respectively for $\hat{p}_{i(i+1)}$ and $\hat{p}_{i(i-1)}$. Although the steady state distribution π_i can be analytically calculated by using the global balance equations for a birth-death type process, we also estimate it for the sake of completeness.

6.3 Sequential HNP

We observe that the estimation of the model parameters, the evaluation of the log-likelihood expression in (6.5) and the calculation of the HNP thresholds can all be sequentially performed through a simple update. Based on this observation, we present our sequential and computationally highly efficient algorithm for the proposed anomaly detection method HNP (Sequential HNP). In the following, we describe the details of the updates for parameter learning and log-likelihood calculations that are necessary in our sequential implementation.

6.3.1 Parameter Learning Updates

We need to develop sequential updates for the observation dependent variables in our parameter estimation equations in (6.9). Suppose that we have $\{\theta_i^{(t,e_n)}, k_i^{+(t,e_n)} k_i^{-(t,e_n)}\}_{i=1}^N$ calculated based on the sequence e_n at time t; and let $y_i \in \{1, 2, ..., N\}$ be the state of the i'th observation in the sequence e_n. After reading the instance x_{t+1}, we can update these variables as

$$\begin{aligned}
\theta_i^{(t+1,e_n)} &= \theta_i^{(t,e_n)} - 1_{\{y_1=i\}} + 1_{\{y_{L_e+1}=i\}}, \\
k_i^{+(t+1,e_n)} &= k_i^{+(t,e_n)} - 1_{\{y_2>y_1=i\}} + 1_{\{y_{L_e+1}>y_{L_e}=i\}}, \\
k_i^{-(t+1,e_n)} &= k_i^{-(t,e_n)} - 1_{\{y_2<y_1=i\}} + 1_{\{y_{L_e+1}<y_{L_e}=i\}},
\end{aligned} \tag{6.10}$$

and we slide the parameter estimation window e_n by one step.

6.3.2 Log-likelihood Updates

Suppose that we have $z^{(t)} = \log f(w_n)$, where $w_L = x_t$, and we would like to calculate $z^{(t+1)}$ with an update over $z^{(t)}$ without a re-calculation after reading the instance x_{t+1}. Let $y_i \in \{1, 2, ..., N\}$ be the state of the i'th observation in the sequence w_n, then it is straightforward to show that

$$
\begin{aligned}
z^{(t+1)} &= z^{(t)} + \log \frac{\hat{\pi}_{y_2}}{\hat{\pi}_{y_1}} \frac{\hat{\lambda}_{y_{L+1}}}{\hat{\lambda}_{y_1}}, \text{ if } y_1 = y_2 \text{ and } y_L = y_{L+1}, \\
z^{(t+1)} &= z^{(t)} + \log \frac{\hat{\pi}_{y_2}}{\hat{\pi}_{y_1}} \frac{\hat{\lambda}_{y_{L+1}}}{\hat{p}_{y_1 y_2}}, \text{ if } y_1 \neq y_2 \text{ and } y_L = y_{L+1}, \\
z^{(t+1)} &= z^{(t)} + \log \frac{\hat{\pi}_{y_2}}{\hat{\pi}_{y_1}} \frac{\hat{p}_{y_L y_{L+1}}}{\hat{\lambda}_{y_1}}, \text{ if } y_1 = y_2 \text{ and } y_L \neq y_{L+1}, \\
z^{(t+1)} &= z^{(t)} + \log \frac{\hat{\pi}_{y_2}}{\hat{\pi}_{y_1}} \frac{\hat{p}_{y_L y_{L+1}}}{\hat{p}_{y_1 y_2}}, \text{ otherwise.}
\end{aligned} \tag{6.11}
$$

Based on these sequential likelihood and parameter updates in (6.10) and (6.11), as well as the threshold rules of HNP in (6.7) and (6.8), we present the sequential HNP in Algorithm 3.

Remark: Unlike the method MCNP, we analytically calculate the appropriate thresholds for a specified desired false alarm rate τ without using Monte Carlo simulations in our anomaly detection method HNP. This is a strong attribute of HNP that allows a generalization of our problem formulation to non-stationary sources with real time processing capabilities in a computationally highly efficient framework. Consider a real time data stream x_t from a non-stationary stochastic process X_t such that its statistical properties change slowly in time; and we would like to sequentially detect the anomalous windows of length L in x_t in real time. Then, a reasonable approach would be to sequentially learn the model parameters in a wider and sliding window; for instance in $\{e_n\}_{n=1}^{L_e}$ with $e_{L_e} = x_t$ and $L_e \gg L$, and sequentially apply a test at every time for windows of length L, $\{w_n\}_{n=1}^{L}$ with $w_L = x_t$. This would continuously require a re-training phase for the method MCNP or for the well-known methods in the literature such as one class SVM or nearest neighbor graph based detections. However, our method

HNP can be applied immediately after observing an instance of x_t without a re-training phase and without a parameter tuning and even sequentially in a computationally highly efficient manner.

In the following section, we demonstrate the superior performance of the proposed anomaly detection methods via several numerical examples.

6.4 Experiments

In our first set of experiments, we concentrate on the achievability of the desired false alarm rate (specified by the user) by using the HNP test method. Note that the MCNP test method can achieve this rate arbitrarily accurately with extensive simulations, cf. Section 6.2.3.1. However, the proposed test HNP is designed to match the desired rate without such simulations. To this end, we devise experiments with the number of states $N \in \{2, 3, 5\}$ and the sequence length $L \in \{10, 50, 100, 250\}$ for varying desired false alarm rates $\tau \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$. For a given L, N and τ as well as a set of randomly chosen model parameters, i.e., $p_{i,i+1}, p_{i,i-1}, \lambda_i$, we generate 5000 length-L (normal) sequences directly from the Markov model and count the number of anomaly decisions (by HNP) that yields a false alarm rate. This process is repeated for 100 randomly initialized model parameters and we report the achieved average false alarm rates in Fig. 6.3.

We observe that the proposed method HNP approximates the desired false alarm rate more accurately for longer sequences since the Gaussianity assumption for the estimation error of the multinomial parameters also improves with the sequence length. Similarly, since the inter-state dependencies decrease as the number of states N increases, we obtain a better achievability for relatively larger N's. Finally, the proposed method HNP achieves the desired false alarm rate in most of the cases with $\pm 1\%$ error with the sequences of length $L = 250$ or longer. The experimental results in this part reinforce our theoretical discussions in Section 6.2.2.

	Desired false alarm rate τ											
	First row is for $N = 2$, the second row is for $N = 3$ and the third one is for $N = 5$											
	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.99
	0.014	0.049	0.091	0.175	0.268	0.352	0.445	0.543	0.671	0.803	0.943	0.998
$L = 10$	0.033	0.080	0.127	0.208	0.281	0.359	0.463	0.552	0.666	0.770	0.919	0.999
	0.050	0.110	0.165	0.250	0.338	0.419	0.508	0.604	0.711	0.822	0.936	0.998
	0.011	0.048	0.093	0.180	0.268	0.364	0.462	0.570	0.682	0.816	0.932	0.995
$L = 50$	0.020	0.061	0.107	0.195	0.281	0.371	0.468	0.569	0.674	0.793	0.912	0.998
	0.025	0.073	0.122	0.213	0.304	0.396	0.491	0.591	0.696	0.806	0.920	0.997
	0.011	0.047	0.091	0.178	0.267	0.366	0.467	0.571	0.686	0.812	0.934	0.996
$L = 100$	0.017	0.058	0.104	0.192	0.282	0.375	0.473	0.576	0.683	0.799	0.914	0.997
	0.021	0.065	0.114	0.205	0.297	0.391	0.490	0.592	0.697	0.806	0.916	0.996
	0.010	0.045	0.090	0.178	0.267	0.365	0.463	0.571	0.683	0.807	0.932	0.995
$L = 250$	0.017	0.059	0.105	0.195	0.287	0.381	0.479	0.581	0.687	0.799	0.918	0.996
	0.016	0.059	0.106	0.199	0.293	0.389	0.488	0.591	0.697	0.805	0.916	0.995

Figure 6.3: Average false alarm rates achieved by the HNP method applied to 100 randomly initialized Markov models over 5000 sequences per each model (i.e. in total 5000×100 sequences) for varying number of states N, sequence length L and desired rates.

We next compare the MCNP test method with the HNP test method in terms of the anomaly detection problem and the resulting Receiver Operating Characteristics (ROC) of the compared tests. In these comparisons, we concentrate on two types of anomalies that are both due to a sudden change in the source statistics, which we would like to detect in presence of many other nominal observations. Recall that the nominal observations are typical sequences, i.e., time series data, following a nominal Markov (DTMC) model with certain parameters. In case of the first type of anomalies, the anomalous sequences are still assumed to be drawn from a Markov model, however, whose parameters and the nominal model parameters are different. In this case, anomalies are not due to a change in the model but a change in the model parameters. On the contrary, the model assumption is also dropped in the case of the second type of anomalies, i.e., anomalies do not necessarily follow a Markov model, where the anomalies are assumed to be uniformly distributed in the observation domain and are equally likely with no prior weighting. We point out that in this second case, our purpose is to demonstrate the NP optimality since the described MCNP test is theoretically and therefore readily known to be optimal when the anomalies

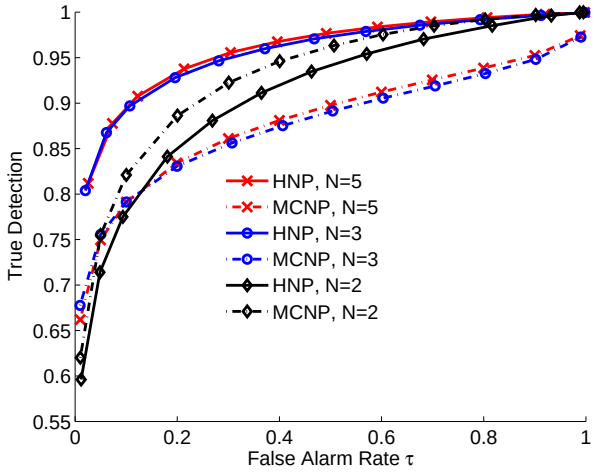


Figure 6.4: The HNP test method significantly outperforms the MCNP test method when the anomalies are in terms of the abrupt model changes (cf. $N = 3$ or $N = 5$). On the other hand, when the anomalies are uniformly distributed, both methods perform comparably (cf. $N = 2$). We point out that the HNP test method is computationally almost free, i.e., highly efficient requiring almost negligible costs, as it does not require -unlike the MCNP test method- heavy simulations to match the desired rate.

are uniformly distributed. To this end, we generate 5000 anomalous sequences of length $L = 50$, each of which is generated with respect to a randomly chosen different set of model parameters, whereas another set of 5000 normal sequences ($L = 50$) is generated with respect to a same and fixed nominal set of model parameters. We calculate the detection and false alarm rates for both of the proposed methods for varying number of states $N \in \{3, 5\}$ and desired rates τ. In Fig. 6.4, we report the average ROC rates for 100 trials. Secondly, we follow the same procedure, however; generate the anomalous sequences truly uniformly for the case $N = 2$, i.e., not specifically following a Markov model, in order to demonstrate the NP optimality. Note that in this part, we set number of states as $N = 2$ for simplicity since generating uniformly distributed anomalous sequences is unnecessarily complicated for $N \geq 3$.

We observe that if the anomalies emerge as a change in the model parameters, then the proposed method HNP detects such anomalies at significantly higher rates compared to the method MCNP, i.e., HNP outperforms as the ROC curves

for HNP are consistently above the ones for MCNP in the cases of $N = 3$ and $N = 5$. Remarkably, the NP optimality is observable only when the anomalies are truly uniformly distributed (due to its commonly used formulation [26, 120] in Section 6.1), cf. the ROC curve for $N = 2$ in Fig. 6.4.

We emphasize that the MCNP test method heavily relies on Monte Carlo simulations to achieve the desired false alarm rate, cf. Section 6.2.3.1, which is prohibitively complex for real time applications. On the contrary, the proposed HNP test method is computationally highly efficient with almost negligible costs since it does not require such Monte Carlo simulations and instead processes the data on-the-fly while successfully achieving the desired rate. This is a strong attribute of the HNP method, which makes it especially attractive when one needs to process non-stationary sources. We next concentrate on the truly sequential implementation of the proposed HNP method, where we perform experiments with non-stationary sources. In this part, we generate a sequence x_t of length $M = 100000$ that is specially exposed to drifting source statistics such that each instance x_t is produced based on the Markov model with the time varying parameters

$$\begin{bmatrix} p_{1,0}^t = 0 & p_{1,2}^t & \lambda_1^t \\ p_{2,1}^t & p_{2,3}^t = 0 & \lambda_2^t \end{bmatrix} = ((1 - \tfrac{t}{M})\mathbf{P}_1 + \tfrac{t}{M}\mathbf{P}_2)$$

with $N = 2$, where

$\mathbf{P}_1 = \begin{bmatrix} 0 & 0.3 & 0.7 \\ 0.9 & 0 & 0.1 \end{bmatrix}$, $\mathbf{P}_2 = \begin{bmatrix} 0 & 0.05 & 0.95 \\ 0.3 & 0 & 0.7 \end{bmatrix}$. Hence, the non-stationary data in this example is generated by simulating a relatively slow change from the Markov model with the transition matrix $\mathbf{P}_1$ to the one with the transition matrix $\mathbf{P}_2$. We would like to detect the anomalous subsequences of x_t of length $L \in \{50, 100\}$ using a sliding window approach, where we use wider sliding windows of length $L_e = 10 \times L$ for estimating the active set of source statistics. We explicitly inject anomalies in x_t by overwriting the first L instances starting from every L_e'th instance of x_t by the values uniformly drawn from the support set $\{1, 2\}$. We also generate a label sequence l_t, where $l_t = 1$ indicates an anomaly such that the window x_{t-L+i} includes more than $0.5 \times L$ anomalous points; and

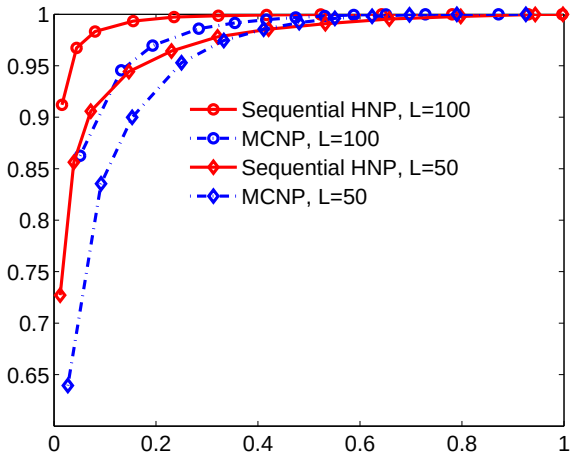


Figure 6.5: When the signal source is non-stationary, i.e., under changing/drifting source statistics, the HNP test method significantly outperforms the MCNP method at almost negligible computational costs.

otherwise, $l_t = 0$. We run the proposed sequential HNP on randomly generated 10 different x_t sequences and report the average ROC curves in Fig. 6.5. On the other hand, since the signal source in this experiment is non-stationary, i.e., the source statistics constantly change over time, the threshold for the MCNP test has to be calculated at every instance via the aforementioned Monte Carlo simulations, which is clearly practically infeasible. Instead, we estimate the threshold only once using the complete sequence x_t and also report the corresponding ROC curves for the MCNP test in Fig. 6.5. Therefore, the NP optimality has to be lost for practical purposes if the source is non-stationary. We observe that the proposed HNP test method significantly outperforms the MCNP test (as demonstrated in Fig. 6.5) at almost negligible computational costs with non-stationary data even when the anomalies are uniformly distributed, i.e., even when the NP optimality conditions are met.

6.5 Discussion

We introduce an online anomaly detection algorithm for temporal data under practical real life constraints to specifically address the contemporary big data

applications. The proposed algorithm is computationally highly efficient such that data streams at extremely fast rates can be processed in real time. Our algorithm also allows real time controllability of the Type-I error, i.e., false alarm rate, by nearly achieving a user specified false alarm rate while maximizing the detection power regardless of whether the source is stationary or non-stationary. Moreover, we do not require parameter tuning (to match the desired rates) even if the source statistics change. The proposed algorithm sequentially learns the possibly varying nominal Markov statistics in a time series and detects the anomalous, i.e., statistically sufficiently deviant, subsequences based on a Neyman-Pearson (NP) characterization for anomalies. The presented study is highly novel since we are the first to provide an online NP solution to the problem such that the NP optimality, i.e., maximum detection power at a specified false alarm rate, is nearly achieved in a truly online/sequential manner. We conclude the thesis with final remarks in the following chapter.

Chapter 7

Conclusion

State-of-the-art signal processing and machine learning systems are constantly being challenged by the "big data", i.e., large scale data of highly unstructured forms, that is generated by the contemporary applications. In this thesis, we provide theoretically established solutions for online processing, e.g., regression, prediction, classification, in such applications. We consider the online learning in adversely conditioned environments, where one is required to process possibly *corrupted* and strongly *non-stationary* data at *extremely fast rates* without a prior information or statistical assumptions on the data source.

We thoroughly study the "predicting with expert advice" and "boosting" frameworks, which have been successfully applied to online learning as well as adaptive signal processing problems and solve their issues in terms of the big data processing challenges. In this regard, we produce results in the deterministic sense such that our results are guaranteed to hold regardless of the data source, i.e., regardless of how adversely the environment is conditioned. Generally, the ensemble of constituent algorithms in these frameworks is chosen as fixed and unstructured. One must increase the ensemble size to achieve a sufficient adaptation to non-stationarity, which then increases the computational load due to the unstructured ensemble representations. On the contrary, we propose an online learning algorithm, which maintains and evolves a dynamical ensemble that is hierarchically represented by a self-organizing tree such that a computationally highly efficient implementation is obtained. Similarly, we present a deterministic

analysis and experimentally show that -both in the stationary and non-stationary settings- the proposed algorithm significantly outperforms the state-of-the-art techniques with strong performance guarantees, which is also especially suitable for applications involving large scale data.

Furthermore, we study the impact of data corruptions on the prediction performance of online learning and present two examples, i.e., studies, where the data is specifically corrupted to represent a wide range of situations in real life applications. In the first example, a Hidden Markov Model (HMM) is assumed for the data source; and in the second, we drop this model assumption and study corruptions in a model free setting. In both of these examples, we introduce versatile algorithms to impute/correct the corruptions in the batch setting, undo their detrimental affects and boost the prediction performance. We show that a corruption can be modeled as an "anomaly" and therefore, an "anomaly detection" approach is extremely useful for corruption imputation performances when used in conjunction with the online learning framework. To this end, we propose -as the first time in the literature- an online implementation of Neyman-Pearson (NP) characterization for anomaly detection such that the NP optimality, i.e., maximum detection power at a specified false alarm rate, is nearly achieved in a truly online manner. The proposed algorithm is highly novel and appropriate especially for big data applications in the case of fast streaming temporal observations due to its parameter-tuning free computational efficient design with the practical NP constraints under stationary or non-stationary source statistics.

In this thesis, we provide a comprehensive treatment of the recently emerging big data challenges of the contemporary applications and introduce effective and efficient solutions. For instance, as the first time in the literature, we present the online implementation of the Neyman-Pearson characterization of anomaly/corruption detection for online controllability of the Type-I error at extremely high rates. Similarly, as another example, the introduced highly dynamical self-organizing tree based ensemble method for online classification/regression is highly superior over the competing techniques. Thus, we strongly believe that this doctor of philosophy dissertation greatly contributes to and shall attract attention from signal processing and machine learning communities.

Bibliography

[1] H. Ozkan, M. Donmez, S. Tunc, and S. Kozat, "A deterministic analysis of an online convex mixture of experts algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–1, 2014.

[2] H. Ozkan, N. Vanli, and S. Kozat, "Online classification with self organizing decision tree structures," *Under review in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[3] H. Ozkan and S. Kozat, "Online anomaly detection under markov statistics with controllable type-i error," *Under review in IEEE Transactions on Signal Processing*, 2015.

[4] O. Yilmaz, I. Ozsarac, O. Gunay, and H. Ozkan, "Cognitive inspired real time vision core," *Under review in Cognitive Computation*, 2015.

[5] F. Porikli and H. Ozkan, "Data driven frequency mapping for computationally scalable object detection," in *IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pp. 30–35, 2011.

[6] H. Ozkan, M. Donmez, O. Pelvan, A. Akman, and S. Kozat, "Competitive and online piecewise linear classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3452–3456, May 2013.

[7] M. Donmez, H. Ozkan, and S. Kozat, "Transient analysis of convexly constrained mixture methods," in *IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–5, Sept 2012.

[8] H. Ozkan, A. Akman, and S. Kozat, "A novel and robust parameter training approach for hmms under noisy and partial access to states," *Signal Processing*, vol. 94, pp. 490–497, 2014.

[9] H. Ozkan, O. Pelvan, and S. Kozat, "Data imputation through the identification of local anomalies," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–1, 2015.

[10] H. Ozkan, A. Akman, and S. Kozat, "Hidden markov model training with side information," in *IEEE conference on Signal Processing and Communications Applications*, pp. 1–4, 2012.

[11] V. Poor, *An Introduction to Signal Detection and Estimation*. Springer Science & Business Media, 1994.

[12] C. Bishop and N. Nasrabadi, *Pattern recognition and machine learning*, vol. 1. Springer New York, 2006.

[13] A. Sayed, "Fundamentals of adaptive filtering," *John Wiley and Sons*, 2003.

[14] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press Cambridge, 2006.

[15] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, pp. 23–37, Springer, 1995.

[16] J. Arenas-Garcia, A. Figueiras-Vidal, and A. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Transactions on Signal Processing*, vol. 54, pp. 1078–1090, March 2006.

[17] S. Kozat, A. Erdogan, A. Singer, and A. Sayed, "Steady state MSE performance analysis of mixture approaches to adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4050–4063, August 2010.

[18] S. Kozat, "Competitive signal processing," *Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois*, 2004.

[19] J. Kivinen and M. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Journal of Information and Computation*, vol. 132, pp. 1–62, January 1997.

[20] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. Helmbold, R. Schapire, and M. Warmuth, "How to use expert advice," *Journal of the ACM*, vol. 44, no. 3, pp. 427–485, 1997.

[21] N. Littlestone and M. Warmuth, "The weighted majority algorithm," *Information and computation*, vol. 108, no. 2, pp. 212–261, 1994.

[22] J. Kolter and M. Maloof, "Using additive expert ensembles to cope with concept drift," in *International Conference on Machine learning*, pp. 449–456, ACM, 2005.

[23] J. Kolter and M. Maloof, "Dynamic weighted majority- an ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[24] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[25] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, p. 15, 2009.

[26] C. Scott and R. Nowak, "Learning minimum volume sets," *Journal of Machine Learning Research*, vol. 7, pp. 665–704, 2006.

[27] S. Wan, "Parameter incremental learning algorithm for neural networks," *IEEE Transactions on Neural Networks*, vol. 17, pp. 1424–1438, November 2006.

[28] N. Liang, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, pp. 1411–1423, November 2006.

[29] W. Wan, "Implementing online natural gradient learning: Problems and solutions," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 317–329, 2006.

[30] T. Silva and L. Zhao, "Stochastic competitive learning in complex networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 385–398, March 2012.

[31] F. Cattivelli and A. Sayed, "Diffusion lms strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.

[32] S. Osgouei and M. Geravanchizadeh, "Speech enhancement using convex combination of fractional least-mean-squares algorithm," *International Symposium on Telecommunications*, pp. 869–872, 2010.

[33] N. Takahashi, I. Yamada, and A. Sayed, "Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4795–4810, 2010.

[34] S. Yuksel, J. Wilson, and P. Gader, "Twenty years of mixture of experts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1177–1193, 2012.

[35] A. Gyorgy, T. Linder, and G. Lugosi, "Tracking the best quantizer," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1604–1625, 2008.

[36] N. Littlestone, "Mistake bounds and logarithmic linear-threshold learning algorithms," *Ph.D. dissertation, University of California, Santa Cruz, Computer Research Laboratory*, 1989.

[37] N. Cesa-Bianchi, P. Long, and M. Warmuth, "Worst-case quadratic loss bounds for prediction using linear functions and gradient descent," *IEEE Transaction on Neural Networks*, vol. 7, pp. 604–619, May 1996.

[38] S. Kwak, B. Han, and J. Han, "On-line video event detection by constraint flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1174–1186, June 2014.

[39] G. Pillonetto, F. Dinuzzo, and G. De Nicolao, "Bayesian online multitask learning of gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 193–205, Feb 2010.

[40] K. Zhang, L. Zhang, and M. Yang, "Fast compressive tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 2002–2015, Oct 2014.

[41] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1409–1422, July 2012.

[42] J. Wang and V. Saligrama, "Local supervised learning through space partitioning," in *Advances in Neural Information Processing Systems*, pp. 91–99, 2012.

[43] V. Vovk, "Aggregating strategies," in *Third Workshop on Computational Learning Theory*, pp. 371–383, Morgan Kaufmann, 1990.

[44] S. Chen, H. Lin, and C. Lu, "An online boosting algorithm with theoretical justifications," *International Conference on Machine Learning*, 2012.

[45] Q. Bai, H. Lam, and S. Sclaroff, "A bayesian framework for online classifier ensemble," in *Proceedings of the 31st International Conference on Machine Learning*, pp. 1584–1592, JMLR Workshop and Conference Proceedings, 2014.

[46] N. Oza and S. Russell, "Online bagging and boosting," in *Artificial Intelligence and Statistics*, pp. 105–112, 2001.

[47] C. Leistner, A. Saffari, P. Roth, and H. Bischof, "On robustness of on-line boosting-a competitive study," in *IEEE 12th International Conference on Computer Vision Workshops*, pp. 1362–1369, 2009.

[48] M. Grbovic and S. Vucetic, "Tracking concept change with incremental boosting by minimization of the evolving exponential loss," in *Machine Learning and Knowledge Discovery in Databases*, pp. 516–532, Springer, 2011.

[49] W. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[50] L. Brieman, J. Friedman, C. Stone, and R. Olsen, *Classification and Regression Trees*. Chapman & Hall, 1984.

[51] C. Scott and R. Nowak, "Minimax-optimal classification with dyadic decision trees," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1335–1353, 2006.

[52] C. Strobl, A. Boulesteix, and T. Augustin, "Unbiased split selection for classification trees based on the gini index," *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 483–501, 2007.

[53] J. Gama, "Functional trees," *Machine Learning*, vol. 55, no. 3, pp. 219–250, 2004.

[54] S. Kozat, A. Singer, and G. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3730–3745, 2007.

[55] E. Takimoto, A. Maruoka, and V. Vovk, "Predicting nearly as well as the best pruning of a decision tree through dyanamic programming scheme," *Theoretical Computer Science*, vol. 261, pp. 179–209, 2001.

[56] Y. Yilmaz and S. Kozat, "Competitive randomized nonlinear prediction under additive noise," *IEEE Signal Processing Letters*, vol. 17, no. 4, pp. 335–339, 2010.

[57] D. Vanli and S. Kozat, "A comprehensive approach to universal nonlinear regression based on trees," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5471–5486, 2013.

[58] S. Dasgupta and Y. Freund, "Random projection trees for vector quantization," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3229–3242, 2009.

[59] I. Mukherjee, C. Rudin, and R. Schapire, "The rate of convergence of adaboost," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2315–2347, 2013.

[60] R. Pelossof, M. Jones, I. Vovsha, and C. Rudin, "Online coordinate boosting," in *IEEE International Conference on Computer Vision Workshops*, pp. 1354–1361, 2009.

[61] X. Liu and T. Yu, "Gradient feature selection for online boosting," in *IEEE International Conference on Computer Vision*, pp. 1–8, 2007.

[62] B. Babenko, M. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1619–1632, Aug 2011.

[63] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *European Conference on Computer Vision*, pp. 234–247, Springer, 2008.

[64] H. Grabner and H. Bischof, "On-line boosting and vision," in *Computer Vision and Pattern Recognition*, vol. 1, pp. 260–267, 2006.

[65] B. Babenko, M. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990, 2009.

[66] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof, "Online multiclass lpboost," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3570–3577, 2010.

[67] B. Babenko, M. Yang, and S. Belongie, "A family of online boosting algorithms," in *IEEE International Conference on Computer Vision Workshops*, pp. 1346–1353, 2009.

[68] D. Haussler, J. Kivinen, and M. Warmuth, "Sequential prediction of individual sequences under general loss functions," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1906–1925, 1998.

[69] M. Herbster and M. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.

[70] C. Monteleoni and T. Jaakkola, "Online learning of non-stationary sequences," in *In Advances in Neural Information Processing Systems*, 2003.

[71] A. Gyorgy, T. Linder, and G. Lugosi, "Efficient tracking of large classes of experts," *IEEE Transactions on Information Theory*, vol. 58, no. 11, pp. 6709–6725, 2012.

[72] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Machine Learning*, vol. 26, no. 1, pp. 5–23, 1997.

[73] O. Bousquet and M. Warmuth, "Tracking a small set of experts by mixing past posteriors," *The Journal of Machine Learning Research*, vol. 3, pp. 363–396, 2003.

[74] L. Minku, A. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010.

[75] H. Wang, W. Fan, P. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *International Conference on Knowledge Discovery and Data Mining*, pp. 226–235, ACM, 2003.

[76] N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377–382, ACM, 2001.

[77] F. Chu and C. Zaniolo, "Fast and light boosting for adaptive mining of data streams," in *Advances in Knowledge Discovery and Data Mining*, pp. 282–292, Springer, 2004.

[78] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 139–148, ACM, 2009.

[79] A. Pocock, P. Yiapanis, J. Singer, M. Luján, and G. Brown, "Online non-stationary boosting," in *Multiple Classifier Systems*, pp. 205–214, Springer, 2010.

[80] V. Attar, P. Sinha, and K. Wankhade, "A fast and light classifier for data streams," *Evolving Systems*, vol. 1, no. 3, pp. 199–207, 2010.

[81] R. Servedio, "Smooth boosting and learning with malicious noise," *The Journal of Machine Learning Research*, vol. 4, pp. 633–648, 2003.

[82] A. Aho and N. Sloane, "Some doubly exponential sequences," *Fibonacci Quarterly*, vol. 11, pp. 429–437, 1970.

[83] D. Vanli, M. Sayin, and S. Kozat, "Predicting nearly as well as the optimal twice differentiable regressor," *CoRR*, vol. abs/1401.6413, 2014.

[84] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, pp. 13–30, March 1963.

[85] F. Willems, Y. Shtarkov, and T. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, pp. 653–664, May 1995.

[86] Y. Freund and R. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

[87] K. Lee and J. Lee, "A study on imm with nphmm and an application to speech enhancement," *Signal Processing*, vol. 84, pp. 1701 – 1707, 2004.

[88] K. Lee and J. Rheem, "Smooth approach using forward-backward kalman filter with markov switching parameters for speech enhancement," *Signal Processing*, vol. 80, pp. 2579–2588, Dec 2000.

[89] D. Sun, L. Deng, and C. Wu, "State-dependent time warping in the trended hidden markov model," *Signal Processing*, vol. 39, pp. 263–275, Sep 1994.

[90] M. Moore and M. Savic, "Speech reconstruction using a generalized hsmm (ghsmm)," *Digital Signal Processing*, vol. 14, pp. 37 – 53, 1 2004.

[91] D. Cutting, J. Kuipec, J. Pedersen, and P. Sibun, "A practical part-of-speech tagger," *Proc. Third Conf. Applied Natural Language Processing*, pp. 133 – 140, 1992.

[92] P. Woodland and D. Povey, "Large scale discriminative training of hidden markov models for speech recognition," *Computer Speech and Language*, vol. 16, pp. 25 – 47, 1 2002.

[93] S. Kozat, K. Visweswariah, and R. Gopinath, "Efficient, low latency adaptation for speech recognition," *IEEE International Conference on Acoustic Speech and Signal Processing*, pp. 777 – 780, 4 2007.

[94] E. Birney, "Hidden markov models in biological sequence analysis," *IBM journal of Research and Development*, vol. 45, p. 449, 2001.

[95] V. Fonzo, F. Aluffi-Pentini, and V. Parisi, "Hidden markov models in bioinformatics," in *Current Bioinformatics*, pp. 49 – 61, 2 2007.

[96] L. Bordes and P. Vandekerkhove, "Statistical inference for partially hidden markov models," *Commun. in Statistics*, vol. 34, pp. 1081–1104, 2005.

[97] B. Merialdo, "Tagging english text with a probabilistic model," *Comput. Linguist.*, vol. 20, pp. 155–171, 6 1994.

[98] D. Elworthy, "Does baum-welch re-estimation help taggers?," in *Proceedings of the fourth conference on Applied natural language processing*, ANLC '94, pp. 53–58, 1994.

[99] K. Seymore, A. Mccallum, and R. Rosenfeld, "Learning hidden markov model structure for information extraction," *In AAAI 99 Workshop on Machine Learning for Information Extraction*, pp. 37–42, 1999.

[100] Y. Ephraim and N. Merhav, "Hidden markov processes," *IEEE Transactions on Information Theory*, vol. 48, pp. 1518–1569, 2002.

[101] K. Thulasiraman and M. Swamy, *Graphs: theory and algorithms*. John Wiley & Sons, 2011.

[102] E. Baccarelli and R. Cusani, "Recursive kalman-type optimal estimation and detection of hidden markov chains," *Signal Processing*, vol. 51, no. 1, pp. 55 – 64, 1996.

[103] T. Scheffer and S. Wrobel, "Active learning of partially hidden markov models," *Proc. of ECML/PKDD Workshop on Instance Selection*, 2001.

[104] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2008.

[105] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1 – 38, 1977.

[106] L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, pp. 164–171, 2 1970.

[107] S. Forchhammer and J. Rissanen, "Partially hidden markov models," *IEEE Transactions on Information Theory*, vol. 42, pp. 1253–1256, 1996.

[108] O. Chapelle, B. Schlkopf, and A. Zien, "Semi-supervised learning (adaptive computation and machine learning)," *MIT Press*, 9 2006.

[109] L. Baum and J. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology," *Bulletin of the American Mathematical Society*, vol. 73, pp. 360 – 363, 1967.

[110] L. Baum and G. Sell, "Growth functions for transformations on manifolds," *Pacific J. Math*, vol. 27, pp. 211 – 227, 1968.

[111] S. Levinson, L. Rabiner, and M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," *Bell Sys. Tech. Journal*, vol. 62, no. 4, 1983.

[112] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260 – 269, 4 1967.

[113] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268 – 278, 3 1973.

[114] R. He, W. Zheng, B. Hu, and X. Kong, "Two-stage nonnegative sparse representation for large-scale face recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 1, pp. 35–46, 2013.

[115] S. Zafeiriou, G. Tzimiropoulos, M. Petrou, and T. Stathaki, "Regularized kernel discriminant analysis with a robust kernel for face recognition and verification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 3, pp. 526–534, 2012.

[116] S. Li and J. Lu, "Face recognition using the nearest feature line method," *IEEE Transactions on Neural Networks*, vol. 10, no. 2, pp. 439–443, 1999.

[117] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 743–761, 2012.

[118] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, pp. 1443–1471, 2001.

[119] G. Li, C. Wen, Z. Li, A. Zhang, F. Yang, and K. Mao, "Model-based online learning with kernels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 3, pp. 356–369, 2013.

[120] M. Zhao and V. Saligrama, "Anomaly detection with score functions based on nearest neighbor graphs," *Advances in Neural Information Processing Systems*, 2009.

[121] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems," in *IEEE Sixth International Conference on Data Mining*, pp. 488–498, 2006.

[122] V. Saligrama and M. Zhao, "Local anomaly detection," in *International Conference on Artificial Intelligence and Statistics*, pp. 969–983, 2012.

[123] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[124] J. Kim and C. Scott, "L2 kernel classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1822–1831, 2010.

[125] J. Sun and S. Keates, "Canonical correlation analysis on data with censoring and error information," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 1909–1919, 2013.

[126] Y. Wang and J. Hu, "Global ridge orientation modeling for partial fingerprint identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 72–87, 2011.

[127] D. Rubin, *Multiple Imputation for nonresponse in surveys*. Wiley Series in Probability and Mathematical Statistics, Wiley, 1987.

[128] R. A. Little and D. Rubin, *Statistical analysis with missing data*, vol. 539. Wiley New York, 1987.

[129] O. Troyanskaya, M. Cantor, and G. Sherlock, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17, pp. 520–525, 2001.

[130] P. Smolensky, *Information processing in dynamical systems: Foundations of harmony theory*. Department of Computer Science at University of Colorado, Boulder, 1986.

[131] G. Batista and M. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence*, vol. 17, pp. 519–533, 2003.

[132] J. Besag, "Statistical analysis of non-lattice data," *The statistician*, vol. 24, pp. 179–195, 1975.

[133] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency networks for inference, collaborative filtering, and data visualization," *The Journal of Machine Learning Research*, vol. 1, pp. 49–75, 2001.

[134] B. Marlin, *Missing data problems in machine learning*. PhD thesis, Department of Computer Science at University of Toronto, Toronto, 2008.

[135] Z. Ghahramani and M. Jordan, "Supervised learning from incomplete data via an em approach," in *Advances in Neural Information Processing Systems*, 1994.

[136] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu, "Missing value estimation for mixed-attribute data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, pp. 110–121, 2011.

[137] Q. Wang and J. Rao, "Empirical likelihood-based inference under imputation for missing response data," *The Annals of Statistics*, vol. 30, pp. 896–924, 2002.

[138] A. Rajwade, A. Rangarajan, and A. Banerjee, "Image denoising using the higher order singular value decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 849–862, 2013.

[139] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition*, 2005.

[140] Y. Weiss and W. Freeman, "What makes a good model of natural images?," in *Computer Vision and Pattern Recognition*, 2007.

[141] S. Lyu and E. Simoncelli, "Statistical modeling of images with fields of gaussian scale mixtures," in *Advances in Neural Information Processing Systems*, 2006.

[142] P. Gallinari, Y. LeCun, S. Thiria, and F. FogelmanSoulie, "Memoires associatives distribuees," *Proceedings of Cognitiva*, vol. 87, p. 93, 1987.

[143] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Advances in Neural Information Processing Systems*, 2008.

[144] M. Ranzato, V. Mnih, J. Susskind, and G. Hinton, "Modeling natural images using gated mrfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2206–2222, 2013.

[145] P. Toh and A. Forrest, "Occlusion detection in early vision," in *Computer Vision Proceedings*, 1990.

[146] N. Yung and A. Lai, "Detection of vehicle occlusion using a generalized deformable model," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 154–157, 1998.

[147] C. Pang, W. Lam, and N. Yung, "A novel method for resolving vehicle occlusion in a monocular traffic-image sequence," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, pp. 129–141, 2004.

[148] W. Zhang, J. Wu, X. Yang, and X. Fang, "Multilevel framework to detect and handle vehicle occlusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 161–174, 2008.

[149] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 349–361, 2001.

[150] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. Gavrila, "Multi-cue pedestrian classification with partial occlusion handling," in *Computer Vision and Pattern Recognition*, 2010.

[151] X. Wang, T. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *IEEE 12th International Conference on Computer Vision*, 2009.

[152] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, "Minimum error bounded efficient 1 tracker with occlusion detection," in *Computer Vision and Pattern Recognition*, 2011.

[153] M. Donmez and S. Kozat, "Steady state and transient mse analysis of convexly constrained mixture methods," *IEEE Transactions on Signal Processing*, vol. 60, pp. 3314–3321, 2012.

[154] S. Godsill, A. Doucet, and M. West, "Maximum a posteriori sequence estimation using monte carlo particle filters," *Annals of the Institute of Statistical Mathematics*, vol. 53, pp. 82–96, 2001.

[155] Y. Bar-Shalom and X. Li, *Multitarget-multisensor tracking : Principles and techniques*. New York: Academic Press, 1995.

[156] J. Mendel and S. Burrus, *Maximum-Likelihood Deconvolution: A Journey Into Model-Based Signal Processing*. Signal Processing and Digital Filtering Series, Springer Verlag GmbH, 1990.

[157] G. Chantas, N. Galatsanos, and A. Likas, "Bayesian restoration using a new nonstationary edge-preserving image prior," *IEEE Transactions on Image Processing*, vol. 15, pp. 2987–2997, 2006.

[158] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition*, 2005.

[159] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, pp. 205–220, 1992.

[160] B. Silverman, *Density estimation for statistics and data analysis*, vol. 26. CRC press, 1986.

[161] H. Wang, M. Tang, Y. Park, and C. Priebe, "Locality statistics for anomaly detection in time series of graphs," *IEEE Transactions on Signal Processing*, vol. 62, no. 3, pp. 703–717, 2014.

[162] M. Filippone and G. Sanguinetti, "A perturbative approach to novelty detection in autoregressive models," *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 1027–1036, 2011.

[163] V. Saligrama, J. Konrad, and P. Jodoin, "Video anomaly identification," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 18–33, 2010.

[164] J. Lehtomaki, J. Vartiainen, M. Juntti, and H. Saarnisaari, "Cfar outlier detection with forward methods," *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4702–4706, 2007.

[165] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, 2012.

[166] M. Gupta, J. Gao, C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014.

[167] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Anomaly detection in wireless sensor networks," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 34–40, 2008.

[168] V. Jumutc and J. Suykens, "Multi-class supervised novelty detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2510–2523, 2014.

[169] X. Ding, Y. Li, A. Belatreche, and L. Maguire, "Novelty detection using level set methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 576–588, 2015.

[170] Y. Chen, X. Dang, H. Peng, H. Bart, and H. Bart, "Outlier detection with the kernelized spatial depth function," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 288–305, 2009.

[171] H. Ferdowsi, S. Jagannathan, and M. Zawodniok, "An online outlier identification and removal scheme for improving fault detection performance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 908–919, 2014.

[172] B. Liu, Y. Xiao, P. Yu, L. Cao, Y. Zhang, and Z. Hao, "Uncertain one-class learning and concept summarization learning on uncertain data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 468–484, 2014.

[173] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, 2002.

[174] M. Basseville, I. Nikiforov, and et al, *Detection of abrupt changes: theory and application*, vol. 104. Prentice Hall Englewood Cliffs, 1993.

[175] J. Hamilton, *Time series analysis*, vol. 2. Princeton University Press, 1994.

[176] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," in *Advances in Knowledge Discovery and Data Mining*, pp. 813–822, Springer, 2009.

[177] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 427–438, 2000.

[178] A. Hero, "Geometric entropy minimization (gem) for anomaly detection and localization," in *Advances in Neural Information Processing Systems*, pp. 585–592, 2006.

[179] L. Ghaoui, M. Jordan, and G. Lanckriet, "Robust novelty detection with single-class mpm," in *Advances in Neural Information Processing Systems*, pp. 905–912, 2002.

[180] C. Bennett and K. Campbell, "A linear programming approach to novelty detection," *Advances in Neural Information Processing Systems*, vol. 13, p. 395, 2001.

[181] E. Keogh, J. Lin, S. Lee, and H. Van Herle, "Finding the most unusual time series subsequence: algorithms and applications," *Knowledge and Information Systems*, vol. 11, no. 1, pp. 1–27, 2007.

[182] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *IEEE International Conference on Data Mining*, 2005.

[183] L. Wei, E. Keogh, and X. Xi, "Saxually explicit images: Finding unusual shapes," in *International Conference on Data Mining*, pp. 711–720, 2006.

[184] Y. Bu, O. Leung, A. Fu, E. Keogh, J. Pei, and S. Meshkin, "Wat: Finding top-k discords in time series database," in *SDM*, pp. 449–454, SIAM, 2007.

[185] A. Fu, O. Leung, E. Keogh, and J. Lin, "Finding time series discords based on haar transform," in *Advanced Data Mining and Applications*, pp. 31–41, Springer, 2006.

[186] J. Lin, E. Keogh, A. Fu, and H. Van Herle, "Approximations to magic: Finding unusual medical time series," in *IEEE Symposium on Computer-Based Medical Systems*, pp. 329–334, 2005.

[187] E. Keogh, S. Lonardi, and C. Ratanamahatana, "Towards parameter-free data mining," in *International Conference on Knowledge Discovery and Data Mining*, pp. 206–215, ACM, 2004.

[188] D. Yankov, E. Keogh, and U. Rebbapragada, "Disk aware discord discovery: finding unusual time series in terabyte sized datasets," *Knowledge and Information Systems*, vol. 17, no. 2, pp. 241–262, 2008.

[189] X. Chen and Y. Zhan, "Multi-scale anomaly detection algorithm based on infrequent pattern of time series," *Journal of Computational and Applied Mathematics*, vol. 214, no. 1, pp. 227–237, 2008.

[190] C. Shahabi, X. Tian, and W. Zhao, "Tsa-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data," in *International Conference on Scientific and Statistical Database Management*, pp. 55–68, 2000.

[191] L. Wei, N. Kumar, V. Lolla, E. Keogh, S. Lonardi, and R. Chotirat, "Assumption-free anomaly detection in time series.," in *SSDBM*, vol. 5, pp. 237–242, 2005.

[192] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams," in *International Conference on Knowledge Discovery and Data Mining*, pp. 336–345, ACM, 2003.

[193] N. Ye and et al., "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166, p. 169, 2000.

[194] C. Michael and A. Ghosh, "Two state-based approaches to program-based anomaly detection," in *Annual Conference on Computer Security Applications*, pp. 21–30, 2000.

[195] V. Chandola, V. Mithal, and V. Kumar, "Comparative evaluation of anomaly detection techniques for sequence data," in *International Conference on Data Mining*, pp. 743–748, 2008.

[196] C. Marceau, "Characterizing the behavior of a program using multiple-length n-grams," in *Proceedings of the 2000 workshop on New security paradigms*, pp. 101–110, ACM, 2001.

[197] A. Pawling, P. Yan, J. Candia, T. Schoenharl, and G. Madey, "Anomaly detection in streaming sensor data," *Intelligent Techniques for Warehousing and Mining Sensor Network Data*, pp. 99–117, 2008.

[198] D. Pokrajac, A. Lazarevic, and L. Latecki, "Incremental local outlier detection for data streams," in *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 504–515, 2007.

[199] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 320–324, 2000.

[200] D. Hill, B. Minsker, and E. Amir, "Real-time bayesian anomaly detection for environmental sensor data," *Proceedings of the Congress-International Association for Hydraulic Research*, vol. 32, no. 2, p. 503, 2007.

[201] R. Laxhammar and G. Falkman, "Online learning and sequential anomaly detection in trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1158–1173, 2014.

[202] B. Moser and T. Natschlager, "On stability of distance measures for event sequences induced by level-crossing sampling," *IEEE Transactions on Signal Processing*, vol. 62, no. 8, pp. 1987–1999, 2014.

[203] D. Morgan, "On level-crossing excursions of gaussian low-pass random processes," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3623–3632, 2007.

[204] S. Karlin, *A First Course in Stochastic Processes*. Academic Press, 2014.

[205] M. Xue and S. Roy, "Spectral and graph-theoretic bounds on steady-state-probability estimation performance for an ergodic markov chain," in *American Control Conference*, pp. 2399–2404, 2011.