VILNIUS UNIVERSITY

Indrė Žliobaitė

ADAPTIVE TRAINING SET FORMATION

Doctoral dissertation
Physical sciences, informatics (09P)

Vilnius, 2010

The dissertation work was carried out at Vilnius University from 2006 to 2010 in cooperation with Bangor University (UK) and Eindhoven University of Technology (the Netherlands) researchers.

Scientific supervisor:

prof. habil. dr. Šarūnas Raudys (Vilnius University, physical sciences, informatics - 09P)

VILNIAUS UNIVERSITETAS

Indrė Žliobaitė

ADAPTYVUS MOKYMO IMTIES FORMAVIMAS

Daktaro disertacija
Fiziniai mokslai, informatika (09P)

Vilnius, 2010

Disertacija rengta 2006 - 2010 metais Vilniaus universitete bendradarbiaujant su Bangoro universiteto (Didžioji Britanija) ir Eindhoveno technologijų universiteto (Nyderlandai) mokslininkais.

Mokslinis vadovas:

prof. habil. dr. Šarūnas Raudys (Vilniaus universitetas, fiziniai mokslai, informatika - 09P)

Acknowledgements

The journey has been challenging and exciting. My warm gratitude goes to the people who inspired me and helped in many ways.

I kindly thank my supervisor prof. habil. dr. Šarūnas Raudys for introducing and showing the beauty of pattern recognition, for challenging discussions, advices and for the invaluable freedom I had in my research. I am grateful to dr. Ludmila I. Kuncheva for working with me. I learned so much from you, it has been an honor. I thank my colleagues and coauthors dr. Mykola Pechenizkiy and Jorn Bakker. The research has been and is fun. I am deeply grateful to dr. Tomas Krilavičius for technical and content support. And largely for inspiration.

I am grateful to the faculty: Rita Ragaišienė, Domicelė Jacikevičienė for kind support. I thank the head of Post-Graduate Studies Office Stanislava Vaškevičienė for keeping the doors open for me. My special thank is addressed to the head of the department of Informatics doc. dr. Rimantas Vaicekauskas for support and encouragement in organizing the defence process.

I thank colleagues from the bank Mindaugas Steikūnas, Justas Skominas, Andrius Tilvytis for tolerance.

Ačiū draugams, artimiesiems ir šeimai už palaikymą.

Indrė Žliobaitė
Vilnius
22nd February 2010

Table of Contents

Notation

A amplitude caused by the feeding screw
a acceleration of a mass change
α_1 the weight of distance in space
α_2 the weight of distance in time
α_{feed} phase of the fluctuations caused by the feeding screw
α_{mix} phase of the fluctuations caused by the mixing screw
B amplitude caused by the mixing screw
$(C_1, C_2, \ldots, C_c)$ categories (of the product) within the context
$C^{(M)}$ a particular classifier (NMC)
c_i class label (the i^{th} class)
$\mathcal{D}_{ij}$ time and space similarity funcion between instances $\mathbf{X}_i$ and $\mathbf{X}_j$
$\Delta_t^{(l)} = \mathbf{x}_t - \mathbf{x}_{t-l}$ l^{th} order difference of a signal
$d^{(S)}$ distance in space
$d^{(\mathcal{T})}$ distance in time
$e(t)$ random peaked high amplitude noise
e_t is the prediction error at time t
$\mathcal{F}$ functional mapping
F_i the ithfeature
$(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m)$ base learners
h^D the training set threshold
h_t is the parameter vector
k size of the neighborhood (number of the nearest neighbors)
κ index for a sudden change
L a lag for calculating a moving average
$\mathcal{L}_t$ a learner trained at time t
$\mathbf{M}$ training set
m_0 is the initial mass at time t_0

N training set size
n size of the dataset
ω_{feed} frequency caused by the feeding screw
ω_{mix} frequency caused by the mixing screw
p dimensionality or feature space
S_t data generating source at time t
$\mathbf{T}$ testing set
Tr_{out} outlier detection threshold
T a function mapping time indices to the actual time values
v_0 the speed of a mass change at time t_0
$\mathbf{X}_i$ the i^{th} instance
$\mathbf{X^H} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ historical data (instances)
$\mathbf{X_t^T} = (\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zt})$ training set at time t
$\bar{\mathbf{x}}^L$ a symmetric moving average with a lag L
$\bar{\mathbf{x}}^{'L}$ an asymmetric moving average with a lag L
$\mathbf{y}_i$ the label of $\mathbf{X}_i$

Acronyms

AI artificial intelligence
AIS artificial immune systems
ALL incremental training without instance selection (all data)
BIF training window algorithm [21]
CAPA context aware sales prediction approach
CBR case based reasoning
DBN dynamic Bayesan networks
GAM training window algorithm [74]
FISH training instance selection algorithm based on distances in time and space
KLI training window algorithm [111]
kNN the k-nearest neighbor classifier
LDC the Linear Discriminant classifier (Fisher classifier)
MAE mean absolute error
MASE Mean Absolute Scaled Error
NMC the Nearest Mean classifier (Euclidean classifier)
OMFP online mass flow prediction
PWC Parzen window classifier
RQ research question
SLIGRO the name of a food wholesales company
SVM support vector machine
TSY training set selection algorithm [199]
UKD ubiquitous knowledge discovery
VTT Technical Research Center of Finland
w.r.t. with respect to
WR window resizing algorithm

Chapter 1

Introduction

We live in a dynamic world, where changes are a part of everyday life. When there is a shift in data, the classification or prediction models need to be adaptive to the changes. In data mining the phenomenon of change in data over time is known as *concept drift*.

In this thesis we consider supervised learning under concept drift. In particular, we are interested in the training set formation strategies, which lead to achieving adaptivity to concept drift.

In this chapter we depict the research area, give motivating application examples and present the research outline and methodology. We define and narrow down a specific research problem and formulate the research questions. We finish the chapter by outlining and depicting the structure of the thesis.

1.1 Application Examples

Changes in underlying data might occur due to changing personal interests, changes in population, adversary activities or they can be attributed to a complex nature of the environment. Consider three motivating examples illustrating *a concept drift* phenomenon.

Example 1.1.1 (Fraud detection). The estimated loss of UK issued credit cards amounts to 610 million pounds in year 2008 [34]. These costs are born by the banking industry and indirectly by users via increased premium for card insurance. The financial institutions employ filters to data mine and monitor daily transactions.

The classification decisions need to be made online, and the behavior of both legitimate users and criminals is shifting. Moreover, the criminals might try using adversary tactics to overcome the fraud detection mechanisms. Therefore the filters used for monitoring need to be reactive to changing adversary behavior to keep the classification accuracy. Similar motivation applies to computer network security monitoring, e-mail spam filtering. □

Example 1.1.2 (News recommendation). Kate reads news on the internet every day. She uses a news recommender system, which provides a ranked list of headlines of potential interest to Kate. The recommender model is constantly updated using the records of her browsing history.

Kate has different interests. She likes Formula-1 sport, thus she reads the overviews of the races every second or third Monday, but not in winter when there are no races (long term interest). Recently she got an assignment at work to write a review on meat prices in New Zealand (short term temporal interest). She is also thinking if it is the right time to purchase a flat, thus her interest in real estate market situation has recently been increasing (gradual increase in interest).

The learning models in the news recommender system need to be adaptive over time to take into account short and long term interests, sudden and gradual changes. □

Example 1.1.3 (Navigation). The DARPA Grand Challenge is a prize competition for driverless cars [47].

> This event required teams to build an autonomous vehicle capable of driving in traffic, performing complex maneuvers such as merging, passing, parking and negotiating intersections. $< \ldots >$

> The autonomous vehicles have interacted with both manned and unmanned vehicle traffic in an urban environment. $< \ldots >$ Robots were also being judged on their ability to follow California driving rules.

The sensors in the vehicles are monitoring the road conditions and classifying them for the selection of a driving mode. The road is changing, the decisions need to be made in real time and it is not possible to account for every possible combination of road changes in advance. Therefore the winning entry in year 2005 'Stanley' [197] was equipped with an adaptive learner (adaptive Mixture of Gaussians).

Obviously, unmanned vehicles are not limited to competitions. They are irreplaceable in situations where it is dangerous (e.g. ecological accidents), infeasible (e.g. in space) to employ a human driver. □

These application examples give a motivation for the learners to be equipped with the concept drift adaptation mechanisms. The need for adaptation mechanisms in data mining are discussed in a number of position papers [55, 82, 83, 104, 118, 207] for about a decade. In the next section we take a closer look, what adaptation mechanisms mean.

1.2 Learning under Concept Drift

The goal of supervised learning is to learn a rule using training data in order to apply this rule to unseen testing data. Training data consists of pairs of objects: input vectors $\mathbf{X}$ and output labels $\mathbf{y}$. The task is to predict the output labels $\mathbf{y}'$, having input vectors of a testing data $\mathbf{X}'$, see Figure 1.1 (a). A rule $\mathcal{L}$, which was learned on the training data, is used for. The function $\mathcal{L}$ can output continuous valued outputs (regression task) or discrete labels (classification task). We will call *a learner* the strategy of classifier training (base classifier, parametrization and training set formation strategy). And when referring to a type of classifier (e.g. SVM, decision tree), we will use *base classifier* term. While referring to an individual classifier in an ensemble we will say *a model*.

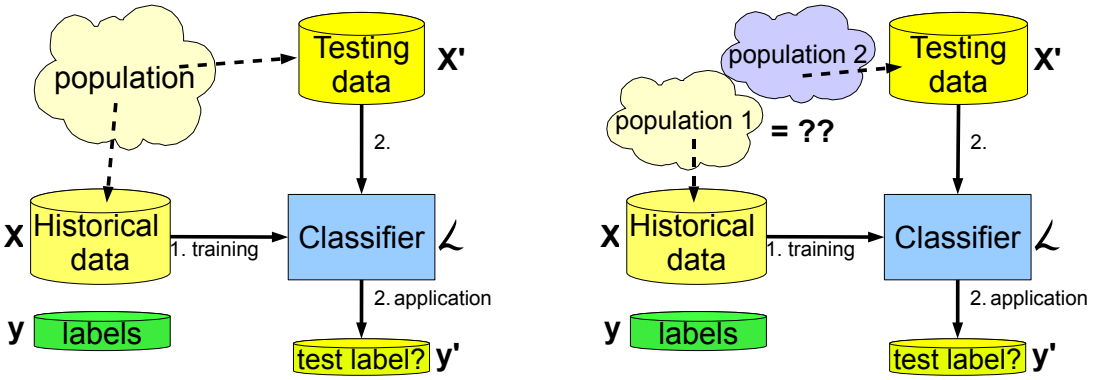


Figure 1.1: Supervised learning: (a) stationary, (b) under concept drift.

In Example 1.1.1 the labels are 'fraud' or 'not fraud', the input vectors are derived from financial transaction records. The goal is to learn to distinguish between 'fraud' and 'not fraud' given unseen transaction data. It is implicitly assumed that the training and the testing data come from the same distribution. This allows to make a generalization to unseen data, i.e. $\mathbf{y}' = \mathcal{L}(\mathbf{X}')$.

Definition 1.2.1. *A learner* is a mechanism assigning the particular form of function $\mathcal{L}$ and specific values based on the training data. □

By default it is assumed in supervised learning that the training and the testing data come from the same distribution. If distributions change over time, the question is if the same L is still applicable or how it should be adapted. In cases of changing environment over time the learners need to be able to handle *concept drift*, see Figure 1.1 (b).

Definition 1.2.2. *Concept drift* is an *unforeseen* change in underlying distribution of the objects of interest (input vectors and output labels). Unforeseen means that the change might be expected based on the domain knowledge but it is not known with certainty *when* or *if* it will occur. □

Learning under concept drift is associated with a sequential decision making. That means, training and testing data has a temporal order (time stamps). Thus the learners are associated with time as well.

Example 1.2.3. Consider a learner $\mathcal{L}$ for handwritten character recognition. While for handwritten character recognition of a child learning to write we might need to have a set of learners $(\mathcal{L}_1, \ldots, \mathcal{L}_t)$ a new one every month, because the handwriting is changing and the old model is no longer accurate[1]. □

What could be the strategies to make the learners to be adaptive. An intuitive approach is to discard an old model and train a new one using the new data. But there are several problems associated with this.

- The changes are unforeseen thus it is not known with certainty, when to discard and retrain.
- The changes might not be sudden but gradual, the contexts might reoccur, thus the exact point of change is not identifiable.
- The new data after the change is scarce. Thus there is a lag from an actual change to the time it can be detected. Moreover, the data after the change might not be enough to train the new learner accurately.

To deal with these problems a number of adaptivity mechanisms for the learners have been developed. The adaptivity can be incorporated into all parts of the learning process. (1) Base learners can be adaptive. For example, a decision tree can have its nodes included and deleted dynamically [93]. (2) Adaptation mechanism can be achieved by tuning the learning parameters. For example, a support vector machine can handle changing weighting of training instances [108]. (3) Adaptivity can be achieved by manipulating training data (instance selection). That is the subject of the thesis work. For example, instead of taking all training history, take a number of the latest instances (training window). Finally, (4) adaptivity can be achieved by the fusion rules of ensembles while maintaining diverse experts [193].

[1] Or an opportunity to make it more accurate is missed, as it will be explained in Chapter 3

1.3 The Thesis Focus and Scope

This thesis focuses on adaptive supervised learning techniques, where adaptivity to changes in data over time is achieved by selective training set formation. These training set formation methods typically can be used plugging in various base classifiers. In this work we consider accuracy (generalization error) as the primary performance measure for concept drift learners.

Our research follows the three main drift types, starting from sudden change, via gradual drift to reoccurring concepts. We give methodological contributions to concept drift phenomenon in data mining tasks as well as present four algorithms for training set formation under different application contexts and expected change types. By methodological contributions we mean training set formation strategies together with analytical reasoning, which if taken into account under given circumstances would contribute to the generalization performance. By algorithms we mean sets of instructions how to form a training set to train a classifier at a given time point given a sequence of historical data. For example, resizing training window based on particular theoretical estimates of the learner accuracy would be a methodological contribution. An algorithm would use the methodological contribution. It would provide precise instructions how to determine the exact window size given a sequence of historical data, how and in what order to estimate the parameters, how to estimate the accuracy.

1.3.1 The basic assumptions

The scope of the work is within the following basic assumptions.

We consider every new testing instance as a time step. An alternative would be to use batch learning, i.e. update a learner after a certain number of new instances are received. We see the latter as a special case of the former, it is mainly considered for computational reasons.

We assume that instances are observed one at a time. The label is not delayed, it becomes available after the classification or prediction decision is casted

and before the next instance is observed.

We retrain a classifier at every time step: $\mathcal{L}_t = f(\mathbf{X}_1, \ldots, \mathbf{X}_{t-1}, h_t)$, where h_t is the parameter vector. That is sequential learning approach. An alternative would be every time step to update the previous model using just the last historical instance: $\mathcal{L}_t = f^*(\mathcal{L}_{t-1}, \mathbf{X}_{t-1}, h_t^*)$. The former is a general case. The latter is specific to a base learner type and considered mostly due to computational reasons.

We assume equal costs of errors in classification.

1.3.2 Problem statement and research questions

In previous sections we overviewed the problem of learning under concept drift. We choose to focus on training set formation approach to achieve adaptivity to concept drift. We formulate the following problem statement:

How to advance adaptive supervised learning methods and develop training set formation algorithms based on these methods, in order to improve the accuracy of classification and prediction models under concept drift?
By improving the accuracy we mean the ability to identify the circumstances under which the proposed methos can outperform the naive strategies (no adaptivity) and selected methods from the recent scientific publications.

We decompose the problem statement into the following research questions:

Research question 1 (RQ1): What determines an optimal training window size under sudden concept drift? To what extent a change point is different from the start of the training window? How this difference can be used to improve the accuracy of an adaptive learner?

Research question 2 (RQ2): How can a training set selection method be developed, which would unify two selection criteria: similarity in time and feature space? What effect the integration of the two criteria has to the accuracy of an adaptive learner?

Research question 3 (RQ3): How can a contextual learning method be developed, which would relate training set formation to the context (the types of historical data 'behavior')? To what extent such method can increase the accuracy of food sales prediction, where recurring concepts are often expected?

Research question 4 (RQ4): How can the considered adaptive training methods be extended to develop a tailored training set selection method for a selected industrial application (industrial boiler)?

1.4 Research Methodology

We address the problem statement and the four research questions following theoretical and empirical research methodology. Each research question is handled in three steps: analyzing the related work, formulating and explaining the research concepts theoretically and empirically, and experimental evaluation. At first, we analyze relevant scientific publications focusing on the available methods closely related to the thesis problem statement and the research questions. After analyzing the existing methods, identifying the drawbacks or the lack of existing approaches, we formulate the strategies and solutions how to overcome the drawbacks. We verify the solutions we are proposing using prototype experiments for basic setups. Finally, we validate our proposed solutions experimentally using a range of relevant data and compare the results with naive approaches as well as existing closely related methods. RQ1 has the largest theoretical part, while RQ4 is application oriented (industrial case study).

We approach each of he four research questions following this methodology.

1. Reviewing the recent literature in a selected focus area. The focus areas correspond to the four research questions: determining training window size under sudden concept drift, instance selection criteria under gradual concept drift, training set construction under expected reoccurring contexts, change detection for sensor data.

2. Identifying the drawbacks or a lack of the existing adaptive learning methods with respect to a particular focus area.
3. Developing a reasoning for training set selection with respect to a particular focus area.
4. Looking for a theoretical justification corresponding to the developed reasoning.
5. Designing experimental scenarios and the experimental plan to verify the developed hypotheses. Finding suitable real problems with corresponding datasets, data preprocessing.
6. Implementing the developed and the peer methods, running the experiments and analyzing the testing results.

We evaluate the results in quantitative and qualitative manner. Quantitative evaluation is focused on generalization (accuracy) error. The generalization error is estimated by a testing error using sequential learning framework, which will be detailed in Section 2.1. Sequential learning does not require a hold out testing set, since testing is always done on the 'future' data in time. Qualitative evaluation aims to identify the competitive edges of the proposed methods, focusing on and analyzing in detail 'unusual' parts of the experiments.

The datasets used in Chapters 3, 4 are publicly available. The datasets used in Chapters 5, 6 are not publicly available due to commercial interests of the data providers, but a permission to use them might be requested. The datasets are described in detail in Appendix C.

1.5 The Main Contributions

The approach taken in this thesis to look at the training set formation strategies following different drift types is novel. There has been no systematic approach to *training set formation* for handling concept drift before.

The main contributions of the thesis to the data mining field are the following. The thesis improves the training strategies under sudden, gradual and recurring

concept drifts. Four adaptive training set formation algorithms (WR*, FISH, CAPA, OMFP) are developed and experimentally validated, which allow to increase the generalization performance of the base models under each of the three concept drift types, as compared to the selected existing adaptive learning algorithms and passive training strategies (no adaptivity).

The author of the thesis has published eleven scientific papers and two extended abstracts on the thesis topic, the list of the papers can be found in in Annex on page 227. In addition the extended versions of the works are now under review as journal papers and five related technical reports are available online.

1.5.1 Scientific novelty

The major aspects of scientific novelty of the thesis are as follows.

1. There was no explicit distinction between the change point and the start of the training window in the field. The historical data was dropped as soon as a sudden change was detected. We made an explicit theoretical distinction between the sudden change point and the training window. We demonstrated that the impact of taking the difference into account to the classification accuracy is increasing along with the more complex data. Based on the theoretical distinction we developed a training window resizing algorithm WR* and demonstrated an improvement in classification accuracy as compared to existing algorithms for variable window size, which do not make this distinction.

2. So far in the field either temporal instance selection (training windows) or instance selection in feature space was used. We developed a new distance measure unifying the distances in time and feature space for training set selection. We argued that both criteria are relevant under gradual concept drift and demonstrate this using real problem setups. Using the new distance measure we developed a training set selection algorithm (FISH) and demonstrated an improvement in classification accuracy as compared to existing algorithms using only time or only space criterion.

3. Using a real problem of food sales prediction, for which recurring concepts are relevant, we developed and experimentally validated a contextual method (CAPA) for training set formation. CAPA forms a training set interactively, based on the type of historical behavior, which is determined employing structural features. Training set formation based on structural features is novel in the field.

1.5.2 Practical significance

The methods developed in the thesis were tested using real data from different domains as well as two industrial problems were addressed: food sales prediction and mass flow estimation for an industrial boiler.

Using the food sales prediction problem CAPA method was developed. CAPA is not specific to food sales, it is applicable for prediction in other domains, for instance other sales, demand prediction, geographic crime maps, bus travel time prediction.

We developed a mass flow estimation method (OMFP) for an industrial boiler, which takes into account concept drifts. Such method is needed for monitoring and control system of the boiler operation. OMFP can be adapted to different tasks related to burning or fuel consumption processes, for example monitoring of the fuel consumption for passenger cars, based on traffic, fuel type.

The developed algorithms WR* and FISH are relevant for supervised learning tasks from different domains, where correspondingly sudden and gradual drifts are expected over time. Sudden drift is particularly relevant for network intrusion detection, financial fraud monitoring, navigation, demand prediction tasks. Gradual drift is particularly relevant for marketing, recommender systems, online shops, adaptive tutoring systems.

The thesis contributes to understanding concept drift problem in general and training set selection under concept drift in particular.

1.6 Statements Presented for the Defense

1. Theoretical distinction between the training window and the change point when determining a variable window size allows to improve generalization performance under sudden concept drift.

2. Integration of similarity in time and feature space when selecting training set allows to improve generalization performance as compared to using only time or only space criterion under gradual concept drift.

3. Contextual training set formation, while connecting the types of historical sales with the training set formation strategies and learning to recognize the types online using structural features, allows to improve generalization performance in food sales prediction task, where reoccurring concepts are expected.

4. The developed adaptive method for online estimation of the mass flow for an industrial boiler, which operates using a changing mix of fuel and changing input styles, allows to achieve more accurate estimates than using no adaptivity to changes and this way allows to improve the control system of the boiler.

1.7 Outline of the Thesis

The thesis is organized as follows. Chapter 2 defines a framework for learning under concept drift, discuss the design issues related to concept drift learners, overviews the literature and the context of the concept drift problem. Chapter 3 deals with sudden concept drift. We analyze how to determine the training window size online. In Chapter 4 we continue with gradual concept drift. We join time and space similarity for the training set selection. Chapter 5 is related to reoccurring contexts. We present context aware learning system. We focus on the problems of context definition and recognizing it online for food sales prediction problem. Finally, Chapter 6 presents an industrial boiler operation case study showing a presence of sudden and gradual drifts. We develop a method for concept drifting signal evaluation over time. In Chapter 7 we answer

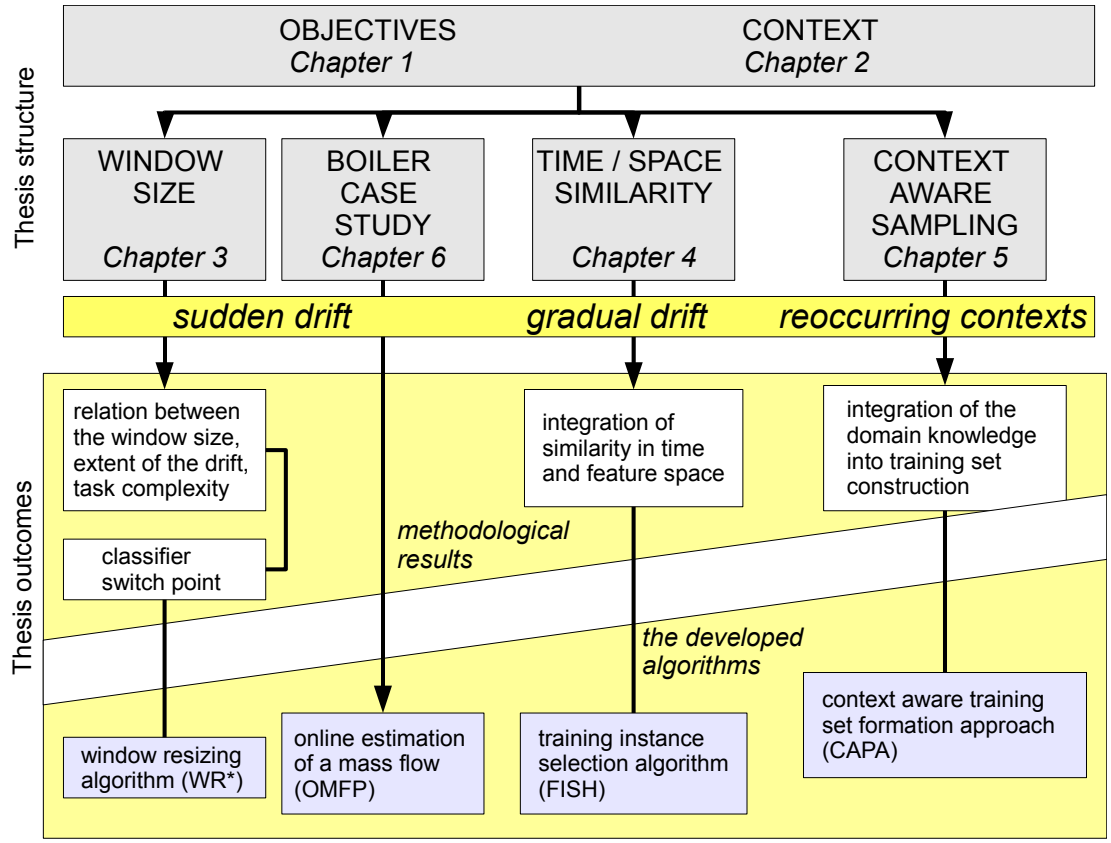


Figure 1.2: Thesis structure and outcomes.

the research questions and conclude. See Figure 1.2 for schematic representation of the thesis structure and outcomes.

Chapter 2

The Context of a Concept Drift Problem

In this chapter we present a context of concept drift problem. We focus on the issues relevant to adaptive training set formation. We present the framework and terminology, and formulate a taxonomy of concept drift learners design.

We start with formalizing the framework for the concept drifting data in Section 2.1, it will be used through all the thesis. In Section 2.2 we discuss the adaptivity mechanisms of the concept drift learners. In Section 2.3 we discuss the principle mechanisms of concept drift learners. In this chapter we give an overview of the available algorithms and categorize them based on their properties. Section 2.5 discusses the related research fields and Section 2.5 groups and presents major concept drift applications.

This chapter is to a large extent based on our overview published as a technical report [228].

This chapter is intended to give a bird's view of concept drift research field, provide a context of the research and position it within broad spectrum of research fields and applications. Chapters 3,4,5,6 will have additional related work sections. These reviews will focus on specific research problems, handled in those corresponding chapters.

2.1 Framework and Terminology

For analyzing the problem of training set formation under concept drift, we adopt *a progressive learning framework*, which is as follows.

A sequence of instances is observed, one instance at a time, not necessarily in equally spaced time intervals. Let $\mathbf{X}_t \in \Re^p$ is a vector in p-dimensional feature space observed at time t and $\mathbf{y}_t$ is the corresponding label. For classification $\mathbf{y}_t \in \mathcal{Z}^1$, for prediction $\mathbf{y}_t \in \Re^1$. We call $\mathbf{X}_t$ *an instance* and a pair $(\mathbf{X}_t, \mathbf{y}_t)$ *a labeled instance*. We refer to instances $\mathbf{X}^H = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ with $\mathbf{y}^H = (\mathbf{y}_1, \ldots, \mathbf{y}_t)$ as *historical data* and $\mathbf{X}_{t+1}$ as *target* (or testing) observation, for which the label is unknown.

2.1.1 Sequential learning with concept drift

We use *sequential learning* framework for testing experiments throughout the thesis. It is sometimes referred as *prequential testing* [75].

At every time step t we have historical data (labeled) available $\mathbf{X^H}$. A target instance $\mathbf{X}_{t+1}$ arrives. The task is to predict a label $\mathbf{y}_{t+1}$. For that we build a learner $\mathcal{L}_t$, using all or *a selection* from the available historical data $\mathbf{X^H}$. We apply the learner $\mathcal{L}_t$ to predict the label for $\mathbf{X}_{t+1}$. A prediction process at time step t is illustrated in Figure 2.1. That is for one time step.

At the next step after the classification or prediction decision is casted, the label $\mathbf{y}_{t+1}$ becomes available. How the instance $\mathbf{X}_{t+1}$ with a label is a part of historical data. The next testing instance $\mathbf{X}_{t+2}$ is observed. We picture a fragment of the sequential learning loop in Figure 2.2. The classifier training phase at time t is zoomed in. Training set formation strategies are the subjects of our investigation. They are depicted as a 'black box' in the figure.

Every instance $\mathbf{X}_t$ is generated by a source S_t. We delay more formal definition of a source until the next section, for now assume that it is a distribution over the data. If all the data is sampled from the same source, i.e. $S_1 = S_2 = \ldots = S_{t+1} = \mathbf{S}$ we say that the *concept* is stable. If for any two time points i and j $S_i \neq S_j$, we say that there is a *concept drift*.

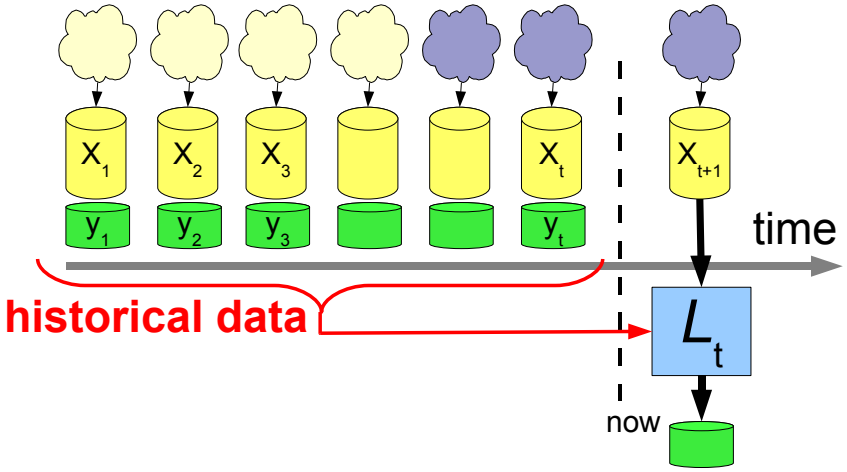


Figure 2.1: One time step (t) of the sequential learning process

Note that a random noise (deviation) is not considered to be a concept drift, because the data generating source is still the same.

The core assumption when dealing with the concept drift problem is uncertainty about the future. We assume that the source of the target instance $\mathbf{X}_{t+1}$ is not known with certainty. It can be assumed, estimated or predicted but *there is no certainty*. Otherwise the data can be decomposed into two separate data sets and learned as individual models or in a combined manner (then it is a multitask learning problem [17]).

We do not consider *periodic* seasonality as concept drift problem. But if seasonality is *not known* with certainty, we consider it as concept drift problem. For instance, a peak in sales of ice cream is associated with summer but it can start at different time every year depending on the temperature and other factors, therefore it is not known exactly when the peak will start.

2.1.2 Causes of a concept drift

Before looking what can actually cause the drift, let us return to the source S_t and provide more rigorous definition of it.

Classification problem independently of presence or absence of concept drift may be described as follows [152]. Let $\mathbf{X} \in \Re^p$ is an instance in p-dimensional

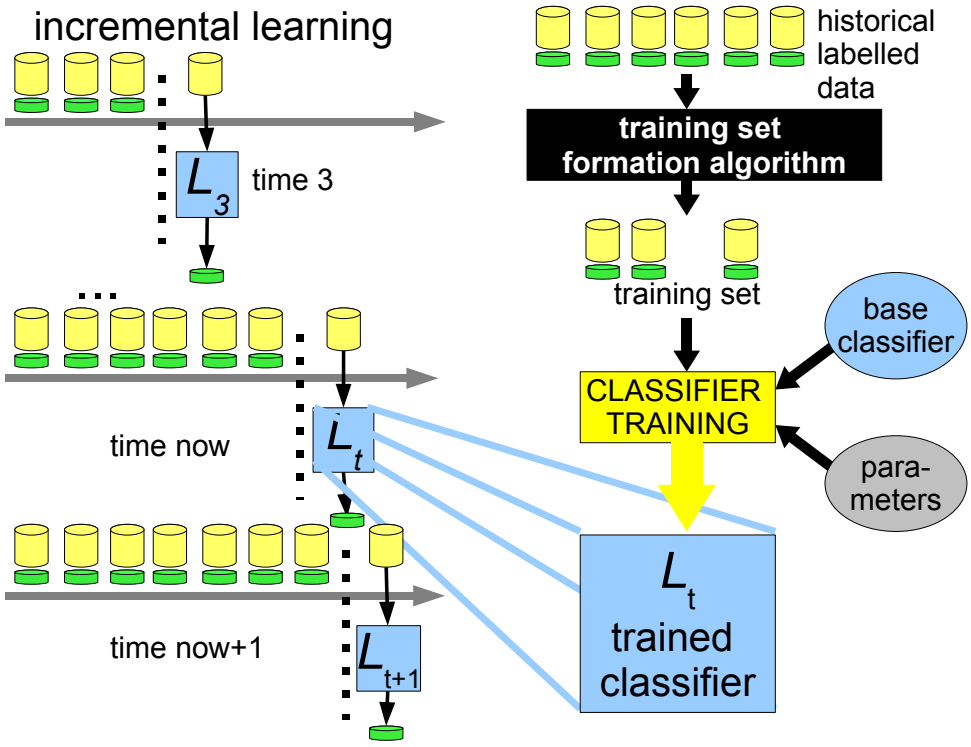


Figure 2.2: Sequential learning process

feature space. $\mathbf{X} \in c_i$, where $c_1, c_2, \ldots, c_k$ is the set of class labels. The optimal classifier to classify $X \rightarrow c_i$ is completely determined by a prior probabilities for the classes $P(c_i)$ and the class-conditional probability density functions (pdf) $p(\mathbf{X}|c_i)$, $i = 1, \ldots, k$.

Definition 2.1.1. We define a set of a prior probabilities of the classes and class-conditional pdf's as concept or *data source*:

$$\mathbf{S} = \{(P(c_1), p(\mathbf{X}|c_1)), (P(c_2), p(\mathbf{X}|c_2)), \ldots, (P(c_k), p(\mathbf{X}|c_k))\}. \qquad (2.1)$$

When referring to a particular source at time t we will use the term *source*, while when referring to a fixed set of prior probability and the classes and class-conditional pdf we will use the term *concept* and denote it $\mathbf{S}$. □

Recall, that in Bayesian decision theory [58] the classification decision for instance $\mathbf{X}$ at equal costs of mistake is made based on maximal a posteriori

probability, which for a class c_i is:

$$p(c_i|\mathbf{X}) = \frac{P(c_i)p(\mathbf{X}|c_i)}{p(\mathbf{X})}, \tag{2.2}$$

where p($\mathbf{X}$) is an evidence of $\mathbf{X}$, which is constant for all the classes c_i.

As first presented by Kelly et al [104], concept drift may occur in thee ways.

1. Class priors $P(c)$ might change over time.
2. The distributions of one or several classes $p(\mathbf{X}|c)$ might change.
3. The posterior distributions of the class memberships $p(c|\mathbf{X})$ might change.

Note, that the distributions $p(\mathbf{X}|c)$ might change in such a way that the class membership is not affected (e.g. symmetric movement to opposite directions).

Sometimes change in $p(\mathbf{X}|c)$ (independently whether it affects $p(c|\mathbf{X})$ or not) is referred as virtual drift and change in $p(c|\mathbf{X})$ is referred as real drift [210]. We argue, that from practical point of view it is not essential whether the drift is real or virtual, since $p(c|\mathbf{X})$ depends on $p(\mathbf{X}|c)$ as in Equation (2.2). In this thesis now on we do not make a distinction between the real and virtual drifts.

2.2 The Design of Concept Drift Learners

Following the framework, which was set-up in the previous section, the learner should provide the most accurate generalization for the data at time $t + 1$. In order to build such a learner, four main design sub-problems need to be solved.

A.1 Future assumption: a designer needs to make an assumption about the future data source S_{t+1}.

A.2 Change type: a designer needs to identify possible change patterns.

A.3 Learner adaptivity: based on the change type and the future assumption a designer chooses the mechanisms which make the learner adaptive.

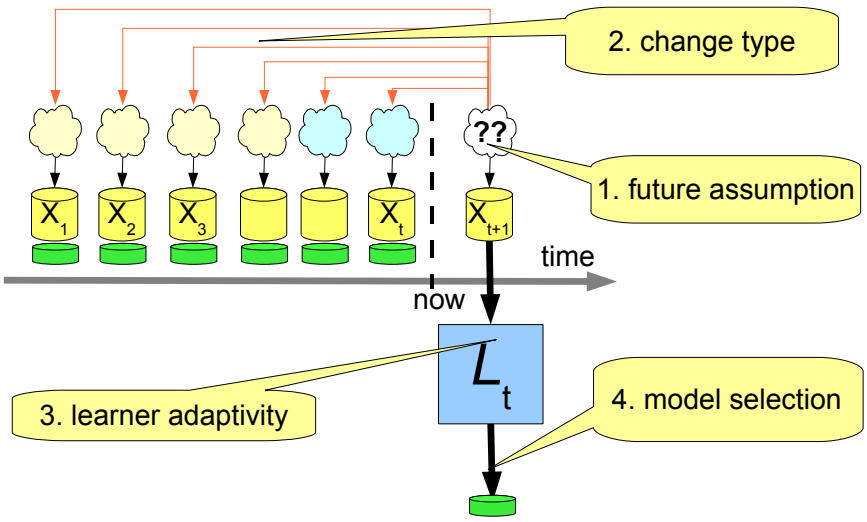


Figure 2.3: Sub-problems of the concept drift learner design.

A.4 Model selection: a designer needs a criterion to choose a particular parametrization of the selected learner at every time step (e.g. the weights for an ensemble members, the window size for variable window method).

All these sub-problems are the choices to be made when designing a learner. In Figure 2.3 we depict a positioning of each design sub-problem within the established learning framework.

In the next subsections we discuss each of the design sub-problems individually.

2.2.1 Future assumption

Future assumption is the assumption to be made about the source S_{t+1} of the target instance $\mathbf{X}_{t+1}$. We identify three types of choices here.

1. Assuming that $S_{t+1} = S_t$.
2. Estimating the source based on $\mathbf{X}_{t+1}$.
3. Predicting the change.

The first option, assuming $S_{t+1} = S_t$, is the most common among concept drift, although rarely explicitly stated. It is assumed that in the nearest future we will see the data coming from the same source as we saw in the latest past.

The second option utilizes information from the unlabeled target instance $\mathbf{X}_{t+1}$. Estimation of the source is usually done by measuring the distance between $\mathbf{X}_{t+1}$ and historical reference instances. The algorithms presented in [50, 133, 165, 199] use this future assumption.

Generally in concept drift problem the future data source is not known with certainty. However, there are methods using trainable prediction rules, to estimate the future state and incorporate that estimation into the sequential learning process. The algorithms using future predictions are presented in [30, 33, 214, 218].

2.2.2 Drift types

In this section we define *the change types*. By change types we mean the patterns the data sources take over time. The types of concept drift are defined based on those patterns. Note that in Section 2.1.2 we identified *the causes* of a drift, that list why the data generating sources might change.

For defining the drift types, let us *for now* restrict the number of possible data generating sources over time to two: S_I and S_{II}.

The simplest pattern of a change is *a sudden drift*. At time t_0 a source S_I is instantly replaced by source S_{II}. Recall an Example 1.1.2 from Chapter 1, where Kate was reading the news. Sudden interest in meat prices in New Zealand when she got an assignment to write an article, is a sudden drift.

Gradual drift is another type often met in the literature. However, there are two types of drift being mixed under this term. The first type of gradual drift is referring to a period when both sources S_I and S_{II} are active (e.g. [152, 192, 211]). As time passes, the probability of sampling from source S_I decreases, probability of sampling from source S_{II} increases. Note, that at the beginning of this gradual drift, before more instances are seen, an instance from the source S_{II} might be

easily mixed up with random noise. In Example 1.1.2 gradual drift is increasing interest in real estate, while Kate prefers real estate news more and more over time when her interest in buying a flat increases.

Another type of drift also referred as gradual includes more than two sources, however, the difference between the sources is very small, thus the drift is noticed only when looking at a longer time period (e.g. [11, 50, 116, 199]). In this thesis we refer to the former type of gradual drift as *gradual* and the latter type of drift as *incremental* (or stepwise). Incremental drift is a sequence of small sudden drifts.

The third type of drift is referred as *reoccurring concepts*. It happens when several data generating sources are expected to switch over time at irregular time intervals. Thus previously active concepts reappear after some time. This drift is not certainly periodic, it is not clear when the source might reappear, that is the main difference from seasonality concept used in statics. In Kate's example these are the biographies of Formula-1 drivers. The interest is related to the schedule of the races. But she does not look up the biographies at the time of the races, because she is watching them at the time. She might want to look up them later in the middle of the week. And the particular drivers she will be interested in might depend on who won the races this time.

In Figure 2.4 we give an illustration of the main drift types, assuming one dimensional data, where a source is characterized by the mean of the data. We depict only the data from one class.

Note that the types of drifts discussed here are not exhaustive. If we think of a data segment of length t and just two data generating sources S_I and S_{II}, the number of possible combinations of the sources (that means possible change patterns) would be 2^t, which is a lot. Moreover, in concept drift research it is often assumed that the data stream is endless, thus there could be infinite number of possible change patterns. We define the major structural types, since we argue, that assumption about the change types is absolutely needed for designing adaptivity strategies.

Recently there has been an attempt to categorize change types into mutually exclusive categories [148] based on number of recurrences, severity, speed and

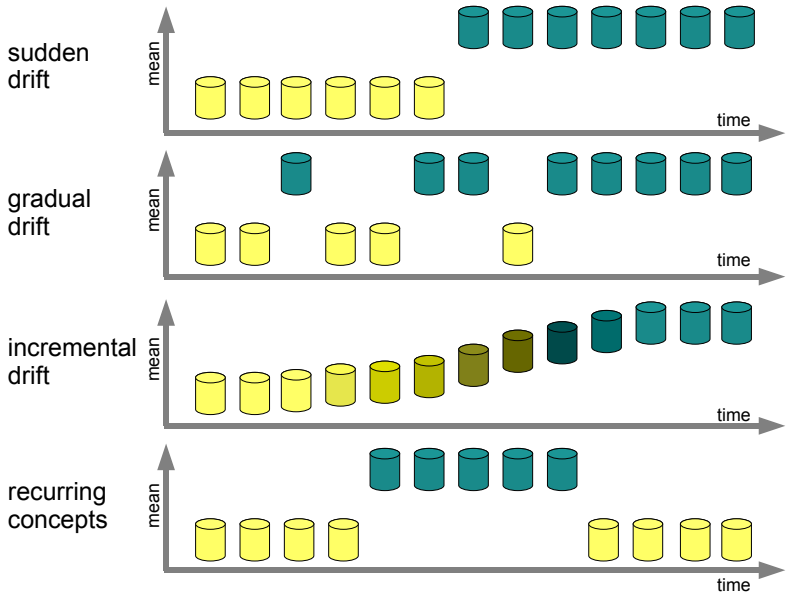


Figure 2.4: Illustration of the four structural types of the drift.

predictability. In principle the proposed categorization tries to quantify the main aspects of the learner design process into change categorization. We keep the design process separate. We argue that the categories cannot be mutually exclusive, because the change frequency count, speed, severity is relative to the length of the subsequence, at which one is looking. Thus we restrict our categorization to very few qualitative categories. We reserve predictability as a part of the learner design process (future assumption), not the change itself.

2.2.3 Learner adaptivity

We briefly discussed learner adaptivity in Chapter 1. We identify four main adaptivity areas:

1. Base learners can be adaptive (e.g. specific configuration of a decision tree nodes [93]).

2. Parametrization of the learners can be adaptively manipulated (e.g. weighting training samples in support vector machines [110]).

3. Adaptive training set formation (e.g. training windows, instance selection) can be employed, which is the scope and focus of this thesis. Training set formation can be decomposed into
   - training set selection,
   - training set manipulation (e.g. bootstrapping, noise),
   - feature set manipulation.
4. Fusion rules of the ensembles ([192, 193, 205]).

The adaptivity strategies, which are based on training set selection selection, can be generally divided into training window (selecting training instances consecutive in time) and instance selection (when instances that are close in the feature space are selected as a training set). The choice of adaptivity strategy strongly depends on the assumption about the change type, discussed in the previous section. For sudden drift training window strategies are preferred, while for gradual drift and reoccurring concepts instance selection strategies prevail. More detail discussion of strategies related to sudden drift will follow in Chapter 3, gradual drift in Chapter 4 and reoccurring concepts in Chapter 5.

2.2.4 Model selection

In the model training phase the procedure of estimating *the expected generalization error* for target instance $\mathbf{X}_{t+1}$ at every time step needs to be defined for model selection (parametrization)purposes. The two main options are:

1. theoretical evaluation of the generalization error, and
2. estimation of the generalization error using cross validation.

In any of the cases error estimation choice is strongly related to the future assumption, because it depends on the expectation regarding the future data source S_{t+1}.

The design process of a concept drift learner is graphically illustrated in Figure 2.5 (a). We see relations (1) and (2) as key issues in designing concept drift

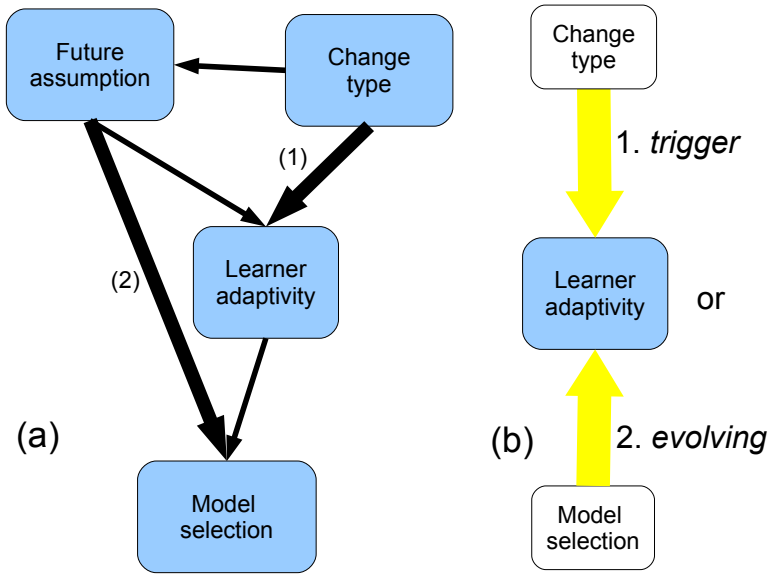


Figure 2.5: The design process of a concept drift learner.

learners. (1) the strategies selected to make the learners adaptive would strongly depend on the assumption about the change type, present in the data. (2) the model selection and evaluation strategies would strongly depend on the assumption about the future data source, on which the learner will be applied.

It is common to categorize concept drift learners into two major groups:

1. learner adaptivity is initiated by a *trigger* (or active change detector), and
2. a learner regularly *evolves* independently of the alarms or detectors.

The two categories can be positioned within the defined design framework. In the first group the initiation for learner adaptivity comes from the 'change type' block, while in the second group the adaptivity is based on 'model evaluation and selection' block. The process is illustrated in Figure 2.5 (b).

We will give more details about the categories of the concept drift learners in the next section, where we overview the related work.

2.3 Taxonomy of the Adaptive Learners

In this section we overview and map the related work. This section is intended to give a general view, the approaches specifically related to the methods developed in the thesis will be presented in the next chapters along with the contributions. The overview is concentrated on a supervised learning under concept drift.

Schlimmer and Granger [185] in 1986 formulated the problem of incremental learning from noisy data and presented an adaptive learning algorithm STAGGER. They are the authors of the term 'concept drift'. Since then a number of studies dealing with concept drift problem appeared. There were three 'peaks' in interest, one around 1998 followed by a special issue of Machine Learning journal [53], the other around 2004 followed by a special issue of Intelligent Data Analysis journal [120]. The third 'peak' started around 2007 and continues now on, as a result of increasing loads of streaming data and computational resources. Several PhD theses have directly addressed the problem of concept drift [20, 36, 153, 191, 212].

The learners responsive to a concept drift can be divided into two big groups based on *when* the adaptivity is 'switched on'. They are either trigger based or evolving. Trigger based means that there is a signal which indicates a need for model change. The trigger directly influences *how the new model should be constructed*. Most often change detectors are employed as triggers. The evolving methods on the contrary *do not maintain an explicit link between the data progress and model construction* and usually do not detect changes. They aim to build the most accurate classifier either by maintaining the ensemble weights or prototyping mechanisms. They usually keep a set of alternative models, and the models for a particular time point are selected based on their performance estimation. This is 'why' dimension in the taxonomy.

Another dimension for grouping concept drift learners is based on *how* the learners adapt. What are the actual adaptation mechanisms? The mechanisms were discussed following the design assumption A.4 presented in Section 2.2. Generally the adaptation mechanisms are either related to training set formation or a design and parametrization of the base learner.

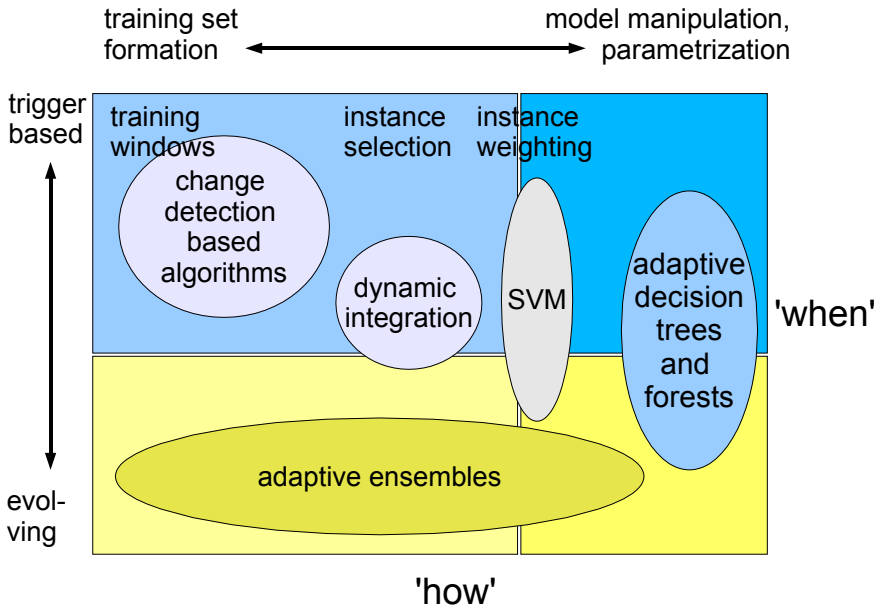


Figure 2.6: A taxonomy of adaptive supervised learning techniques.

Based on those two dimensions we overview the main methodological contributions available in the literature. The taxonomy is graphically presented in Figure 2.6. The positions of popular techniques (our interpretation) are indicated by ellipses.

2.3.1 Evolving learners

We start by overviewing the evolving techniques. Some of the techniques discussed above employ change detection mechanisms, still these are not the triggers of adaptation ('detect and cut'), but rather a tool to reduce computational complexity. First we discuss ensemble techniques, which make the largest group, and then other evolving techniques.

2.3.1.1 Adaptive ensembles

The most popular evolving technique for handling concept drift is *classifier ensemble*. Classification outputs of several models are combined or selected to

get the final output. The combination or selection rules are often referred as fusion rules.

There is a number of ensembles for concept drift, where the ideas are not specific to particular type of base learners (although some studies are limited to testing one base learner) [15, 60, 100, 112, 155, 178, 186, 192, 193, 199, 205, 206, 220]. There are also base learner specific ensembles. In those classifier combination rules usually depend on the base learner specific parameters of the learned models: [108, 110] with SVM, [214] with Gaussian mixture models, [172] with perceptrons, [131] with kNN.

In both cases *adaptivity* is achieved by fusion rules, i.e. by assigning the weights to the individual model outputs at each point in time. In a discrete case an output of a single model might be selected. In this case all except one models get zero weights. The weight indicates the 'competence' of a base learner, expected in the 'nearest future' (future assumption A.1). The weight is usually a function of the historical performance [15, 100, 112, 155, 173, 174, 178, 192, 193, 205, 220] in the past or estimated performance using selective cross validation [60, 131, 186, 199, 206] or base learner specific performance estimates [108, 110, 172, 214]. The historical evaluation is restricted to sudden and incremental drifts, while cross validation allow taking into account gradual drifts and reoccurring contexts.

In adaptive ensembles much attention is drawn to model evaluation and fusion rules (A.4), while little attention is drawn to the model construction (A.3). Still there is a number of options how to build diverse base classifiers. Usually the implicit aim is to have at least one classifier in the ensemble trained for each distinct concept. This can be achieved using different training set selection strategies.

The straightforward approach is to divide historical data into blocks, which include instances sequential in time. Often these blocks are non overlapping [15, 100, 108, 110, 131, 155, 193, 199, 205], sometimes overlapping [60]. These techniques are suitable for sudden and to some extent to incremental drifts, they favor reoccurring contexts. Another approach is using different sized training windows [112, 172, 186, 192], which implicitly assume that once off sudden drift has happened. Training windows are overlapping sequential blocks of instances, but all of them have fixed ending 'now' (time t). The individual models in

ensembles can also be constructed using non sequential instance selection [206]. This technique is more suitable to gradual drift, as well as reoccurring contexts.

Another approach to building diverse base classifiers is to use the same training data, but different types of base learners (e.g. SVM, decision tree, Naive Bayes) [178, 220].

All these techniques build individual models from what has already been seen in the past. In principle base classifiers can also be built adding unseen data, for instance noise or unlabeled testing data, which is listed as our future work.

2.3.1.2 Instance weighting

Instance weighting weighting methods make another group of evolving adaptation techniques. The algorithms can consist of a single learner [115, 157, 220] or an ensemble [22, 42, 77], but the adaptivity here is achieved not by combination rules, but by systematic training set formation. Ideas from boosting [69] are often employed, giving more attention to the instances which were misclassified.

2.3.1.3 Feature space

There are models, manipulating feature space to achieve adaptivity. [67] uses ideas from transfer learning to achieve adaptivity. New features are added to the training instances, which contain information from the past model performances. [24] augments the feature space by a time stamp. [101, 208] use dynamic feature space over time. In [6] the variables to observe next are adaptively selected.

2.3.1.4 Base model specific

There are also models to be mentioned, where adaptivity is achieved by managing specific model parameters or design. [156] maintain variable training window via adjusting internal structure of decision trees. Regression parameters are being adjusted in [104]. Past support vectors are transfered and combined

with the recent training data in [196]. The later examples illustrate the variety of possible specific model designs.

2.3.2 Learners with triggers

Another group of methods uses triggers, which determine how the models or sampling should be changed at a given time.

2.3.2.1 Change detectors

The most popular trigger technique is change detection, which is often implicitly related to a sudden drift. Change detection can be based on monitoring the raw data [21, 160], the parameters of the learners [194] or the outputs (error) of the learners [11, 74, 154]. [57] develop change detection methods in each of the three categories. The detection methods usually cut the training window at change point, although the change point and training window might not be the same [233].

2.3.2.2 Training windows

There are methods using heuristics for determining training window size [10, 134, 211, 218]. The heuristics is related to error monitoring. The training window is determined using look up table principles, where there is an action for each possible value of a trigger. There also are base learning specific methods, for determining training windows [93, 128, 198, 220]. The window size is also determined based on historical accuracy.

2.3.2.3 Adaptive sampling

The listed trigger based methods were using training windows. Another group of trigger based methods use instance selection. The incoming testing instances (unlabeled) are inspected. Based on the relation between the testing instance

and predefined prototypes [50, 103, 109, 219] or historical training instances directly[18, 88, 133, 165] a training set for a given instance is selected.

2.3.3 Discussion

In Table 2.1 we provide a summary of the listed algorithms. The properties are structured according to the four design assumptions, which were discussed in Section 2.2. The categorization is based on our interpretation of the methods.

Change detectors and ensembles are the two most popular techniques. Change detectors are naturally suitable for the data where sudden drift is expected. Ensembles, on the other hand, are more flexible in terms of change type, while they can be slower in reaction in case of a sudden drift.

We overviewed general methods for handling concept drift in supervised learning. A discussion of specific applications will follow in Section 2.5. Before proceeding to applications let us look at the broader context of learning with changing data.

2.4 Related Research Areas

After reviewing the adaptive techniques for supervised learning, which were mostly developed in data mining and machine learning communities, we now give an interdisciplinary perspective of the concept drift problem. In this section we point the 'neighboring' research fields. We pick the works, which are not necessary the 'key' references in these fields, but the ones which touch a problem of dataset change.

We present the research fields in three categories, which we identified as connections with the concept drift problem: time, knowledge transfer and adaptivity. In Figure 2.7 we position the related areas within these three categories. We discuss them in the following sections.

Table 2.1: Summary of concept drift responsive algorithms

Papers	Trigger	Design Assumptions			
		A.1 Future	**A.2 Change**	**A.3 Learner**	**A.4 Selection**
Without triggers					
[112, 192]				tr. windows	
[15, 100, 155, 193, 205]	ensemb.	last	sud./ inc.	time blocks	hist. err
[178, 220]				learner sp.	
[60, 131, 186]		last	sud./ inc.	tr. windows	
[199]	ensemb.	estim.	grad./ sud.	time blocks	cross v.
[206]		estim.	grad./ sud.	inst. select.	
[108, 110]		last	sud./ inc.	time blocks	
[172]	ensemb.	last	sud./ inc.	tr. windows	learner dep.
[214]		pred.	grad./ sud.	learner sp.	
[115, 157, 220]	inst. wght.	last	inc.	inst. wght.	hist. err
[22, 42, 77]					
[24, 67, 101, 208]	feature sp.	last	various	feature sp.	hist. err
[104, 196]		last	various	learner sp.	hist. err
[156]					learner dep.
With triggers					
[11, 21, 57, 74, 154, 160]	ch. detec.	last	sud.	tr. windows	hist. err
[10, 134, 211, 218]	windows	last	sud. /inc.	tr. windows	hist. err
[93, 128, 198, 220]				learner sp.	learner dep.
[50, 103, 109, 219]	inst. sel.	estim.	grad./ sud.	inst. sel.	prototyping
[18, 88, 133, 165]					cross v.

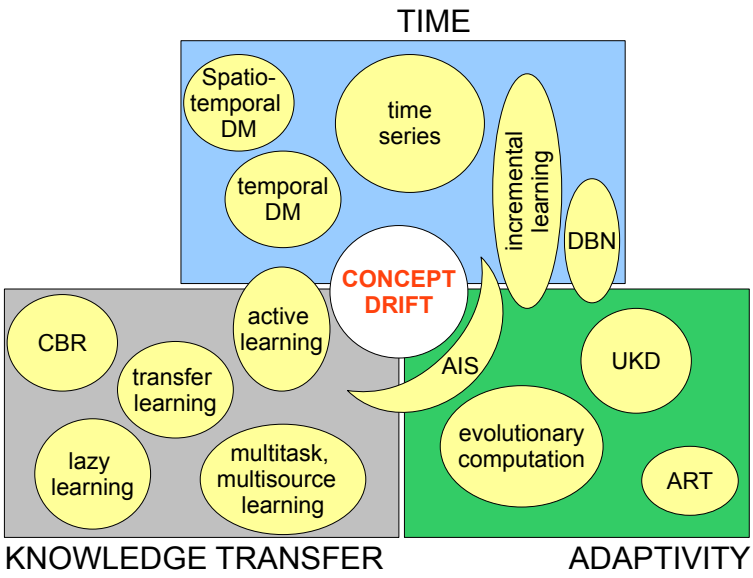


Figure 2.7: Categorization of the related areas. AIS - Artificial Immune Systems; DBN - Dynamic Bayesian Networks; UKD - Ubiquitous Knowledge Discovery; ART - Adaptive Resonance Theory; CBR - Case Based Reasoning.

2.4.1 Time context

Time context in concept drifting problems means that the data is sequential in time and the models are also associated with time and need to be continuously updated. There are research fields focusing on the aspects of model update primarily for a stationary data.

Incremental learning focuses on machine learning the tasks, where all the training data is not available at once [64, 80]. The data is received over time thus the models need to be updated or retrained, to increase the accuracy. Schlimmer and Granger [185] introduced the assumption of concept change in incremental learning context.

Over decades *incremental learning* area became less active. It was gradually overtaken by *data stream mining*, where the data flow is continuous an rapid [72]. Data stream mining focuses on the processing speed and complexity, thus naturally the attention toward timely *change detection* [139] including *anomaly detection* [37] has increased.

Spatio - temporal data mining deals with database models to accommodate tempo-

ral aspects [163, 177]. *Temporal data mining* [132] incorporates time dimension into data mining process.

Dynamic Bayesian networks are causal models assuming forward relation between the variables in time [41].

Finally, in *time series analysis* non stationarity is handled using ARIMA models [31].

2.4.2 Knowledge transfer

Knowledge transfer means that regularly there is a potential difference between the distribution of training data and the data to which the models will be applied (testing data). Thus the information from the old data needs to be adapted to fit to the new data. In concept drift problem this discrepancy arises in time, due to changes in the data generating process. However, a dataset shift can have a number of other reasons, for instance, sample selection bias [19], domain shift due to changes in measurements, model shift due to imbalance of data [167], discrimination in decision making [99], which are out of the scope of this thesis. In addition, the knowledge from related problem might be transfered to solve a related one.

Case based reasoning (CBR) [48] is the process of solving new problems based on the solutions of similar past problems. Generally CBR can be treated as *lazy learning*. Lazy learning does not build generalizing models, but maintain a database of reference data and uses the relevant past data only when a related query is made [3]. In this domain Aha [4] introduced noise tolerant instance based algorithms, IB3 was the first instance based technique capable of handling concept drift.

A great part of lazy learning research is devoted to *instance selection* methods to increase accuracy. There is another related instance selection research area (not necessarily lazy learning) aiming to reduce the learning complexity by data reduction [175].

In machine learning the process of applying the knowledge gained on solving a similar problem is referred as *transfer learning* [180] or inductive transfer. The ideas of inductive transfer were extended to temporal representation and used for learning under concept drift [67].

Adaptive knowledge transfer has been exploited in *multitask learning* [147] and *learning from multiple sources* [44, 142]. Non stationarity problem in machine learning community is sometimes called *covariate shift* [28, 97].

Finally, a field of *active learning* [187] is remotely related to the problem of concept drift. In active learning the data is labeled on demand, the methods select the instances which need to be labeled to make the learner more accurate or reduce labeling costs. The relation to concept drift problem is in the ways the methods identify, how well the unlabeled instances correspond to a particular concept.

2.4.3 Model adaptivity

Model adaptivity here means the models which have the properties of adaptation incorporated into learning. The adaptation might be to a change, as in concept drift problem. Adaptation can also mean the learning process (in stationary or non stationary environment), when the accuracy of the model is increasing along with more incoming examples.

Artificial immune systems (AIS) are inspired by immunology [68]. They are adaptive to changes like biological immune systems. AIS use evolutionary computation and memory to learn to recognize changing patterns.

Adaptive resonance theory, dating back 30 years [81], is based on the model of information processing by the brain [35]. Having self-adjusting memory as one of the desired system properties.

In *evolutionary computation* dynamic optimization problems are actively studied [150]. The goal is track the optima which is dynamically changing in time. The major approaches are related to maintaining and enhancing diversity, expecting that once the optima changes, there are suitable models available within the

pool [217]. A a next step in this direction is to seek for a relation between the past models and current task [176]. In [179] a relation between the change type and magnitude and the evolutionary algorithm is introduced.

Ubiquitous knowledge discovery is an emerging area, which focuses on learning in distributed and mobile systems [145]. The systems work in environment, they need to be intelligent and adaptive. The objects of UKD systems exist in time and space in a dynamically changing environment, they can change location and might appear or disappear. The objects have information processing capabilities, know only their local spatio-temporal environment, act under real-time constraints and are able to exchange information with other objects. These objects are humans, animals, and, increasingly, computing devices.

To sum, the problem of change is far not limited to data mining and machine learning community. Concept drift problem lies in all three dimensions: time dimension, need for adaptivity and knowledge transfer.

2.5 Applications

In this section we survey applications, where concept drift problem is relevant in supervised (and unsupervised) settings. We present the real life problem, discuss the sources of a drift and the actual learning tasks in the context of these problems.

We find four generic types of applications: monitoring control, personal assistance, decision support and artificial intelligence. *Monitoring and control* often employs unsupervised learning, which detects abnormal behavior. It includes detection of adversary activities on the web, computer networks, telecommunications, financial transactions. *Personal assistance and information* applications include recommender systems, categorization and organization of textual information, customer profiling for marketing. *Decision support* includes diagnostics, evaluation of creditworthiness. The 'ground truth' is usually delayed, i.e. the true answer whether the decision was correct becomes available only after certain time. *Artificial intelligence* applications include a wide spectrum of moving

Table 2.2: Types of applications with concept drift.

	Decision speed	Accuracy	Costs of mistake	Labels	Adversary
1. Monitoring & **control**	high	approximate	medium	hard	active
2. Assistance & **information**	medium	approximate	low	soft	low
3. Decision support	low	precise	high	delayed	possible
4. AI and robotics	high	precise	high	hard	low

and stationary systems, which interact with changing environment, for instance robots, mobile vehicles, smart household appliances.

We define five dimensions, relevant to the applications facing concept drift:

1. the speed of learning and output,
2. classification or prediction accuracy,
3. costs of mistakes,
4. true labels,
5. adversary activities.

The speed of learning output means what is a relative volume of data and how fast the decision needs to be made. For example, in credit card fraud detection the decision needs to be fast to stop the crime and the data loads are huge, while in credit evaluation a decision regarding the credit can be made even in a few days time. In both cases adversary activities to cheat the system might be expected, while adversary activities in diagnostics would make less sense. The precise accuracy in diagnostics is generally much more significant than in movie recommendations, moreover, in movie recommendations the decision support be 'soft' in a sense the viewer is not always deterministic, which movie he or she liked more.

Our global interpretation of the four types of applications in accord with these dimensions is provided in Table 2.2.

In the following sections we discuss each of the application types separately and give arguments for the choices we made in the table.

2.5.1 Monitoring and control

In monitoring and control applications the data volumes are large and it needs to be processed in real time. Two types of tasks can be distinguished: prevention and protection against adversary actions, and monitoring for management purposes.

2.5.1.1 Monitoring against adversary actions

Monitoring against adversary actions is often an unsupervised learning task or one class classification, where the properties of 'normal behavior' are well defined, while the properties of attacks can differ and change from case to case. Classes are typically highly imbalanced with a few real attacks.

Computer security. *Intrusion detection* is one of the typical monitoring problems. That is a detection of unwanted access to computer systems mainly through network (e.g. internet). There are passive intrusion detection systems, which only detect and alert the owner, and active systems, which take protective action. In both cases here we refer only to a detection part.

Adversary actions is the primary source of concept drift in intrusion detection. The attackers try to invent new ways how to attack, which would overcome the existing security. The secondary source of concept drift is technological progress in time, when more advanced and powerful machines are created, they become accessible to intruders. 'Normal' behavior can also change over time.

Lane and Brodley [127] explicitly formulated the problem of concept drift in intrusion detection a decade ago. They presented a detection system using instance based learning. Current research directions and problems in intrusion detection can be found in a general review [159]. From supervised learning,

lately, ensemble techniques have been proposed [144]. Artificial immune systems are widely considered for intrusion detection[106].

Telecommunications. Adversary behavior also applies to *telecommunications industry*, both intrusion and fraud. Mobile masquerade detection problem [146] from research perspective is closely related to intrusion detection. The goal is to prevent adversaries from unauthorized access to a private data. The sources of concept drift are again twofold: adversary behavior trying to overcome the control as well as changing behavior of legitimate users. Fraud detection and prevention in telecommunication industries [89] is also subject to concept drift due to similar reasons.

Finance. In *financial sector* data mining techniques are employed to monitor streams of financial transactions (credit cards, internet banking) to alert for possible frauds. A case was discussed in Example 1.1.1. Insider trading in stock market is one more application.

Both supervised and unsupervised learning techniques are used [29] for detection of fraudulent transactions. The data labeling might be imprecise due to unnoticed frauds, legitimate transactions might be misinterpreted and the imbalance of the classes is very high (few frauds as compared to legitimate actions). Concept drift in user behavior is one of the challenges.

Insider trading is trading in stock market based on non-public information about the company, in most countries it is prohibited by law. Inside information can come in many forms: knowledge of a corporate takeover, a terrorist attack, unexpectedly poor earnings, the FDA's acceptance of a new drug [56], inside trading disadvantages regular investors. There is a potential for concept drift, since the inside traders would try to come up with novel ways to distribute the transactions in order to hide.

2.5.1.2 Monitoring for management

Monitoring for management usually uses streaming data from sensors. It is also characterized by high volumes of data and real time decision support; however, adversary cases usually are not present.

Transportation. *Traffic management* systems use data mining to determine traffic states [45], e.g. car density in a particular area, accidents. Traffic control centers are the end users of such systems. Transportation systems are dynamic (always moving).The traffic patterns are changing seasonally as well as permanently, thus the systems have to be able to handle concept drift.

Data mining can also be employed for prediction of public transportation travel time [149], which is relevant for scheduling and planning. The task is also subject to concept drift due to traffic patterns, human driver factors, irregular seasonality.

Positioning. Concept drift is also relevant in remote sensing in *fixed geographic locations*. Interactive road tracking is an image understanding system to assist a cartographer annotating road segments in aerial photographs [221]. In this problem change detection comes into play when generalizing to different roads over time. In place recognition [140] or activity recognition [136] dynamics of the environment cause concept drift in the learned models.

Climate patterns, such as floods, are expected to be stationary, but the detection systems have to incorporate not regular reoccurring contexts. In a light of a climate change the systems might benefit from adaptive techniques, for instance, sliding window training [119]. In [117] the authors use active learning of non stationary Gaussian process for river monitoring.

Industrial monitoring. In *production monitoring* human factor can be the source of concept drift. Consider a boiler used for heat production. The fuel feeding and burning stages might depend on individual habits of a boiler operator, when the fuel is manually input into the system [12]. The control task is to identify

the start and end of the fuel feeding, thus algorithms should be equipped with mechanisms to handle concept drift.

In *service monitoring* changing behavior of the users can be the source of a drift. For example, data mining is used to detect accidents or defects in telecommunication network [161]. A change in call volumes may be the results of an increased number of people trying to call friends or family to tell them what is happening or a decrease in network usage caused by people being unable to use the network. Or the change might be unrelated to the telecommunication network at all. The fault detection techniques have to be able to handle such anomalies.

2.5.2 Personal assistance and information

These applications mainly organize and/or personalize the flow of information. The applications can be categorized into individual assistance for personal use, customer profiling for business (marketing) and public or specified information. In any case, the class labels are mostly 'soft' and the costs of mistake are relatively low. For example, if a movie recommendation is wrong it's not a world disaster and even the user himself or herself might not know for sure, which of the two given movies he or she likes more.

2.5.2.1 Personal assistance

Personal assistance applications deal with user modeling aiming to personalize the flow of information, which is referred as *information filtering*. A rich technical presentation on user modeling can be found in [78]. One of the primary applications of user modeling is representation of queries, news, blog entries with respect to current user interests. Changes in user interests over time are the main cause of concept drift.

Large part of personal assistance applications are related to *textual data*. The problem of concept drift has been addressed in news story classification [23, 213] or document categorization [111, 135, 151]. [102] in a light of changing user

interests address the issue of reoccurring contexts. Recall an example 1.1.2 about Kate reading the news. Drifting user interests are relevant in building personal assistance in digital libraries [87] or networked media organizer [65].

There is also a large body of research addressing web personalization and dynamics [32, 46, 184, 215], which is again subject to drifting user interests. In contrast to end user text mining discussed before, here mostly interim system data (logs) is mined.

Finally, concept drift problem is highly relevant for spam filtering [51, 62]. First of all there are adversary actions (spamming) in contrast to the personal assistance applications listed before. That means the senders are actively trying to overcome the filters therefore the content changes rapidly. Adversaries are intelligent and adaptive. Spam types are subject to seasonality and popularity of the topics or merchandize. There is a drift in the amount of spam over time, as well as in the content of the classes [61]. Spam messages are disjunctive in content. Besides, personal interpretation of what is spam might differ and change.

2.5.2.2 Customer profiling

For customer profiling aggregated data from many users is mined. The goal is to segment the customers according to their interests. Since individual interests are changing over time, customer profiling algorithms should take this non stationarity into account.

Direct marketing is one of the applications. Adaptive data mining methods are used in customer segmentation based on product (cars) preferences [45] or service use (telecommunications) [24]. Lately in addition to similarity measures between individual customers social network analysis has been employed into customer segmentation [129]. It is observed that user interests do not evolve simultaneously. The users that used to have similar interests in the past might no longer share the interests in the future. The authors model this as an evolving graph. Adaptivity is also relevant to association rule mining applied to shopping basket identification and analysis [181].

Automatic recommendations can be related to both customer profiling and personal assistance. The recommender systems are characterized by sparsity of data. For example, there are only a few movie ratings per user, while the recommendations need to be inferred over the while movie pool. The publicity of recommender systems research has increased rapidly with a NetFlix movie recommendation competition. The winners used temporal aspect as one of the keys to the problem [16, 113]. Three sources of drift were noted movie biases (popularity changes over time), user bias (natural drift of users' rating scale benchmarking to the recent ratings) and changes in user preferences. There are earlier works on recommender systems in which changes over time were addressed [54] via time weighting.

2.5.2.3 Information

Information applications are related to changes in data distribution over time, which is sometimes referred as virtual drift in concept drift literature [210]. Then changes in class assignment is called real drift. Virtual drift would typically occur over longer period of time. For example, in news recommendation system presented in Example 1.1.2, the news about meat prices in New Zealand suddenly become relevant for Kate (the label changes, but the document comes from the same distribution as before). It might happen that the consumers in New Zealand would switch from pork to beef, thus the distribution of articles about meat would change independently from Kate's interests.

Document organization is the first category of information applications. Given e-mail, news or document streams, the task is to extract meaningful structures, organize the data into topics. Temporal order is necessary for making sense. The topics themselves and even the vocabulary for particular topics change in time.

The state of the art Latent Dirichlet Allocation model for probabilistic document corpus modeling was recently equipped with a time dimension [27, 204]. In [27] the dynamics of scientific topics articles of Science magazine from 1881 to 1999 (120 years) was analyzed, the emergence, peak and decline of topics was showed, the topic vocabulary representation was build. [218] incorporated the

time stamp into the static model. [107] presented a method for organization of e-mail messages, to provide a framework for content analysis. Intuitively this is similar to including time feature into the original observation.

Economics. Concept drift is relevant in making macroeconomic forecasts [79], predicting the phases of a business cycle [109]. The data is drifting primary due to large number of influencing factors, which are not feasible to be taken into prediction models. Due to the same reason financial time series are known to be non stationary to predict [85].

In **business management**, in particular, software project management, careful planning can be inaccurate if concept drift is not taken into account. [59] employ data mining models for project time prediction, the models are equipped with concept drift handling techniques.

2.5.3 Decision support

Decision support and diagnostics applications usually involve limited amount of data (might be sequential or time stamped). Decisions are not required to be made in real time, thus the applied models might be computationally expensive. But high accuracy is essential in these applications and the costs of mistakes are large.

Finance. *Bankruptcy prediction* or individual credit scoring is typically considered to be a stationary problem [122]. However, in these problems concept drift is closely related to a hidden context [86], changes in context, which is not observed or measured in the original model. The need for different models for bankruptcy prediction under different economic conditions was acknowledged and proposed in [195]. The need for models to be able to deal with non stationarity has been rarely acknowledged [91]. Although concept drift problem is present, adversaries might make use of full adaptivity of the models. Thus offline adaptivity, which would be restricted to already seen subtypes of customers, is needed [231].

Biomedical applications can be subject to concept drift due to adaptive nature of microorganisms [190, 199]. The effect of antibiotics to a patient is often naturally diminishing over time, since microorganisms mutate and evolutionary develop antibiotic resistance. If a patient is treated with antibiotic when it is not necessary, a resistance might develop and antibiotics might no longer help when they are really needed.

Clinical studies and systems need adaptivity mechanisms to changes caused by human demographics [73, 121]. The changes in disease progression can also be triggered by changes in a drug being used [25]. In incremental drug discovery experiments the drift between training and testing sets can caused by non uniform sampling [66].

Data mining can be used to discover emerging resistance and monitor nonsomnical infections in hospitals (the infections which result from the treatment) [96]. Given patient and microbiology data as an input, the task is to model the resistance. The resistance changes over time.

Finally, concept drift occurs in biometric authentication [164, 216]. The drift can be caused by changing physiological factors, for example growing beard. Like in credit applications, here adaptivity of the algorithms should be used with caution, due to potential adversary behavior.

2.5.4 AI and robotics

In AI applications the problem of concept drift is often called dynamic environment. The objects learn how to interact with the environment and since the environment is changing, the learners need to be adaptive.

2.5.4.1 Mobile systems and robotics

Ubiquitous Knowledge Discovery (UKD) deals with the distributed and mobile systems, operating in a complex, dynamic and unstable environment. The word 'ubiquitous' means distributed at a time. Navigation systems, vehicle

monitoring, household management systems, music mining are examples of UKD.

DARPA navigation challenge was presented in Example 1.1.3. A winning entry in 2005 used online learning for road image classification into drivable and not drivable [197]. They used an adaptive Mixture of Gaussians, for gradual adaptation they were adjusting the internal Gaussian and rapid adaptation by replacement of the Gaussians with the new ones. The needed speed of adaptation would depend on the road conditions.

Adaptivity to changing environment has been addressed in robotics [166], for instance in designing a player for robot soccer [130].

2.5.4.2 Intelligent systems

'Smart' home systems [169] or intelligent household appliances [7] also need to be adaptive to changing environment and user needs.

2.5.4.3 Virtual reality

Finally, virtual reality needs mechanisms to take concept drift into account. In computer game design [38] adversary actions of the players (cheating) might be one of the drift sources. In flight simulation the strategies and skills differ across different users [86].

In Table 2.3 we summarize the discussed applications with concept drift.

2.6 Terminology

Concept drift is relatively new research field and the terminology is not yet fixed. Moreover, the problem of shifting data is discovered and handled in very broad domain area. With the loads of data more and more attention is drawn to the differences between training and testing data distributions. We

Table 2.3: Summary of applications with concept drift

CATEGORIES		APPLICATIONS	REFERENCES
		Monitoring and Control	
against adversaries	computer security	intrusion detection	[106, 127, 144]
	telecommunications	intrusion detection, fraud	[89, 146]
	finance	fraud, insider trading	[29, 56]
for management	transportation	traffic management	[45, 149]
	positioning	place, activity recognition	[136, 140, 221]
	industrial mon.	boiler control, telecom mon.	[12, 161]
		Assistance and Information	
personal assistance	textual information	news, document classification	[23, 111, 135, 151, 213]
		spam categorization	[51, 62]
	web	web personalization	[32, 46, 184, 215]
		libraries, media	[65, 87]
customer profiling	marketing	customer segmentation	[24, 45, 129, 181]
	recommender systems	movie recommendations	[16, 54, 113]
information	document organization	articles, mail	[27, 107, 204, 218]
	economics	macroeconomics, forecasting	[79, 85, 109]
	project management	software project mgmt.	[59]
		Decision Support	
finance	creditworthiness	bankruptcy prediction	[91, 195, 231]
biomedicine	drug research	antibiotic res., drug disc.	[66, 96, 199]
	clinical research	disease monitoring	[25, 73, 121]
security	authentication	biometrics	[164, 216]
		AI and Robotics	
	mobile systems	robots, vehicles	[130, 166, 197]
	intelligent systems	'smart' home, appliances	[7, 169]
	virtual reality	computer games, flight sim.	[38, 86]

provide alternative terminology in Table 2.4. Lithuanian terminology related to the problem of concept drift is presented in Annex *Vocabulary*, page 231.

Table 2.4: Terminology across research fields.

AI and Robotics	dynamic environment
Databases	concept drift, load shedding
Data mining	concept drift
Evolutionary computation	changing environment
Information retrieval	temporal evolution
Machine learning	concept drift, covariate shift
Statistics, time series	non stationarity

2.7 Concluding Remarks

We started the chapter by defining the framework for supervised learning under concept drift. We then discussed the design elements, which are present in concept drift learners. Next we provided a systematic overview of the available concept drift responsive techniques. We categorized the methods based on the framework and the design assumptions. We took even broader view of the available work, we listed the related areas and discussed the connection with the concept drift problem. Finally, we presented an overview of the real problems, where concept drift is present, and discussed what techniques are employed there.

The research area related to the problem of concept drift is broad. In addition, there are a lot of specific problems which are only related by the fact that there is a non stationarity in the process. In this chapter we presented the context of the thesis work and positioned the thesis within this context.

Chapter 3

Sudden Drift: Training Window Length

In this chapter we focus on training windows. That is a strategy of selecting sequential instances in time as the training set. The main question for training windows is how to determine the optimal window length at a given time. We start by analyzing the dynamics of training window length theoretically. Then we make a theoretical distinction between the change point and classifier switch point. Finally we present a plug-and-play algorithm for determining fixed and variable training window size.

This chapter is based on our publications [124, 125, 224, 233] and includes a material from our papers [225, 232] and our publications [222, 223].

The chapter is organized as follows. We start by setting up the theoretical framework for training window selection. In Section 3.2 we analyze the relation between the classification task complexity, extent of the drift and training window size. We investigate two cases: a sudden drift and an incremental (stepwise) drift. In Section 3.3 we discuss the related work. In Section 3.4 we make an explicit distinction between the drift point and the classifier switch point and analyze relation between the two. In Section 3.5 we propose an algorithm for determining variable training window size online, based on estimating the classifier switch point theoretically. Section 3.6 gives experimental results. Section 3.7 concludes.

3.1 Motivation

The adaptation of the classifier can be controlled by the size of a moving *training window*.

Definition 3.1.1. Training window at time t is a training set containing the latest N historical instances $(\mathbf{X}_{t-N+1}, \ldots, \mathbf{X}_t)$ with labels $(\mathbf{y}_{t-N+1}, \ldots, \mathbf{y}_t)$. N is the window length. □

Larger training windows are preferred for stationary distributions, while shorter windows are preferred after a sudden concept change. If the distribution is stationary, the training window should be allowed to expand up to a sufficient pre-defined size.

Assume a sudden concept drift, when the data sources change suddenly at some time. Such drift was illustrated in Example 1.1.2. When Kate is given a task to write a report on meat prices, she suddenly becomes interested in this topic. Following the sequential learning framework (defined in Section 2.1) classification model is retrained at every time step after receiving new information. Thus at every time step a window length needs to be determined.

Known methods for choosing the window size rely mostly on heuristics. More importantly, they do not make a clear-cut difference between the window size for *detecting* the change and the window size for *training* the classifier. The two are differen, because even if the change point is known, the new data is scarce after the drift. Thus a portion of old data is still useful to be included into a training window. We start with showing analytically how the optimal window size depends on the data complexity (quantified by separability and dimensionality) and the extent of the drift.

3.2 Dynamics of Training Window Size with NMC

In order to analyze the relation between the window size, extent of the drift and data complexity we use the expression for theoretical generalization error.

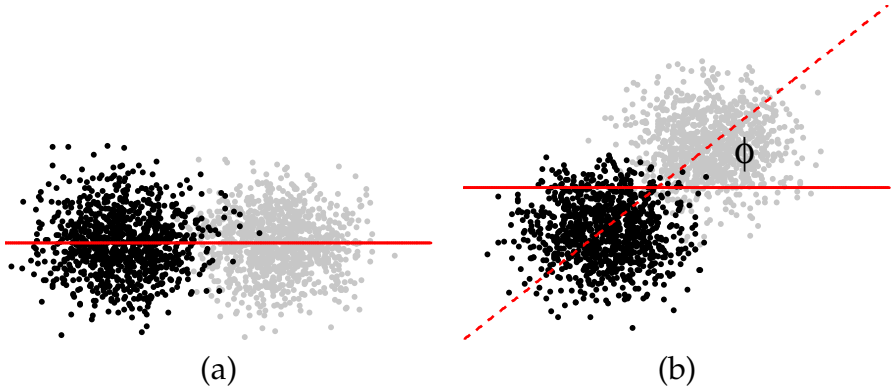


Figure 3.1: The two classes of data (a) before the drift, and (b) after the drift.

The goal is to minimize the error at every time step by varying the training window size. We employ a fixed setup of drift and use NMC as a base classifier to calculate analytical expression for the generalization error.

3.2.1 Setup and basic assumptions

Assume that a single sudden concept drift happens at time $t+1$. Let the data before the drift are distributed as $\mathbf{X} \sim \mathcal{N}(\mu_i^{(1)}, I)$, $\mathbf{X} \in \Re^p$, $i = 1, 2$ are the class indices. The data after the drift are generated by a different source and are distributed as $\mathbf{X} \sim \mathcal{N}(\mu_i^{(2)}, I)$.

Let's assume a drift occurs as data rotation. The data is rotated counterclockwise by angle φ, with respect to the point $[0, 0]$, see 3.1. A rotation matrix $\mathbf{H}_{p\times p}^{rot}$ is used:

$$\mathbf{H}_{p\times p}^{rot} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & \dots & 0 \\ -\sin\varphi & \cos\varphi & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Without loss of generality, let the means of the two classes before the drift are $\mu_1^{(1)} = (\frac{\delta}{2}, 0, \dots, 0)^{\mathbb{T}}; \mu_2^{(1)} = (-\frac{\delta}{2}, 0, \dots, 0)^{\mathbb{T}}; \mu_1^{(1)} = -\mu_2^{(1)}$. Then the means after the drift are respectively $\mu_1^{(2)} = \mathbf{H}\mu_1^{(1)} = (\frac{\delta}{2}\cos\varphi, \frac{\delta}{2}\sin\varphi, 0, \dots, 0)^{\mathbb{T}}$ and $\mu_2^{(2)} = \mathbf{H}\mu_2^{(1)} = -\mu_1^{(2)}$.

Let β is the proportion of the new data (after the drift) in the training sample. Then the class means of the mixture are $\mu_2^{(M)} = (1-\beta)\mu_1^{(1)} + \beta\mu_1^{(2)}$ and $\mu_2^{(2)} = -\mu_1^{(2)}$.

We investigate the accuracy of NMC on the defined data model with a single drift (sudden) and a sequence of drifts (incremental drift).

3.2.2 Accuracy of the mixed classifier after a sudden concept drift

A mixed classifier is the classifier trained on a mixture of old and new data after the drift. *We derive* the expression for generalization error of NMC trained on such data in Appendix A. In the limit when $N \to \infty, p \to \infty, \frac{N}{p} = const$, the generalization error is:

$$E_M^{(2)} = \Phi\left\{(-1)^i \frac{\delta}{2\sqrt{\frac{2p}{N\delta^2K^2} + \frac{L}{NK^2} + \frac{1-2L}{K^2}}}\right\}. \tag{3.1}$$

If $K >= 0$ then $i = 1$, otherwise $i = 2$.

Here $K = (1-\beta)\cos\varphi + \beta$, $L = \beta(1-\beta)(1-\cos\varphi)$, N is the number of training instances in *one class*, p is the dimensionality. We assume equal number of instances from each class are present in the training set.

3.2.3 Accuracy of the mixed classifier under incremental drift

Incremental drift consists of a number of small consecutive sudden drifts, say k drifts. Now we form a data sequence is generated by k data sources. Switching from one source to another means that the data (means) is rotated by 1^o. Thus round the circle we would have 360 data sources.

Let each data source generates an equal number of instances V before switching. Then the rotation speed is $v = \frac{1}{V}$. We assume that only full *old* concepts (consisting of V observations) can be included into the training set. The number of

training instances available at each point in time would be $n^{(k)} = V * (k-1) + t_2$. We derive generalization error of the mixed NMC classifier, trained on incrementally drifting data in Appendix A. The expression is:

$$E_M^{(k)} = \Phi \left\{ -\frac{\delta(T_C \cos\gamma_{k-1} + T_S \sin\gamma_{k-1})}{2\sqrt{\frac{4pKv}{\delta^2} + vK + (K-v)(T_C^2 + T_S^2)}} \right\}, \tag{3.2}$$

here $T_C = 1 + \sum_{i=1}^{k-2} \cos\gamma_i + t_2 v \cos\gamma_{k-1}, T_S = \sum_{j=1}^{k-2} \sin\gamma_j + t_2 v \sin\gamma_{k-1}$, $\gamma_i = \frac{i\pi}{180}$.

3.2.4 Analysis of the training window dynamics

Having the expressions for generalization error of NMC classifier we can analyze the dynamics of the optimal training window in relation to complexity and change.

Let $n^{(2)} = 2\beta N$ be the size of the training set available after the drift and $n^{(1)} = 2(1-\beta)N$ be the size of the data before the drift. Change happens at time t_D. Then for each time point after the change $t_i = n_i^{(2)}$ it is possible to find a training set size $n^* = n^{(1)} + n^{(2)}$, which would minimize the generalization error $E_M^{(2)}$ in Equation 3.1 and $E_M^{(k)}$ in Equation 3.2. That would be an optimal training window at time step t_i, which would depend on the complexity of the classification problem and the extent of drift.

3.2.4.1 Training window in relation to the extent of drift

An optimal size of a training window depends on the magnitude of a concept drift. The magnitude of the drift in our analytical model is determined by the rotation angle φ. $\varphi = 0^o$ represents no concept drift, $\varphi = 180^o$ represents a complete switch of the two class labels.

We are interested in finding an optimal training window size n^*. For that we need to minimize Equation (3.1) with respect to $n^{(1)}$, which is monotonically increasing. We do an optimization numerically, based on exhaustive search for $n^{(1)} = 1, \ldots, 200$, where $n^{(1)} = 200$ corresponds to the classifier trained on all the available data, while $n^{(1)} = 0$ is the new classifier (no old data).

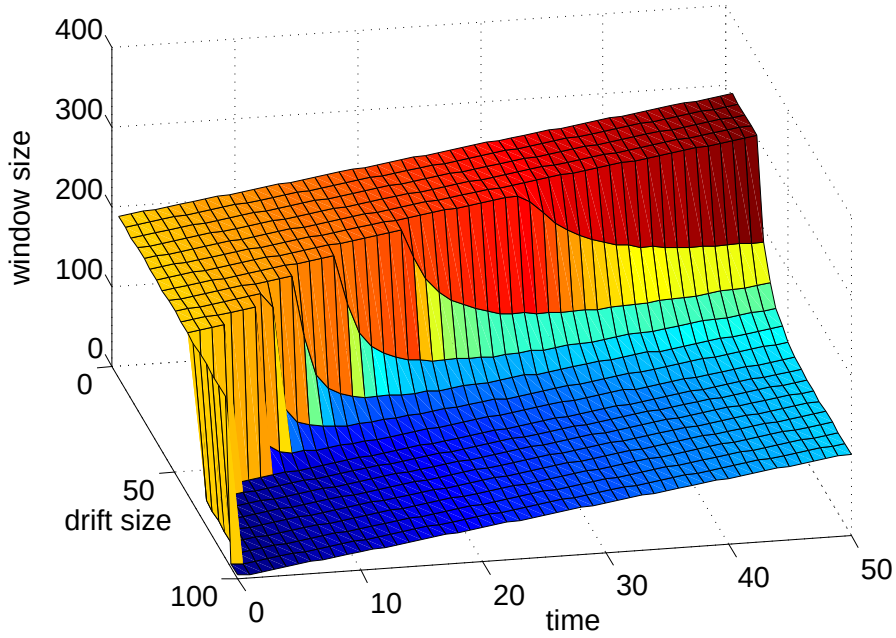


Figure 3.2: An optimal training window in relation to the drift magnitude

In Figure 3.2 the relation between the angle of rotation φ and optimal training window size n^* is depicted for time steps t after the drift. The experimental setting is: $\delta = 1$, $p = 7$.

The top 'balcony' area represents the cases when a lot of the old data is useful. That is when φ is small, way below 50^o, the data before and after the drift is rather similar. If the rotation angle gets very large (close to 90^o), optimal training set length is very small right from the first time steps after the drift. The data is too different.

3.2.4.2 Training window in relation to data complexity

Under similar setting we look at the relation between the training window and data complexity. Data complexity in classification problem can be delimited to dimensionality and separability of the data. In our model the separability is determined by the distance between the class means δ.

Similarly as in Section 3.2.4.1, we are interested in finding an optimal classifier training set length depending (a) on the separability between the classes and (b)

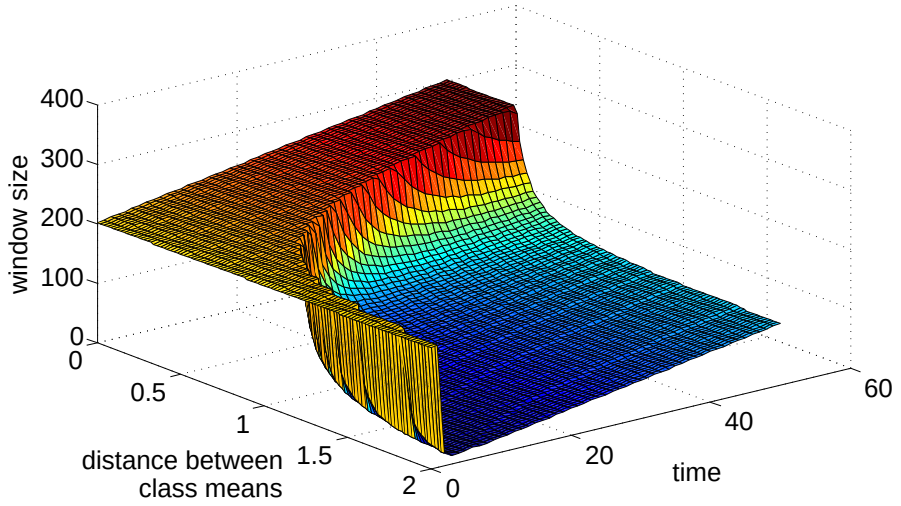


Figure 3.3: An optimal training window in relation to data separability.

on the dimensionality of the data, having the other parameters fixed. For that we do an optimization of D numerically for $n^{(1)} = 1, \ldots, 200$.

In Figure 3.3 the relation between the separability of the data δ and optimal training window size n^* is depicted for time steps t after the drift. The experimental setting is: $\varphi = 30^o$, $p = 7$.

The lower is the separability, the longer is the old data useful. When $\delta = 2$ a few new samples are enough to build an optimal classifier. In such cases the large distance between the classes allows maintaining the accuracy even having variations in the discriminant line.

In Figure 3.4 the relation between the dimensionality of the data p and optimal training window size n^* is depicted for time steps t after the drift. The experimental setting is: $\varphi = 30^o$, $\delta = 1$.

The old data is apparently useful when the dimensionality is high. The edge in the Figure 3.4 shows that under this setting the new classifier becomes more accurate than the old one when the sample from the source S_2 reaches 5 times the dimensionality.

The problem is often referred as 'a curse of dimensionality' [170]. If the sample size is small, increase in the number of features might actually degrade the classifier performance.

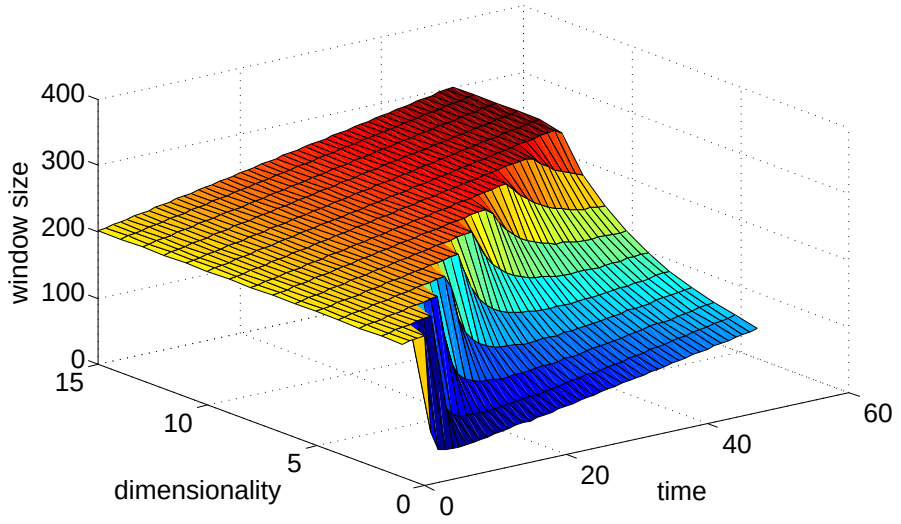


Figure 3.4: An optimal training window in relation to dimensionality.

3.2.4.3 Training window in relation to the speed of incremental drift

We use Equation (3.2) to analyze the relation between the speed of drift v and the optimal training window. Having the other parameters fixed we do an optimization of $E_M^{(k)}$ numerically for $n^{(train)} = 1, \ldots, 200$. $n^{(train)}$ will consist of a mix of data from the latest sources.

In Figure 3.5 the relation between the speed of drift and optimal training window is depicted for time steps t. The experimental setting is: $\delta = 1$, $p = 7$.

Note the two flat regions on the graphs A and B. Region A represents accumulation of training sample, since we start the gradual drift experiment with zero training instances. Region B on the graphs shows the actual relation between the drift speed and the training window. Under *constant speed* of drift the size of an optimal training window gets fixed.

3.2.5 Implications

NMC classifier requires relatively small number of training instances (at least $2p$), therefore can be called *a simple classifier*. There is no need to estimate the covariance matrix as compared to the Linear Discriminant classifier (LDC). In

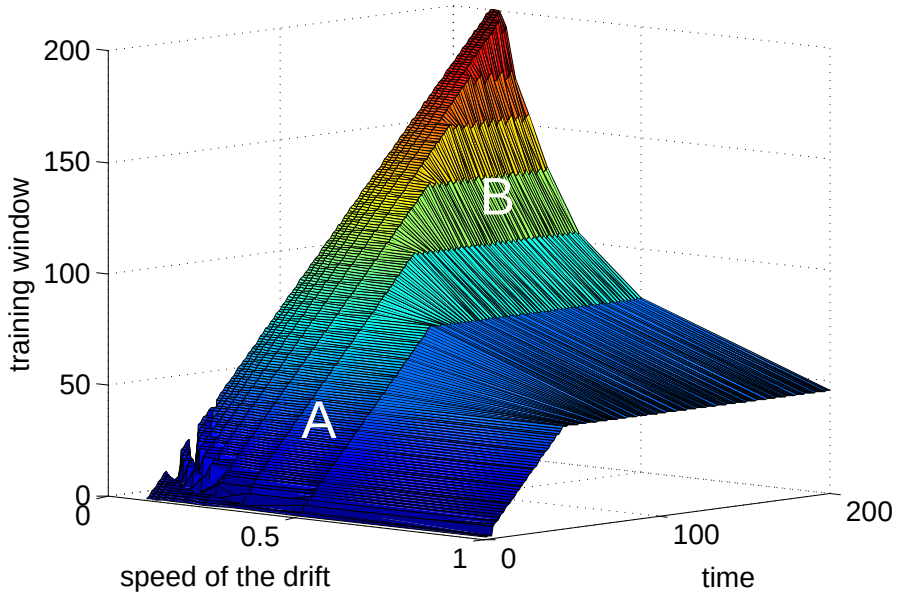


Figure 3.5: An optimal training window in relation to the speed of incremental drift.

streaming data setting NMC is fast, as compared to e.g. k Nearest Neighbors (kNN), decision tree (tree), Parzen Window classifiers (PWC). Even having a simple classifier we observe the regions where the switch point apparently differs from the change point. More complex classifiers would generally require more training data and thus exhibit even larger difference between the switch point and the change point.

We employ rotation model and NMC for analytical purposes. It might be argued that the pattern recognition problems are much more complicated in reality. We showed that even using a fast classifier (NMC) under simple settings there are cases when the old data is relevant after the drift. Thus the training window selection should not be limited to detecting the drift point.

This analytical study contributes to the understanding of the concept drift problem and indicates classifier training strategies under sudden and incremental concept drift.

3.3 Work Related to the Training Window Size

There are two main approaches to handling variable training window size. First, an explicit change detection is followed by a procedure to determine the size of the new training window [11, 74, 111, 134, 154]. The window resizing is guided by heuristics, and little attention is paid to the fact that the window needed for training the classifier and the window used for change detection should be considered separately.

The second approach to resizing the training window is based upon constant monitoring of the classification error, always assuming that there might have been a change in the past. A backward search is launched at each new observation (or batch thereof) in order to detect a past change point [1, 21, 105, 108]. While Klinkenberg [108] chooses the new window by directly estimating its classification accuracy, the other detection methods only determine the possible change point. There is no recommendation of what the *t*raining window should be. It is thereby assumed that the amount of data coming after the change is sufficient for training the new classifier. Window Adjustment Heuristics (WAH) proposed by Widmer in [211] is specific to a rule-based classification. The window is increased or decreased heuristically (e.g. by 20%), based on prespecified accuracy thresholds.

Some classifier ensemble methods developed specifically for concept drift can be regarded as more sophisticated examples of the latter group. They execute a version of the backward search by training ensemble members on past data of different lengths [112, 192, 222, 223]. The final classification is usually made by weighted voting where the weights correspond to the most recent accuracy of the individual classifiers. Thus the window representing the best classifier after a hypothetical change will acquire a large weight and will dominate the decision. Stanley [192] refers to this method as a 'windowless ensemble' but acknowledges that an implicit window selection takes place. An interesting recent addition to this group is the paired learners method [10] where change is sought by monitoring the difference in the accuracies of a stable classifier trained on the data since the last change and a 'reactive' classifier trained on a small window containing the latest w observations. A large discrepancy between the two accuracies is a signal of a change, and the old classifier is replaced with the

new one. The training window of the new classifier is of size w, the same as the length of the window for discovering the change. The method relies on two parameters, one of which is the window size w.

Below we detail three window resizing methods that can be used with any online classifier model. The three methods are used as the closest rivals to the window resizing method proposed here. The pioneering WAH [211] is specific to a rule-based classification thus the results were not competitive and we do not include it into peer algorithm group.

Drift Detection (GAM).[1] The drift detection method proposed by Gama et al. [74] keeps track of the probability of error for the incoming observations. The training window grows until a change is detected. When the error is found to exceed a certain threshold, the system enters a *w*arning mode and stores the time, t_w, of the corresponding observation. If the error drops below the threshold again, the warning mode is canceled. However, if the error exceeds the second (higher) threshold while in the warning mode, a change is recorded. The new training window is taken to be the streaming data that came after time t_w. The classifier is re-trained and the warning and detection thresholds are re-set.

Window Selection Algorithm for Batch Data (KLI). Klinkenberg [108] proposes to select a past window of batches so that the classifier trained on that data has minimum error rate on the newest data batch. The original method was designed for Support Vector Machines (SVM) and was equipped with a fast approximation of the leave-one-out error. Without such an approximation, the cost of applying KLI could be prohibitive.

Adaptive Windowing Algorithm (BIF). Bifet and Gavalda [21] propose a method for change detection and window resizing in 1-dimensional streaming data. The idea is: whenever two 'large enough' sub-windows of the past data exhibit 'distinct enough' averages, the older sub-window is dropped. The remaining window is partitioned again. The process is repeated until no partition into two pieces indicates distinct distributions. A confidence value D is used within the

[1] We chose to abbreviate the methods that we will use in the comparison by the first three letters of the surname of the first author. The method that we propose will be referred to as Window Resizing and abbreviated as WR.

algorithm. If there is no change in the mean of the streaming data from 1 to t, then the probability that BIF shrinks the window at step t is at most D (a bound on the false positive rate). Also, if there exists a split of the data such that the two true means differ by more than twice the threshold ϵ, then with probability $1 - D$ BIF detects the change and shrinks the window (a bound on the false negative rate).

3.3.1 Change detection

Explicit or implicit detection of concept drift can be based upon change in

- Probability distributions. If the class-conditional distributions or prior probabilities for the classes drift away from their initial values, the new data will not fit the old distributions. Based on how well the assumed distribution accommodates most recent data, a change can be detected and old data should be forgotten [50, 76, 143, 183]. Methods for change detection in this case include estimating the likelihood of new data with respect to the assumed distributions, and comparing the likelihood with a threshold.
- Feature relevance. A concept drift may lead to a different relevance pattern of the features describing the observations. Features or even combinations of attribute values that were relevant in the past may no longer be sufficiently discriminative [67, 85, 209].
- Classification accuracy. This is the most widely used criterion for implicit or explicit change detection [11, 74, 110, 111, 134].

To sum, there are theoretical statistical change detection methods, which implicitly assume that when the change is detected, we already do not need the old data.

3.4 Change Point vs. Classifier Switch Point

In this section we analyze the relationship between concept drift point and the classifier switch point. We derive an expression for the switch point, i.e., the number of observations after the change, sufficient to train the new classifier. Until this size is reached, the old classifier should be used. A detailed expression for the window size is derived for two equiprobable Gaussian classes, using LDC as the base classifier.

Switch point is directly related to the training window. Switch point is the point in time, where an optimal training window starts after the change. Change point is the point in time, where one data generating source is instantly replaced by another.

3.4.1 Setup and basic assumptions

Consider a real-time classification scenario. At time t_D a sudden concept drift occurs, in which the data generating source is replaced. Assume that t_D is known but the probability distributions corresponding to the two sources are unknown. Suppose that we choose a classifier model and train it incrementally by expanding the training window. At t_D because of the change in data generating source the trained classifier becomes obsolete and needs to be replaced. Let C_1 be the classifier trained on the data from the previous source, and C_2 be the classifier trained on the data from the new source. Since the data comes in a stream, it would be in deficit straight after the change, and the newly trained C_2 will have erratic performance. On the other hand, if the two data generating sources are similar, the old classifier may still be more accurate than the new classifier until a sufficient amount of the new data is accumulated.

We are interested in finding a relationship between the error jump and the size of the data window used for training the classifier after the concept drift. In this way we can estimate the 'switch point', i.e., the time point t_{switch} ($t_{switch} > t_D$) at which we should stop using C_1 and start using C_2. Figure 3.6 illustrates the problem.

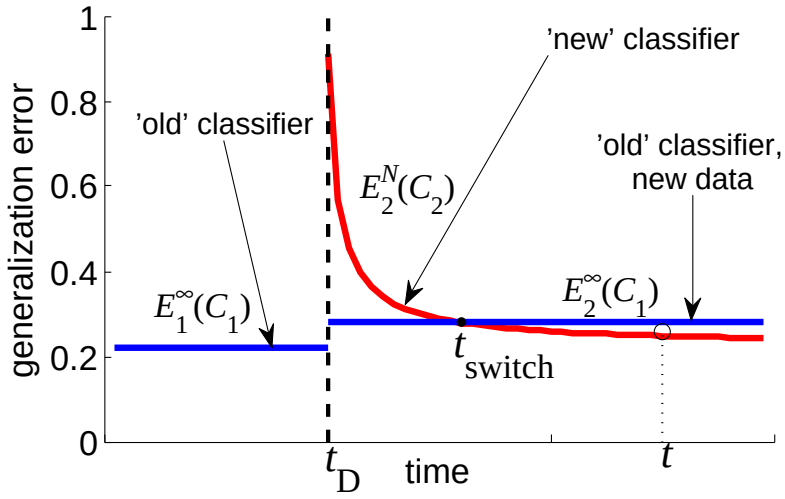


Figure 3.6: Error rates of C_1 ('old') and C_2 ('new'). The time of the concept shift, t_D, is indicated by a vertical dashed line.

3.4.2 Optimal window size after sudden concept drift

In this section we give the theoretical derivation of the optimal window size, determining the classifier 'switch point'.

3.4.2.1 The general case

Let C be the chosen classifier whose parameters are calculated from a sample of size N. Denote by $E^N(C)$ the theoretical error achievable by C on a training data set of size N. Let $E(C)$ be the asymptotic error rate of C obtained as $E(C) = \lim_{N\to\infty} E^N(C)$. Fukunaga and Hayes [71] show that, for a parametric classifier C, regardless of the types of the probability density functions (pdfs) and the priors, the classification error can be expressed approximately as

$$E^N(C) \approx E(C) + \frac{1}{N} f(C), \tag{3.3}$$

where $f(C)$ is a function that depends on the classifier type, the pdfs, but not on N. Denote by $E_i^N(C_j)$ the generalisation error of classifier C trained on N data points from source S_j with respect to the probability distributions in source S_i. Assuming that the training window for S_1 is sufficiently large, C_1 is trained to reach its asymptotical error $E_1(C_1)$. At the onset of the change at t_D, the

error of the classifier jumps to $E_2(C_1)$. It is expected that the change renders C_1 inadequate for the data from S_2, hence $E_2(C_1) > E_1(C_1)$ (Figure 3.6). If a new classifier is trained starting with the first observation after t_D, and the training set is augmented after each observation, the error of this classifier would be $E_2(C_2) + \frac{1}{N} f(C_2)$. To find the optimal switch point from C_1 to C_2, we solve for N the following equation

$$E_2(C_1) = E_2(C_2) + \frac{1}{N} f(C_2). \tag{3.4}$$

The switch point is when the size of the training window of data coming from S_2 reaches

$$N^* = \frac{f(C_2)}{E_2(C_1) - E_2(C_2)}. \tag{3.5}$$

Variants of $f(C)$ are tabulated for various classifiers and pdfs in references [71, 171]. The error values $E_i(C_i)$ can be derived for specific distributions and classifiers [170].

Note that N^* is not merely a window in the standard sense, it is the 'switch point' from the old to the new classifier. It is the number of training instances from N_D to N_{switch}.

3.4.2.2 LDC for two Gaussian classes

Let C be the linear discriminant classifier (LDC) [58] applied to two equiprobable p-dimensional Gaussian classes with identical covariance matrices Σ. Let $\delta^{(j)}$ be the Mahalanobis distance between the class means for source S_j, $j = 1, 2$. The error of LDC for this case is the Bayes error, and is calculated as [170]

$$E_2(C_2) = \Phi\left(-\frac{\delta^{(2)}}{2}\right), \tag{3.6}$$

where Φ is the cumulative distribution function of the standard normal distribution. The function relating the sample size and the classification error for LDC is [71]

$$f(C) = \frac{1}{2\sqrt{2\pi}\delta}\left[\left(1 + \frac{\delta^2}{4}\right) n - 1\right] \exp\left(-\frac{\delta^2}{8}\right). \tag{3.7}$$

For $f(C_2)$ we use $\delta = \delta^{(2)}$. The only unknown term in (3.5) is $E_2(C_1)$ which depends on the type and magnitude of the change. Assuming that only the class means change while the common covariance matrix remains the same from S_1 to S_2, we derive in the Appendix A Section A.3 the following expression for $E_2(C_1)$

$$E_2(C_1) = \frac{1}{2}\left\{\Phi\left(-\frac{\mathbf{w}^T\Delta_1}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right) + \Phi\left(\frac{\mathbf{w}^T\Delta_2}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right)\right\}, \tag{3.8}$$

where Δ_i is the difference between the means for class ω_i after and before the changes. Let $\mu_i^{(j)}$ be the mean of class ω_i in source S_j, $i, j = 1, 2$, and Σ be the common covariance matrix for the classes in both sources. Then $\Delta_1 = \mu_1^{(2)} - \mu_1^{(1)}$ and $\Delta_2 = \mu_2^{(2)} - \mu_2^{(1)}$. The vector with coefficients $\mathbf{w}$ comes from C_1 trained on S_1, and is given by $\mathbf{w}^T = (\mu_1^{(1)} - \mu_2^{(1)})^T\Sigma^{-1}$. With all terms in place, the optimal switch point for this special case is

$$N^* = \frac{\frac{1}{2\sqrt{2\pi}\delta^{(2)}}\left[\left(1 + \frac{\left(\delta^{(2)}\right)^2}{4}\right)n - 1\right]\exp\left(-\frac{\left(\delta^{(2)}\right)^2}{8}\right)}{\frac{1}{2}\left\{\Phi\left(-\frac{\mathbf{w}^T\Delta_1}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right) + \Phi\left(\frac{\mathbf{w}^T\Delta_2}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right)\right\} - \Phi\left(-\frac{\delta^{(2)}}{2}\right)}. \tag{3.9}$$

For clarification regarding estimation of the covariance matrix see Appendix A Section A.4.

It is assumed that the change point N_D is known with certainty.

Example 3.4.1. Consider two Gaussian classes in $\Re^5$ with $\mu_1^{(1)} = [0.5, 0, 0, 0, 0]^T$, $\mu_2^{(1)} = [-0.5, 0, 0, 0, 0]^T$ for source S_1, and $\mu_1^{(2)} = [1.0, 0.3, 0, 0, 0]^T$, $\mu_2^{(2)} = [0.5, 0, 0, 0, 0]^T$ for source S_2. In both sources the covariance matrix Σ was obtained from an identity matrix of size 5 by setting $\sigma_{1,2} = \sigma_{2,1} = 0.3$. An illustration is shown in Figure 3.7.

Using Equation (3.9), we get an optimal training window size $N^* = 42$. This means that after the substitution of S_1 with S_2, the old classifier is expected to be more accurate than the new classifier for the first 42 instances from S_2, then the new classifier is more accurate. □

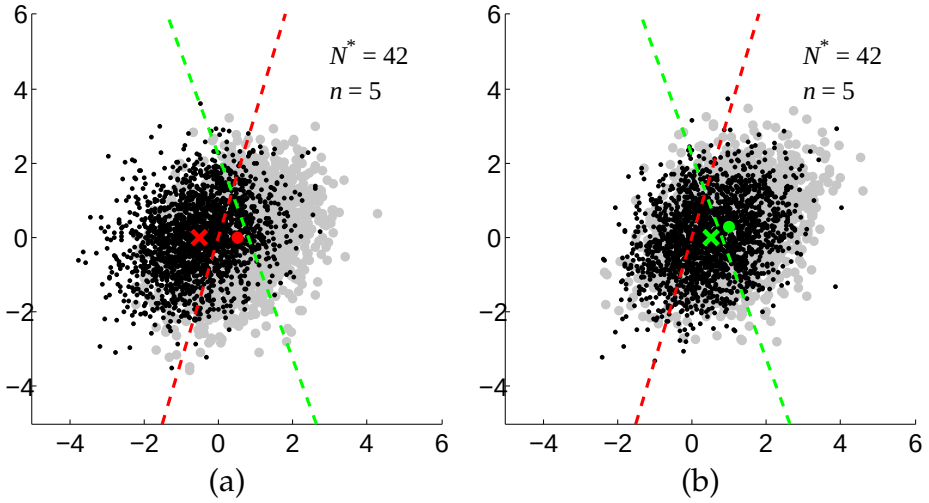


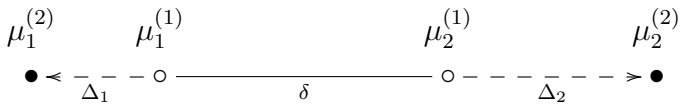
Figure 3.7: Data from sources S_1 (a) and S_2 (b), plotted in the first two dimensions. The centers of the two classes are marked. The optimal discriminant lines for the two sources are shown in both plots.

3.4.3 Analysis and simulations

Deriving the theoretical switch point does not automatically offer an algorithm for classification in the presence of concept drift. The obtained result can be used further for constructing plug-and-play algorithms. Such an algorithm requires a multitude of choices to be made, e.g., classifier model, change detection method, pdf approximations, error approximations, $f(C)$ approximation. The success or failure of such an algorithm can be attributed to any of the choices.

3.4.3.1 Optimal N^* in relation to the magnitude and the direction of the drift

For the 1-dimensional case, we can investigate the relationship between the direction and the magnitude of the drift in the class means and the optimal switching point N^* (3.9). Without loss of generality, assume that $\mu_1^{(1)} < \mu_2^{(1)}$. Let $\delta = \mu_2^{(1)} - \mu_1^{(1)}$ be the distance between the two means in the distributions of source S_1. In the new distributions coming from source S_2, $\mu_1^{(2)} = \mu_1^{(1)} + \Delta_1$ and $\mu_2^{(2)} = \mu_2^{(1)} + \Delta_2$. The common variance in both cases was 1. The set-up is depicted below



The experimental simulation is designed as an illustration and not proof of concept. We varied Δ_1 and Δ_2 independently in the interval $[-\delta, \delta]$. Figure 3.8 gives a color plot and a surface plot of $\log(N^*)$ as a function of Δ_1 and Δ_2 for $\delta = 4$.

The ridge along the diagonal section from $(\delta/2, -\delta/2)$ to $(-\delta, \delta)$ reflects the case where the two centers migrate symmetrically about the midpoint, i.e., when $\Delta_1 = -\Delta_2$. The denominator in (3.9) collapses to 0 because the old classifier is optimal for the new distributions ($E_2(C_1) = E_1(C_1)$), and N^* approaches infinity. For visualization purposes we cut off the peak by resetting all denominator values smaller than 10^{-5} to 10^{-5}. This is the cause of the flat top at the upper left corner $(-4, 4)$. The part of the diagonal starting from $(0, 0)$ and ending at $(-4, 4)$ corresponds to the case where the centers move apart, while the part from $(0, 0)$ down to $(-2, 2)$ corresponds to the two centers moving symmetrically towards one another. In both cases, the old classifier is optimal (regardless of the error), and $N^* \to \infty$. At $(-2, 2)$, the centers fall on top of one another, and no classifier can be better than random chance. Hence the ridge across from $(-\delta, 0)$ to $(0, \delta)$. For all pairs (Δ_1, Δ_2) on this line, $\Delta_2 - \Delta_1 = \delta$, which means that $\mu_1^{(2)} = \mu_2^{(2)}$ (the concept change merges the two classes into one). For this case the old classifier will be as useless as any classifier, $E_2(C_1) = E_1(C_1) = 0.5$, therefore $N^* \to \infty$.

The points (Δ_1, Δ_2) on the right diagonal correspond to the case $\Delta_1 = \Delta_2$, i.e. the classes are shifted together to the left or to the right. The old classifier in this case becomes progressively more inadequate with the size of the offset.

The dark subregions of A and B correspond to very small N^* (negative $\log(N^*)$) reflecting the case where even a very coarse and undertrained classifier for source S_2 is better than the old classifier C_1 trained on source S_1. Small values of N^* are not necessarily related with large error. Consider the pair $(\Delta_1 = -\delta, \Delta_2 = 0)$. Class 1 moves to the left by δ while class 2 stays put. The separability increases substantially, which leads to a smaller $f(C_2)$ term. On the other hand, due to the increased separability $E_2(C_2)$ may be much smaller than $E_2(C_1)$. Therefore N^* will be small. This situation is most prominently expressed in

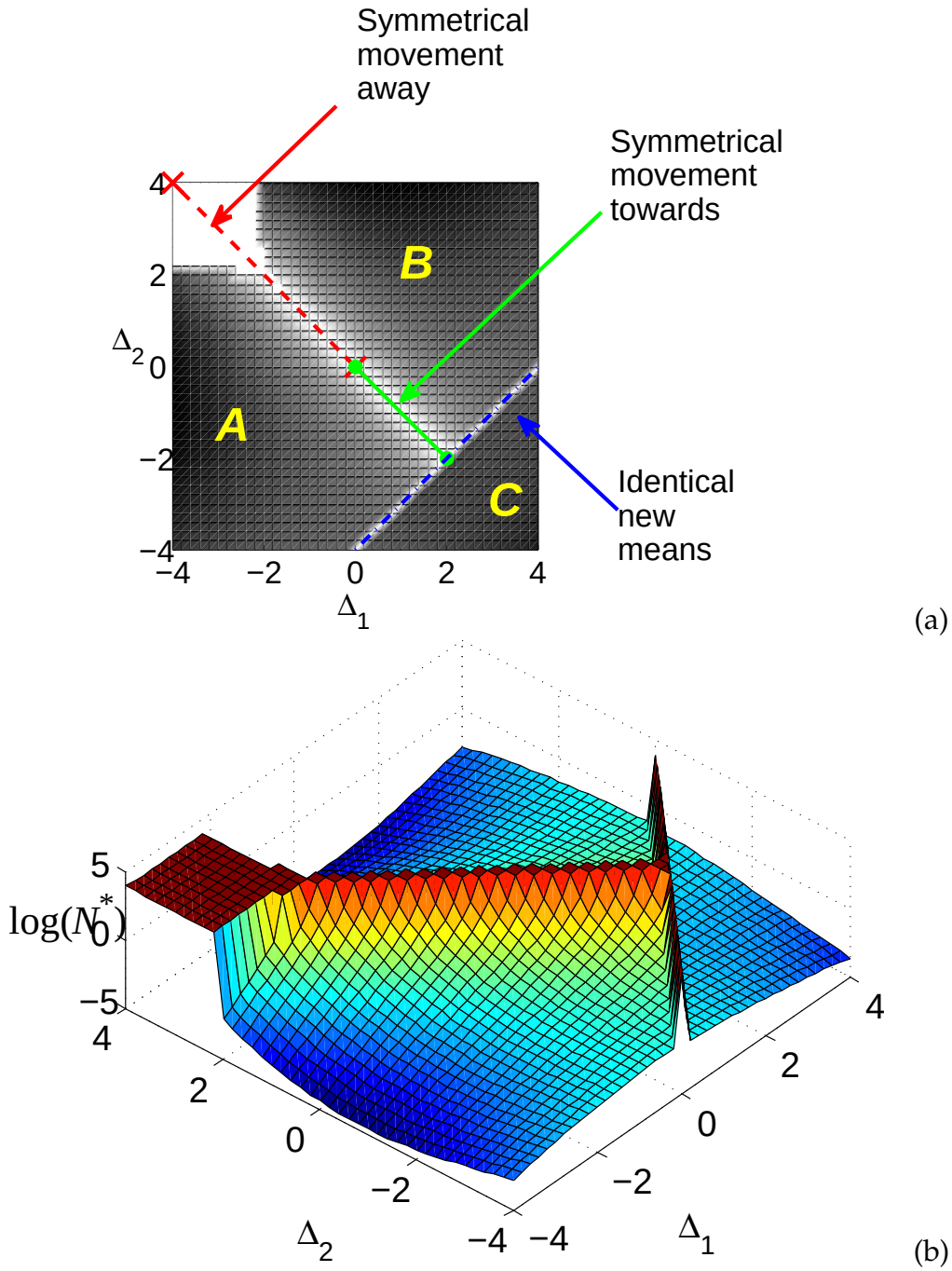


Figure 3.8: Plots of $\log(N^*)$ as a function of the offsets of the means, Δ_1 and Δ_2. (a) Color plot. Dark color corresponds to smaller N^*; (b) Surface plot (with restricted height) for $\log(N^*)$.

region C where the two means "swap places" so that, while $\mu_1^{(1)} < \mu_2^{(1)}$, after the concept change $\mu_2^{(2)} < \mu_1^{(2)}$. In this case C_1 will give the opposite labels in S_2 and will be worse than chance. The old classifier should be immediately replaced with C_2, even though C_2 might act as the largest probability classifier before proper training.

3.4.3.2 Numerical experiments

In order to examine equation (3.9) under controlled settings we use four artificial datasets and two real datasets with simulated drift. The characteristics of the datasets are summarized in Table 3.1, more details can be found in Appendix C.

Table 3.1: Summary of the used datasets (*all with simulated drift*).

Name	Dimensions	Size	Class balance	Drift type	Type of data
Gaus1	4	200	0.5:0.5	sudden	artificial
Stagger	9	120	changing	2×sudden	artificial
Hyper1	2	250	0.5:0.5	4×sudden	artificial
SEA*	3	800	changing	3×sudden	artificial
Vote1	16	435	0.61:0.39	sudden	real
Iono1	34	351	0.61:0.39	sudden	real

Gaus1 includes two Gaussian classes, the mean of the two classes shift independently at the drift point. Stagger is generated using STAGGER framework [185]. Hyper1 is a version of a moving hyperplane data, rotating clockwise at the drift points [93, 112]. SEA* is generated based on the framework provided in [193], the classes are determined by a straight line in a rectangle. The drift is simulated by moving the line in parallel directions.

We compare the following classifier training scenarios:

- A. *No update.* Running C_1 all the way through the streaming data.
- B. *Update without forgetting.* Online updating of C_1 without forgetting past data.
- C. *Complete forgetting.* Switching classifiers at t_D by dismissing C_1 and starting the training of C_2.

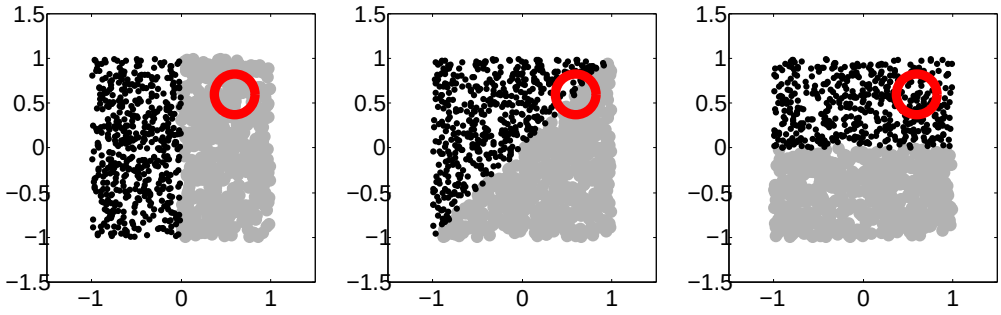


Figure 3.9: Three stages of the moving hyperplane data (0°,45°,90°.)

- D. *Partial forgetting.* Switching classifiers at N^*, where the means in both distributions are estimated from the streaming data.

In order to bypass the issue of change detection, we assume that the change points are known for scenarios A, C and D.

For each data point submitted as a part of the stream, an independent testing set of 100 objects was generated and labeled according to the current class description. The classifier was retrained and tested after each observation. The four classifiers were trained and tested on identical data in order to enable pairwise comparison. The nearest mean classifier (NMC) was used. The evaluation of (3.9) requires only the class means to be estimated from the old and the new distributions. We should mention the following three issues

- *Multiple changes.* We note that in scenario A, the classifier is trained on the first bout of streaming data, up to the first change. The same classifier was applied in any subsequent changes. In the calculation of $N*$, however, we used a different C_1 after each change. For example, with the moving hyperplane data, classifier C_2 trained after rotation to 45° becomes the 'old' classifier with respect to the next rotation to 90°. Following that, the new classifier is taken as C_1 with respect to the next rotation to 135°, and so on.

- *Single class.* If only points from one of the classes are available, then NMC will label all the data to that class. As the points come randomly, the probability of error of C_2 at the first few observations is likely to reach level $(1 - \max_i P(\omega_i))$, i.e., 0.5. In that case $N^* = \infty$, because there is no difference as to which classifier to use. That means no switch is

Table 3.2: Total error for the 4 scenarios (in %). Statistical significance is evaluated using a paired t-test w.r.t. to the scenario D. '•' indicates that D significantly outperformed the scenario, '∘' indicates that D was significantly worse than the compared scenario, '−' indicates no statistical difference ($\alpha = 0.05$).

Data	A = Without update	B = Without forgetting	C = Complete forgetting	D = Partial forgetting
Gaus1	20.53 •	19.52 −	19.74 •	19.47
Stagger	48.92 •	31.00 •	9.99 ∘	12.57
Hyper1	51.36 •	31.78 •	9.78 •	9.62
SEA*	11.12 •	9.85 •	9.14 •	8.90
Vote1	15.01 •	12.19 •	12.73 •	11.19
Iono1	28.60 •	26.89 •	27.92 •	26.54

recommended. In this case $\delta = 0$ and due to that $f(C) = \infty$, while $E_2(C_2) = E_2(C_1) = (1 - \max_i P(\omega_i))$.

- *Premature switch.* Suppose that we start counting the observations from t_D onwards. At observation t_i we re-evaluate the window, denoted N_i^*. If $i < N_i^*$, then the old classifier is still useful, otherwise we switch to C_2. Since we do not switch back and since there is noise in the estimate of N^*, it is likely that the switch comes earlier than necessary.

One hundred runs were carried out for the artificial data. Figure 3.10 plots the testing error rate for the three data sets. Table 3.2 shows the overall error and its 95% confidence interval. The overall error is the testing error averaged across the whole online run. Since all 4 classifiers were trained and tested on identical data, paired t-tests were carried out between scenario D on the one hand, and A, B, and C, on the other hand. The results are indicated in the table.

For the real data with simulated drift, three hundred runs were carried out, using different random permutations of the data. The sequential learning framework was employed, i.e., the classifier was trained on data instances $1, 2, \ldots, t$ and tested on instance $t + 1$. Table 3.2 shows the overall error and its 95% confidence interval.

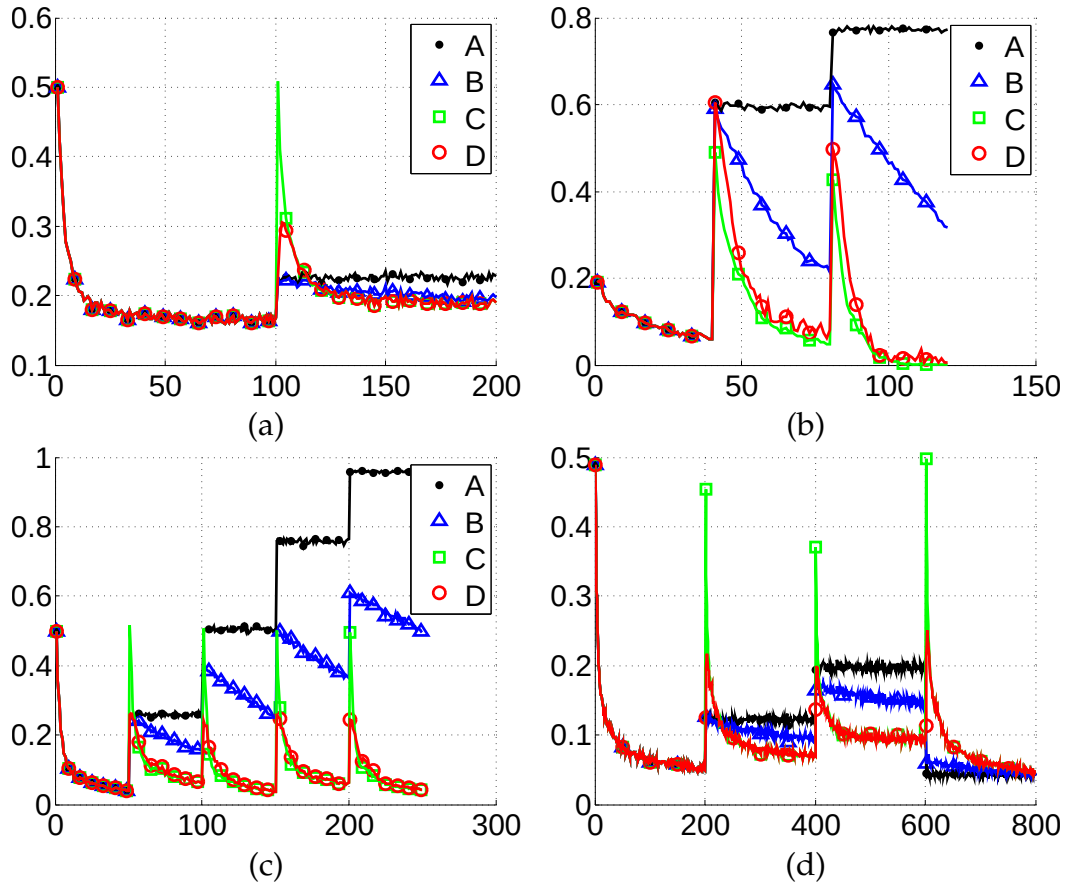


Figure 3.10: The 4 scenarios with the Gaussian data (a); STAGGER data (b); the moving hyperplane data (c) and SEA data (d) where A = No update; B = Update without forgetting; C = Complete forgetting; D = Partial forgetting with optimal window size.

The window approach D is significantly better than the other three approaches except for the Stagger data where immediate switching to C_2 is significantly better than the window approach. The reasons for this exception are several: (1) there is no peak of the error $E_2^N(C_2)$ above $E_1(C_2)$ for small N; (2) the concepts are very different and also have different priors; (3) there is no noise in the data. Therefore the new classifier is more useful right away. The statistical differences are estimated only as an illustration because they depend on the chosen time length of the streaming data. Our approach is meant to act as an 'oracle' identifying the most accurate of the two original classifiers at each time and switch as soon as N^* is reached. Thus there are intervals where the running error of D coincides with that for either C_1 or C_2. In the part before the change, the NMC classifiers for the 4 scenarios use identical training and testing data

and have the same running error. If the streaming data is let to run long enough, the significance of the error peak being shaved off will be smoothed over, and scenarios C and D will be indistinguishable.

Note that only the Gaus1 data satisfies the assumptions of the model; the other five cases illustrate that the theory might be useful even when the assumptions do not hold.

Assuming that sufficient data is available from the old distribution, the estimates of $\mu_1^{(1)}$ and $\mu_2^{(1)}$ would be stable. The inaccuracy of estimating $\mu_1^{(2)}$ and $\mu_2^{(2)}$ from a small sample of streaming data coming after the change will induce instability of the estimate of N^* causing premature switch. With the Gaussian data, where the distributions are guessed correctly, this will lead to curve D (partial forgetting) being closer to curve C (complete forgetting) rather than following curve A (no update), or even better curve B (update but no forgetting). Bias in N^* comes from wrongly guessed distributions as for the Stagger data and moving hyperplane data.

3.4.4 Practical issues

To calculate the optimal switch point N^* from data, we need to know the time of the change, t_D, the errors $E_2(C_1)$ and $E_2(C_2)$, and the term $f(C_2)$ that accounts for the error component coming from inaccurate estimates of the parameters of the classifier.

Change detection methods can be employed to determine t_D. If the change time t_D is detected at a later time $t_d < t_D + N^*$, nothing is lost because the optimal classifier has been running all the way to the detection. Large changes can be detected quicker than small changes. For small changes, however, N^* is larger, thus allowing for a larger detection time.

Application of the switch point calculation is not straightforward if we drop the assumption of Gaussian densities. In that case the estimation of the three terms in (3.5) requires suitable data sets to train and test the chosen classifier. With estimation techniques in place, an algorithm for resizing of a moving training window can be developed.

3.5 Online Window Resizing Algorithm

In previous section we derived a theoretical classifier switch point after a sudden concept drift. Here we present a method for choosing the training window size for online classification of sequential data. The method is based on theoretical classifier switch point estimation. The method gives the optimal window size for two equiprobable multivariate Gaussian classes where the known change consists in a shift of the means. We develop a generic framework and implement the algorithm using LDC classifier.

3.5.1 Problem setup

Consider a streaming data scenario described in Section 3.2. We are interested in finding an optimal training window for time point t. For that we need to detect the change point t_D and decide when classifier C_1 needs to be replaced by C_2 so as to minimize the generalization error. Figure 3.6 illustrates the problem.

Let $E^N(C)$ be the error rate of classifier C trained on N observations. Denote by $E^\infty(C)$ the asymptotic error rate of C obtained as $E^\infty(C) = \lim_{N\to\infty} E^N(C)$. Let $E_i(C_j)$ denote the error incurred by classifier C_j trained on data from source S_j with regard to the probability distributions in source S_i, where $i = 1, 2$. Shown in Figure 3.6 are the error rates of C_1 and C_2, the concept drift point t_D and the point of classification decision t. At t we should be using C_2 because its error rate $E_2^N(C_2)$ is smaller than the error rate of the old classifier C_1 i.e., $E_2^N(C_2) < E_2^\infty(C_1)$. It should be noted that the 'paired learners' method of Bach and Maloof [10] also examines the estimated accuracies $E_2^N(C_2)$ and $E_2^\infty(C_1)$ and makes a decision in favor of C_2 when the above inequality holds.

The optimal window size after the change as in Equation (3.5) is

$$N^* = \frac{f(C_2)}{E_2^\infty(C_1) - E_2^\infty(C_2)}. \tag{3.10}$$

3.5.2 Training window for two Gaussian classes

Consider two equiprobable Gaussian classes in $\Re^n$ with means μ_1 and μ_2, and equal covariance matrices Σ. Assume that the change is an instant (uncoordinated) shift of both class means occurring at some known time point t_D.

With $E_2^\infty(C_2) = E_2(C_2)$ (Equation (3.6)), $E_2^\infty(C_1) = E_2(C_1)$ (Equation (3.8)) and $f(C_2) = f(C)$ (3.7) in place, and with a known change point t_D, we can calculate the optimal window size N^* from (3.10). Denoting the **optimal switch point** by t_{switch}, we get $t_{\text{switch}} = t_D + N^*$. Thus, the optimal window size at t is

$$N(t) = \begin{cases} t, & \text{if } t < t_{\text{switch}}, \\ t - t_D + 1, & \text{if } t \geq t_{\text{switch}}. \end{cases} \tag{3.11}$$

We propose to use this result even though the true distributions may not be Gaussian.

3.5.3 Estimating class means

The class means before and after the change are estimated from the data.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_{N_1}$ be the data set from class 1 and $\mathbf{y}_1, \ldots, \mathbf{y}_{N_2}$ be the data set from class 2. The estimates for the parameters needed to calculate N^* (3.10) are

- $\bar{\mathbf{x}} = \dfrac{1}{N_1} \displaystyle\sum_{i=1}^{N_1} \mathbf{x}_i, \quad \bar{\mathbf{y}} = \dfrac{1}{N_2} \sum_{j=1}^{N_2} \mathbf{y}_j;$
- $S = \dfrac{1}{(N_1 + N_2 - 2)} \left(\displaystyle\sum_{i=1}^{N_1} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T + \sum_{i=j}^{N_2} (\mathbf{y}_j - \bar{\mathbf{y}})(\mathbf{y}_j - \bar{\mathbf{y}})^T \right);$
- $\hat{\delta} = (\bar{\mathbf{x}} - \bar{\mathbf{y}})^T S^{-1} (\bar{\mathbf{x}} - \bar{\mathbf{y}}).$

If dimensionality permits, a sample covariance matrix can also be estimated and the linear discriminant classifier can be applied. Otherwise, we can resort to the Nearest Mean Classifier (NMC) which only requires the class means in order to operate. In the latter case δ becomes the Euclidean distance. An unbiased

estimate of the squared Euclidean distance δ^2 can be derived (Appendix A Section A.5 as

$$\delta^2 = \hat{\delta^2} - n\left(\frac{1}{N_1} + \frac{1}{N_2}\right), \tag{3.12}$$

where $\hat{\delta^2}$ is calculated from the *estimates* of the means, N_1 and N_2 are the sample sizes for classes 1 and 2, respectively, and n is the data dimensionality. The correction should be applied for both $(\delta^{(1)})^2$ (before the change) and $(\delta^{(2)})^2$ (after the change).

3.5.4 Change detection using the raw data

The estimation of the probabilities of error and the parameters of the distributions needed for evaluating Equation (3.10) hinges upon an accurate estimate of the change point t_D.

We propose to use the raw data for the change detection. The reason for this choice is threefold. First, the change detection is not tied up with a classifier model. Second, assuming that the classifier trails along at a suboptimal performance, the error variations may not be sufficiently indicative of a concept change. On the other hand, differences in the data may signal a change that will lead to training a better classifier than the ongoing suboptimal one. Third, usually error-based change detection only looks for increase in the error. A change in the data that is manifest by a trough in the running error may remain undetected. The current classifier may still be adequate. However, without detecting the change, we may miss the chance for an even better classifier for the new distribution.

Suppose that we have the labeled sequence of observations labeled in c classes. To estimate the likelihood of a change at time d, where $1 \leq d \leq t$, we assume that the class means migrate independently of one another. Then the probability that there is a change at time d is

$$P(\text{change}|d) = 1 - \prod_{k=1}^{c} P(\text{no change in } \mu_k|d),$$

where μ_k is the mean for class k, $k = 1, \ldots, c$. Given that the data lives in $\Re^n$,

the value of $P(\text{no change in } \mu_k|d)$ can be estimated using the p-value of the Hotelling multivariate T^2-test. This test compares the means for class k before and after the hypothetical change at d.

If we use the notation $p_k(d)$ as the p-value returned by the Hotelling T^2-test comparing class k samples before and after time moment d, the probability of change at d can be estimated as

$$P(\text{change}|d) = 1 - \prod_{i=1}^{c} p_k(d). \tag{3.13}$$

Calculation details of this change detection can be found in Appendix A Section A.6.

3.5.5 The WR* method

The WR* method is shown in Figure 3.11. We propose to use the optimal window size taking the change point with the maximum likelihood. Using (3.13) and (3.11), the optimal window size N^{WR^*} is calculated as

$$t_D = \arg\max_{j=1}^{t} P(\text{change}|j), \tag{3.14}$$

$$N^{WR^*} = N^*(t_D). \tag{3.15}$$

For comparison we will also include a version of the proposed method, called WR, where we do not use N^* but take instead the whole sample after the change

$$N^{WR} = t - t_D + 1. \tag{3.16}$$

The proposed method differs from the previous work in the following ways:

- We consider separately a detection window and a training window for the online classifier.
- The window size is derived using estimates of the class means, which eliminates the need to build and test classifiers on past windows.

THE WINDOW RESIZE ALGORITHM (WR*)

input: a sequence of labeled observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$.

1. Run backward search to find the likelihood $P(\text{change}|j)$ for $j = 1, \ldots, t$, using (3.13).
2. Estimate the change point with maximum likelihood
$$t_D = \arg\max_{j=1}^{t} P(\text{change}|j).$$
3. Use (3.11) to calculate $N^{WR^*} = N(t_D)$. (For WR, use $N^{WR} = t - t_D + 1$.)

output: window size N^{WR^*} (N^{WR}).

Figure 3.11: The Window Resize* Algorithm (WR*)

- The change detection is done on the raw (multidimensional) data rather than on the running error rate.

3.6 Experimental Evaluation of WR*

We compare WR* with the three methods GAM, KLI, BIF reviewed in Section 3.3 and with WR. We added a control scenario where the window is kept growing with the data regardless of any change ALL (see Appendix B for details). Thus the set of competing window resizing methods is: WR*, WR, GAM, KLI, BIF and ALL. For each dataset, artificial or real, we ran 6 synchronized experiments, one for each window resizing variant. The synchronization ensured that, once collated, the same sequence of data was submitted to each method. The Nearest Mean Classifier was used in all the experiments. The experimental design was chosen to showcase the proposed method, e.g., using small datasets and complex learning tasks.

BIF works only for 1-dimensional data, e.g., the running error. For this method to be used for n-dimensional raw data, a separate window is maintained for each dimension. The parameters needed for evaluating the discriminant functions

are calculated on the respective windows. For example, the n components of the cluster means in $\Re^n$ may be derived using different window sizes. This model reflects the fact that the features may change at different pace.

KLI, BIF, WR* and WR include complete backward search starting at the current observation whereas GAM limits the search to the latest detected change.

3.6.1 Results on artificial data

Three data sets were generated: Gaussian data (Gaus2), STAGGER data (Stagger) and the moving hyperplane data (Hyper2). All three of then are frequently used in the literature on concept change. However, the exact implementation varies from one study to another. Our protocol is detailed in Appendix C and the characteristics are summarized in Table 3.3.

Table 3.3: Summary of the used datasets.

Name	Dimensions	Size	Class balance	Type of drift
Stagger	9	120	changing	$2\times$sudden
Gaus2	7	400	0.5:0.5	sudden
Hyper2	2	250	0.5:0.5	$4\times$sudden

Note. All data is artificial and with simulated drift.

With each data set, for each observation in the sequence, we generated a random set of 100 observations to serve as the testing set at the current time point. The same testing set was used for all online classification methods in order to enable statistical comparison between them via paired t-test. Let $E(t)$ be the error rate of the online NMC at time point t, evaluated on the respective bespoke testing set. As an overall measure of the performance of the NMC we took the average of the errors at $t = 1, \ldots, t_{\text{end}}$, i.e., $E_{\text{total}} = \frac{1}{t_{\text{end}}} \sum_t E(t)$. Due to the random nature of the streaming data, we average E_{total} over 100 independent runs and report in the table that error, denoted $\bar{E}_{\text{total}}$.

The values of the parameters of Gaussian data were chosen to comply with the assumptions for optimality of WR*. These are the cases when the drift is not very large and the complexity of the task is relatively high.

Table 3.4: Testing error $\bar{E}_{\text{total}}$ (in %) for the artificial data. The best accuracy for each column is underlined. The symbol next to the error rate indicates that the respective method is significantly: • worse than, '$\circ$' better than, '$-$' no different to WR* ($\alpha = 0.05$).

Method	Gaus2	Stagger	Hyper2
WR*	<u>17.63</u>	28.49	<u>11.05</u>
WR	18.33 •	<u>21.39</u> $\circ$	11.49 $-$
BIF	18.30 •	36.87 •	22.46 •
KLI	18.26 •	39.77 •	11.99 •
GAM	18.30 •	36.81 •	17.18 •
ALL	18.30 •	36.81 •	22.46 •

The results are presented in Table 3.4. Statistically significant differences from a paired t-test are indicated. It should be noted that the overall performance metrics of the compared methods depends upon the composition of the sequential data for the test. If shorter intervals around the change were examined, the differences would have been more prominent but also more prone to flukes and mishaps. For Gaussian and Hyperplane data the batch size for KLI was fixed at 12 which is not a common multiplier of the change time.

For Stagger data a warm-up stage of 30 observations was used with all methods in order to train an initial NMC. The batch size for KLI was set to 30. Table 3.4 contains error $\bar{E}_{\text{total}}$. The accuracy of KLI in STAGGER example suffers from the batch mode, due to unequal priors the batches need to be made large.

Table 3.4 contains the results. Both STAGGER and hyperplane data illustrate situations with multiple drifts in contrast to the Gaussian data where only one change was simulated. STAGGER and hyperplane data illustrate the case when the underlying assumptions for optimality of WR* do not hold. In STAGGER the three concepts are extremely different, thus it makes sense to start a new classifier as soon as possible after the drift. This is perfectly in line with WR outperforming WR*. In Hyper2 data some of the regions are shared by the changed concepts with previous ones. Thus WR* is advantageous even though the distribution assumptions do not hold.

Table 3.5: UCI data.

Name	Dimensions	Size	Class balance
WRaustralian	14	690	0.56:0.44
WRbreast	30	596	0.64:0.36
WRcylinder	36	540	0.58:0.42
WRgerman (num)	24	1000	0.70:0.30
WRhepatitis	19	155	0.79:0.21
WRionosphere	34	351	0.64:0.36
WRStatlog heart	13	270	0.56:0.44
WRSPECT heart	22	267	0.79:0.21
WRsonar	60	208	0.53:0.47
WRvote	16	435	0.61:0.39

Note. All tada is real, with simulated sudden drift.

3.6.2 Results on real data sets

We used 10 datasets from the UCI repository [8] and simulated a concept change. With all the data sets, a cumulative error rate was maintained with the data sequence. The single error rate at time t, denoted $e(t)$, was estimated by testing the online classifier on the unseen data point coming at time t, before acquiring its class label ($e(t) = 0$ if correctly labeled, and $e(t) = 1$ if mislabeled). The cumulative error at time t is $E(t) = \frac{1}{t}\sum_{i=1}^{t} e(i)$. The final error $E(t_{\text{end}})$ was taken as the performance measure for the respective data set.

The data set that we chose (Table 3.5) are all 2-class problems with moderate size (100 -1000 instances) and dimensionality (10-100 attributes).

To simulate concept change, we first permute the data and fix this sequence. Then we take the second half of the data and shift-rotate features from 1 to 5. Thus for the second half of the data original feature 1 is fed to the classifier as feature 2, original feature 2, as feature 3, and so on, while original feature 5 is submitted as feature 1. We used the same change pattern with all the datasets. Note that the feature values or the class labels are not changed in any way. Similar procedures are commonly used for evaluating concept drift learners (e.g. [18, 131]), since it does not introduce any artificial elements to the real data itself, just manipulates the order of the data and allows to control the presence of drift.

Note, that since we employ a sequential testing procedure (presented in Section

2.1, the accuracies are not directly comparable with published accuracies on the same datasets, where cross validation or holdout testing procedures have been employed.

Sometimes there is an explicit suspected change point, e.g., due to change of operational circumstances, spatial location, or due to a time gap. For example, classification of network traffic may be affected by the release of a new software product at a particular (known) time; classification of customer preferences from retail records may face a concept change if a specialized store near by closes, etc. Thus, in addition to the experiments where the change point is unknown, we also test the methods for a known change point. KLI and ALL will give the same result as with an unknown change point. GAM, BIF and WR will cut the window at the change point. WR* will evaluate the data before an after the change point and then output the new window size. The results are shown in Tables 3.6 and 3.7. The differences between the errors of WR* and the other methods were not statistically significant apart from BIF in cylinder and GAM in SPECT heart, which are significantly worse than WR* in those cases.

We measure statistical significance of the differences between WR* an the peer algorithms using Bonferroni-Dunn test for multiple hypothesis testing as presented in [52]. The test is designed for comparing all classifiers against the control classifier (WR* in our case). The test statistics for comparing the i-th and j-th classifier is

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}}, \tag{3.17}$$

here R_i and R_j are the mean ratings of the classifiers, k is the number of classifiers and N is the number of datasets.

3.6.3 Analysis of the results

Since the same classifier (NMC) was used in all experiments, the differences in the error rates were due to the success of the change detection and the subsequent resizing of the training window. In WR*, WR, BIF, CLI and GAM the window size is related to the change point. Hence the results for these five methods are a direct consequence of the accuracy of the detection. On the other

Table 3.6: Testing error $\bar{E}_{\text{total}}$ (in %) for the experiment with change detection. The best methods for each data set are underlined. The methods are sorted by their mean rank. 'Statistically significant at' means that the difference is significant at the listed level.

Data set	WR*	KLI	WR	ALL	BIF	GAM
WRaustralian	35.34	34.33	35.92	35.34	35.34	35.34
WRbreast	11.71	11.88	11.71	11.71	11.71	11.71
WRcylinder	44.62	47.22	48.89	44.62	48.89	44.62
WRgerman	38.29	37.89	38.29	38.59	38.59	38.59
WRStatlog heart	38.48	39.22	38.48	39.22	38.85	39.22
WRSPECT heart	26.50	25.00	29.14	25.75	25.75	25.75
WRhepatitis	42.53	36.69	42.53	43.18	43.18	43.18
WRionosphere	25.86	25.00	26.14	27.57	27.57	28.43
WRsonar	37.92	41.30	37.92	37.92	37.92	37.92
WRvote	11.18	12.10	11.64	11.87	11.87	11.87
rank	2.60	3.20	3.50	3.80	3.95	3.95
statistically significant at	$\alpha =$	20%	10%	4%	3%	3%

hand, WR* calculates an *optimal* window size that may include observations from the stream *before* the change as well. As Table 3.4 shows, WR* performs better than the chosen competitors on the artificial data sets. To look for an explanation, we examined the progression of the window sizes along the sequence of observations. Figure 3.12 plots the window sizes against time for the Gaussian data experiment. The change point is indicated by a vertical dashed line. The diagonal line corresponds to taking all incoming observations in the training window. This happens for the methods that did not detect any change, in this case BIF, GAM and ALL. The classification error of these methods will decrease up to the change point, will peak after the change and gradually return as the observations from the new data source become the overwhelming part of the training window. On the other hand, KLI, WR and WR* react to the change by reducing the window size. The reduction is more prominent in the two WR variants. The advantage of WR* over WR is in the larger window sizes for the data before the change.

This demonstrates the robustness of WR* to false change detection. If a change is falsely detected, it is likely that the means before and after the change point

Table 3.7: Testing error $\bar{E}_{\text{total}}$ (in %) for the experiment without change detection. The best methods for each data set are underlined.

Data set	WR*	KLI	WR	ALL	BIF	GAM
WRaustralian	35.34	34.33	35.70	35.34	35.70	35.70
WRbreast	11.71	11.88	11.80	11.71	11.80	11.80
WRcylinder	44.62	47.22	42.12	44.62	42.12	42.12
WRgerman	38.09	37.89	38.09	38.59	38.09	38.09
WRStatlog heart	36.99	39.22	36.80	39.22	36.80	36.80
WRSPECT heart	25.75	25.00	27.07	25.75	27.07	27.07
WRhepatitis	43.83	36.69	44.48	43.18	44.48	44.48
WRionosphere	27.57	25.00	28.00	27.57	28.00	28.00
WRsonar	37.92	41.30	38.16	37.92	38.16	38.16
WRvote	11.87	12.10	11.98	11.87	11.98	11.98
rank	2.70	3.45	3.95	3.00	3.95	3.95
statistically significant at	$\alpha =$	12%	3%	28%	3%	3%

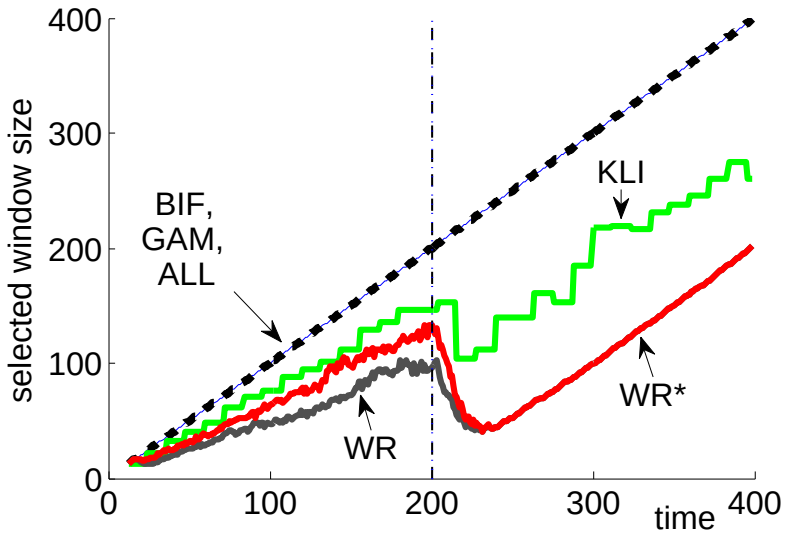


Figure 3.12: Training window size as a function of time for the Gaussian data.

would be similar. This implies that the old classifier may be useful longer, which will be picked up by the small denominator in the calculation of the optimal window size N^* in (3.11). On the other hand, WR will only use the data after the false change point which will adversely affect its performance.

Table 3.6 shows the average ranks of the methods as an overall performance

gauge. The best method for a given data set obtains rank 1, the second best obtains rank 2, and so on; the worst method obtains rank 6. If the errors tie, the ranks for the tied places are shared, which ensures that the sum of the ranks for each data set is $1+2+3+4+5+6 = 21$. The 6 methods were ranked with respect to each data set, and the ranks were then averaged (bottom row in Table 3.4). WR* has the lowest rank by a large margin. This suggests that the difference in the method's performance is rather due to the proposed window resizing than due to clever change detection. This result is matched in Table 3.7, where the change point is known. Our primary interest here is to compare WR* with WR. In this scenario WR was cutting the window at change point even if the change was not significant enough. Again the rank for WR* is better by a large margin. WR* benefits from separate change detection and window resizing mechanisms. We should also point out that KLI, which is based on heuristic search, performs very well too, scoring the second lowest rank after WR*.

The only parameter that is required by WR* is the significance level for the change detection. If we use the standard choice of $\alpha = 0.05$, the method is parameter-free. The change detection is done on the raw data because the calculation of N^* is based on changes in the class means anyway. This would be an advantage over methods that use the running error rate which may not be able to pick up a large change that does not have a very profound effect on the running error. Using the class means for change detection and for resizing the window can be regarded as a disadvantage at the same time because this only captures one aspect of a possible change. Changes in the variances of the classes or the class prevalences will not be picked by WR*. The method is optimal for two Gaussian classes, Linear Discriminant Classifier (LDC, equivalent to NMC for identity covariance matrices) and concept change manifest by shift of the means. Our experiments showed that WR* is useful even when the assumptions may not hold.

3.7 Conclusion

In this chapter we addressed the training set selection problem under sudden concept drift, which happens when one data generating source is instantly

replaced by another. Under these settings historical data needs to be cut at some point in time, that is a training window strategy. The major question for the training window strategy is what should be the window length at a given point in time.

The first research question (RQ1) consisted of three parts. (1) What determines an optimal training window size under sudden concept drift? (2) To what extent a change point is different from the start of the training window? (3) How this difference can be used to improve the accuracy of an adaptive learner?

We showed how the optimal window size under sudden concept drift, in terms of the model generalization performance, depends on data complexity (dimensionality and separability) and the extent of drift. For that we built a dynamic model of two Gaussian data classes under sudden drift. The theoretical analysis of generalization error of the nearest mean classifier showed that the more complex is the data (higher dimensionality, lower separability), the longer is the optimal training set, i.e. the longer history of the data is relevant, even if it comes from a different source. The analysis also showed that the larger the drift (or the faster in case of multiple drifts), the shorter history is relevant to build an accurate model.

Having identified that the training window cut point is different from the change point, we theoretically quantified this difference by constructing a model of two parametric classifiers with different lengths of training history and aligning their generalization errors. Based on that we derived an expression for the optimal training window size for the case of two Gaussian classes, the linear discriminant classifier and change abrupt consisting of shift in the means.

Based on the theoretical results we develop a window resizing algorithm WR* for classification of sequential data, which incorporates an explicit distinction of the change point and the training window cut. By extensive numerical experiments we demonstrate statistically significant difference in the testing accuracy in favor of the proposed method in comparison with three window resizing methods from the recent literature as well as with a baseline training strategy, which uses no adaptivity at all.

Thus, theoretical distinction between the training window and the change

point when determining a variable window size allows to improve generalization performance under sudden concept drift.

We note that the method particularly is useful in the domains where the changes are moderate and the complexity of the data is high. In such cases different detection and training windows are particularly needed because the old classifier is useful for longer time after the drift.

Chapter 4

Gradual Drift: Combining Similarity in Time and Space

In this chapter we present a concept of combining time and space similarity for training set selection under concept drift. The chapter focuses on learning under gradual drift, while in addition the two boundary cases (only time based similarity or only space based similarity) relate the findings also to sudden drift (Chapter 3) and reoccurring concepts (Chapter 5). A combined view to instance selection is needed due to complex nature of the real data. Therefore unified view to training sample selection is proposed which is flexible with respect to the actual changes.

The combination of time and space similarity under concept drift has not been addressed before. The concept itself and the following algorithm are our contributions to the field of concept drift research.

Using time and space similarity concept we develop a method for classifier training, particularly relevant when the expected drift is gradual. Training set selection is based on the distance to the target instance. Distances in feature space and in time is linearly combined. The proportions of time and space can be fixed a priori or learnable online. The developed algorithm can determine an optimal training set size online using cross validation on the historical data. The algorithm is a wrapper approach, that means it can be used plugging in different base classifiers. The proposed algorithm shows the best accuracy in

the peer group on the real and artificial drifting data. The algorithm complexity is reasonable for the field applications. The algorithm is particularly expected to demonstrate a competitive advantage under gradual non uniform drift scenarios in small and moderate size data sequences.

This chapter is based on our publication [226] and includes a material from our papers [227, 230].

The chapter is organized as follows. We start by giving an intuitive motivation for the need of combined (time and space similarity) view to training instance selection under concept drift in Section 4.1. In Section 4.2 we fix the framework and basic assumptions. Next we introduce and illustrate the concept of time and space similarity in Section 4.3. The following Section 4.4 outlines particularly related work and maps the proposed method within the related work. In Section 4.5 we present the proposed algorithm. Section 4.6 gives experimental setup and the results. Sections 4.8 and 4.9 discuss the results and conclude.

4.1 Motivation

Recall an application Example 1.1.2, where Kate was reading the news online. When she became more and more interested in real estate, market news were appearing more and more often as the most interesting topic. At the same time she was still interested in general news, but the relative interest was declining. Thus the relevance of a given document to Kate's interests depended on the age of the document (distance in time) and the content (distance in space).

In order to build a classifier, which would be reactive to a gradual concept drift we aim to select the most relevant historical instances as a training set. In Kate's example, for each incoming new document (unlabeled) we would look for *similar* documents within historical stock.

Similarity between two objects in instance based learning [4] defined as a function of distance in space. If the problem is non stationary, similarity in time is as well relevant. For a visual illustration look at a snapshot of Electricity data [84], which is provided in Figure 4.1.

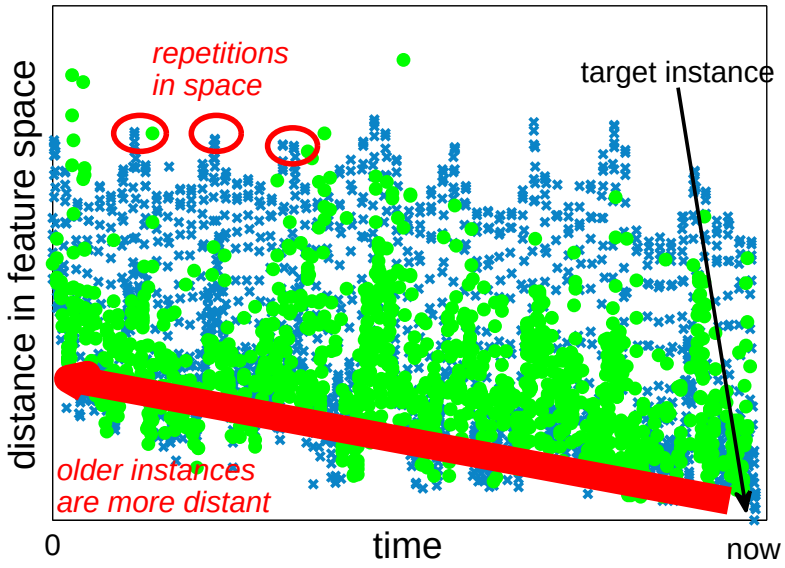


Figure 4.1: A snapshot of electricity demand data, green . and blue $\times$ mark classes of low and high demand.

We plot the similarity of the historical instances to the target instance over time. We use the Euclidean distance as similarity in space measure. The target instance is the very last in time (denoted 'now'), and its distance to itself in space is 0. The older instances are generally more distant from the target instance (declining slope along with x axis), which indicates the relevance of similarity in time. More recent instances are more similar to the target instance. In addition there are notable reoccurrences in space, indicated by circles. This implies that not all the most recent instances are the most similar in space. Thus the intuition suggests that in order to select the most similar instances as a training set, both space and time similarity needs to be taken into consideration.

4.2 Setup and Basic Assumptions

Let us set up a formal framework of the problem.

We employ sequential learning framework for testing, as presented in Section 2.1 and also used in Chapter 3. We shortly recap the setup below.

One data instance $\mathbf{X} \in \Re^p$ is received at a time. At time $t+1$ the task is to

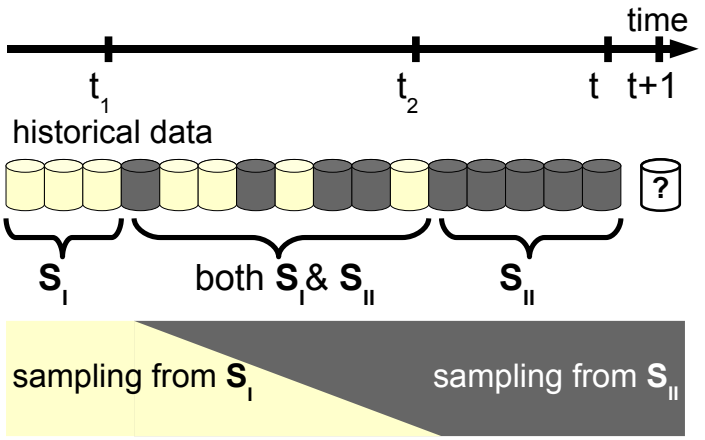


Figure 4.2: Gradual drift scenario.

predict the class label for the target instance $\mathbf{X}_{t+1}$. In this chapter without loss of the generality of the time and space similarity concept, we assume binary classification task and equal prior probabilities of the classes.

Any selected or all the historical labeled data $\mathbf{X}_1, \ldots, \mathbf{X}_t$ can be used to train a classifier. The class labels of the historical data are known, the label of $\mathbf{X}_{t+1}$ is unknown at time $t+1$. At time $t+2$ after the classification decision and receiving the true label, we can add $\mathbf{X}_{t+2}$ to the training data and proceed with the decision making for a target instance $\mathbf{X}_{t+1}$.

Consider a gradual drift scenario, illustrated in Figure 4.2. Up to time t_1 data generating source S_I is active. From time t_2+1 on the source S_I is completely replaced by the source S_{II}. In time interval (t_1+1, t_2) both sources are active and an instance comes from either one or the other source with a prior probability. The probability of sampling from S_{II} increases with time. A designer does not know when the sources switch. The task is to assign a class label to an observation received at time $t+1$. It is expected that a concept drift might have taken place.

We aim to select a training set, consisting of the instances, which are *similar* to the target observation. Similarity is a share of commonality. A detailed discussion on similarity concept can be found in [137].

4.3 Similarity in Time and Space for Training Set Selection

In this section we introduce and explore the concept of combining time and space similarity for training set selection for learner adaptation to a concept drift. First we define how to measure similarity and then look how to use it for training set selection.

4.3.1 The concept of similarity in time and space

Definition 4.3.1 (Time and Space similarity). We define *time and space similarity* between the target instance $\mathbf{X}_j$ and a historical instance $\mathbf{X}_i$ as a function

$$\mathcal{D}(\mathbf{X}_i, \mathbf{X}_j) = f(d_{ij}^{(S)}, d_{ij}^{(\mathcal{T})}),$$

where $d_{ij}^{(S)}$ is the distance between the two instances in space and $d_{ij}^{(\mathcal{T})}$ is the distance between the two instances in time. □

Distance in time between the instances $\mathbf{X}_i$ and $\mathbf{X}_j$ in case of equally spaced time intervals is defined as a function

$$d_{ij}^{(\mathcal{T})} = f(|i - j|). \tag{4.1}$$

The function is chosen by a designer. For instance, exponential function can be used if a designer is willing to emphasize the recent times, $d_{ij}^{(\mathcal{T})} = e^{|i-j|}$. In this study we use a linear distance, which is the straightforward option $d_{ij}^{(\mathcal{T})} = |i - j|$. Time intervals can be not equally spaced, e.g. stock prices are recorded only on weekdays, thus there is a three days gap between the Friday value and the Monday value. In that case $d_{ij}^{(\mathcal{T})} = |\mathcal{T}(i) - \mathcal{T}(j)|$, where $\mathcal{T}$ is the function mapping indexes to actual time values.

Distance in space can have alternative metrics (e.g. Cityblock, Euclidean distances), a discussion of the most common metrics can be found in [2, 95].

We use two terms 'similarity' and 'distance' which are inversely related. The larger is distance the smaller is similarity. We use the term similarity when referring to a general concept and the term distance when referring to actual metric.

4.3.2 Combining the distances in time and feature space

The form of a combination function $\mathcal{D}$ depends on the actual data and domain at hand, as well as on the expectations of a designer. Recall the four assumptions of a concept drift learner design: future assumption, change type, learner adaptivity and model selection, which were discussed in Section 2.2. The model selection choice includes a combination function $\mathcal{D}$, which strongly depends on the observed change type and the future assumption. The dependence is as follows.

The goal is to select the training set in a way that in would represent well the expected future data. Returning to Kate's news Example 1.1.2, if after observing the tendency the designer expects that she will be more and more interested in real estate news, the designer will put more emphasis on selecting real estate related training samples (space similarity). If, on the other hand, the designer expects that a sudden increase in meat prices will persist, the designer will put more emphasis on the recent historical data (time similarity). Thus the choice of time and space combination directly depends on the observed change types and the future expectations.

Let us look at the two boundary cases. If a designer selects training set only based on time similarity, we have a training window case. The most recent instances are selected as a training set in a sequential order. See Figure 4.3 (a) for visualization. Another boundary case is to disregard time distance and select training set only based on distance in space. In Figure 4.3 (b) only space distance based selection is depicted. Note that in both illustrations the toy datasets are the same. The selection strategy strongly depends on the future assumption the designer is using.

We introduced a concept of combining time and space distances for training set selection. In this study we delimit the discussion to linear combination of

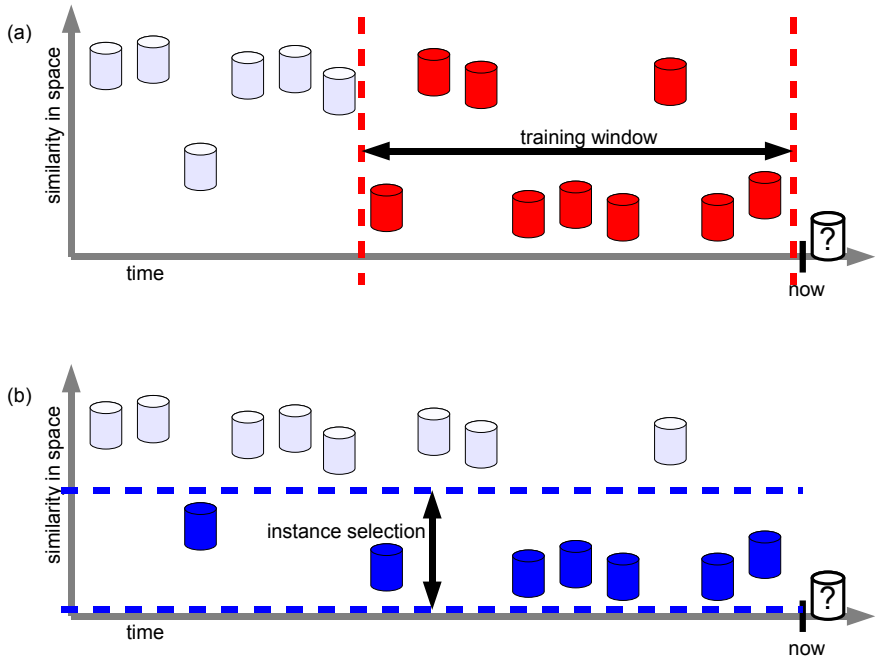


Figure 4.3: Training set selection: (a) based only on similarity in time (training window), (b) only on similarity in the feature space (instance selection).

distances in time and space, the concept is illustrated in Figure 4.4 (a). However, a designer is not limited to linear combination. An example of the second order distance boundary is depicted in Figure 4.4 (b).

The linearly combined similarity between the instances $\mathbf{X}_i$ and $\mathbf{X}_j$ would be

$$\mathcal{D}(\mathbf{X}_i, \mathbf{X}_j) = \alpha_1 d_{ij}^{(S)} + \alpha_2 d_{ij}^{(\mathcal{T})}, \tag{4.2}$$

where α_1 and α_2 are the weight coefficients. If $\alpha_1 = 0$, it is a training window, as in Figure 4.3 (a). If $\alpha_2 = 0$, it is an instance selection, as in Figure 4.3 (b). As a choice of the algorithm design, the weights α_1, α_2 can be fixed based on the domain knowledge or visual inspection of the data or they can be trainable on a validation set.

For interpretation of the balance between the time and space distances across different datasets, $d^{(S)}$ and $d^{(\mathcal{T})}$ need to be normalized. We suggest scaling the values of each feature in $\mathbf{X}$ to form an interval $[0, 1]$. This way we get $d^{(S)} \in [0, p]$, where p is the dimensionality. Then we scale again in order $d^{(S)} \in [0, 1]$. We also scale the time distances so that $d_{ij}^{(\mathcal{T})} \in [0, 1]$. For a single dataset scaling is not

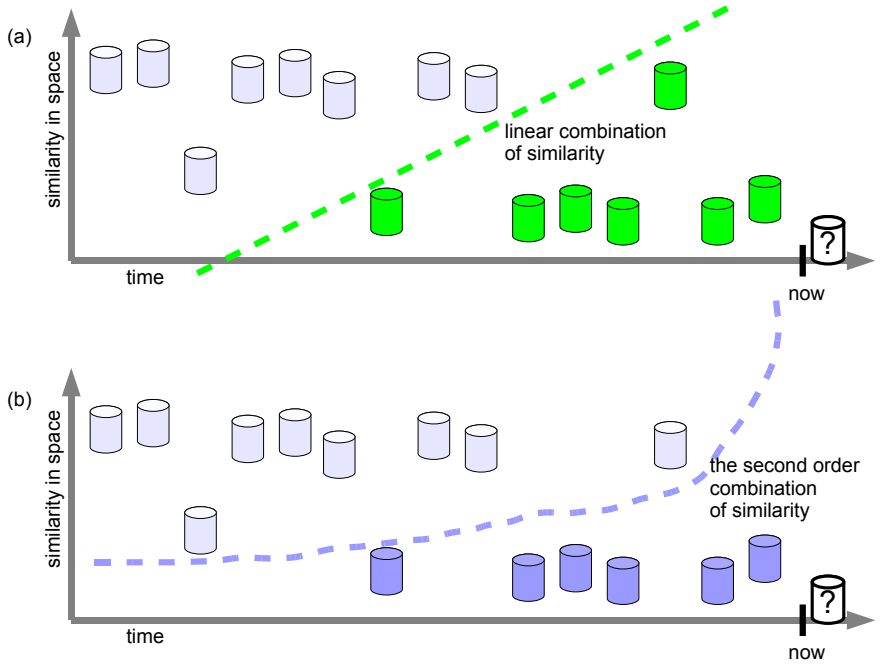


Figure 4.4: Training set selection based on time *and* space: (a) linear combination, (b) the second order combination.

essential, since the balance can be regulated by the weights α_1 and α_2. However, this way time and space distances become comparable across different datasets.

For training set selection under concept drift we are interested in relative distances (ranking). Thus for simplicity α_1 and α_2 can be replaced by $A = \frac{\alpha_2}{\alpha_1}$, assuming that $\alpha_1 \neq 0$. We are interested in the distances between a range of historical instances and the target instance $\mathbf{X}_{t+1}$. Thus, we can simplify Equation 4.2 to

$$\mathcal{D}^*(\mathbf{X}_i, \mathbf{X}_{t+1}) = d^{(S)}_{i,t+1} + Ad^{(T)}_{i,t+1} = \mathcal{D}^*_i. \tag{4.3}$$

4.3.3 The training set size

We defined the similarity in time and space $\mathcal{D}^*$, intended to be used for ranking the historical data $(\mathbf{X}_1, \ldots, \mathbf{X}_t)$ according to the similarity to the target instance $\mathbf{X}_{t+1}$. Another important choice in constructing a training set is where to cut. Or *how many* from the most similar instances to include into a training set? This choice belongs to the forth design assumption (A.4 model selection), which was

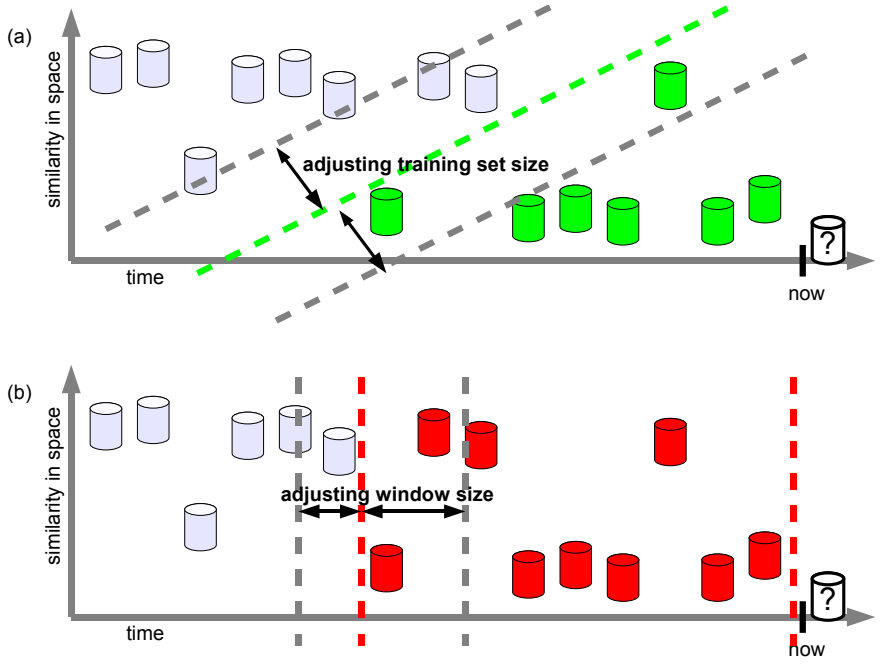


Figure 4.5: Variable training set size: (a) with selection based on time and space, (b) training window.

formulated Section 2.2.

The training set size is specified applying a threshold to the similarity measure. Intuitively, after the similarity measure $\mathcal{D}$ is fixed, the training set size can be decided by moving the decision threshold, as shown in Figure 4.5 (a). Note, that the slope of the line is fixed while it is moved along the diagonal. The slope indicates the proportions of time and space distances in the final similarity measure, as described in Equation (4.2).

The threshold principle is valid for variable window size selection, which is illustrated in Figure 4.5 (b).

More formally, having an unlabeled target instance $\mathbf{X}_{t+1}$, for $i = 1, \ldots, t$ the instance $\mathbf{X}_i$ is selected into a training set if $\mathcal{D}(\mathbf{X}_i, \mathbf{X}_{t+1}) < h^D$, where h^D is the training set threshold. The threshold can be fixed by a designer or trainable based on a validation set.

We presented a framework based on time and space similarity for training set selection for learning under concept drift. The framework generalizes over

adaptive training set selection strategies and maps them. The framework can be extended for an application to instance addition (or instance weighting, e.g. [108, 115]), which is limited to certain types of base classifiers and are out of the scope of this work. Though, instance weights can be viewed as a function of similarity D. In such case every instance selected to form a training set would get a weight [1] $v_i\mathbf{X}_i$, where the weight $v_i \sim f(\mathcal{D}_i^*)$.

4.4 Positioning within the Related Work

After presenting the concept of time and space similarity, in this section we will look what related work is available and how it relates to the approach we are proposing.

We contribute to the field by generalizing training set selection using time and space similarity. To our best knowledge the representation *unifying* windowing and instance selection under concept drift has not been formulated before. There are related works which are implicitly using instance forgetting when employing instance selection strategy, e.g. [131], which is mainly to overcome computational challenges in data streams.

The approach integrates windowing techniques and instance selection techniques under unified framework of systematic training set selection. Moreover, it extends the existing approaches to a combination of both windowing and instance selection.

The issue of systematic training set selection in space under concept drift has been brought up in [18, 76, 103, 133, 199, 201]. Ganti et al. [76] give a generic interpretation of systematic training data selection without real plug-and-play algorithm. The blocks (intervals) of training data can be picked using moving window based templates.

The following two works use space based approaches to training set selection. Tsymbal et al. [199] use an ensemble, where the competence of the base classifiers

[1] not to be mixed with the weights, which are assigned to the output of the learner $\mathcal{L}_t$ in classifier ensembles.

is determined by cross validation on the nearest neighbors of the target instance. However, they use training windows to *build* the individual base classifiers. Katakis et al [103] organize the training data into clusters, derive prototypes for each cluster and then select the clusters for training based on the similarity between the target instance and the prototypes. Since the main focus is on reoccurring contexts, time similarity is not integrated there.

Valizadegan and Tan [201] use intelligent training set selection procedure after a change is detected. They aim to acquire more samples from the regions where classification is unreliable. They call the strategy deferred-boosting and deferred active approach, where deferred means that resampling is triggered by a change detection.

The above approaches limit the history in time from which the instances can be selected. That is an implicit assumption in data stream mining, where the data streams in principle are endless. The approaches overviewed here have a clear cut in history, without incorporating time features into instance selection procedure.

Beringer and Hullermeier [18] organize training data into prototype clusters, referred to as case bases. In contrast to the peer works, they exclude the instances which are too similar to the ones already present in the training set. They explicitly address relevance in time and space, as well as consistency. The major difference in our and their approach is in the future assumption. They assume continuous concept (and call it consistency). Track the concept itself and drop out inconsistent data. The approach would be unfavorable to reoccurring contexts and robust to noise. In our approach we determine the concept for a target instance without tracking the change. This way relevant training set might be found as well in case of reoccurring contexts and even in case of noise.

Lazarescu and Verkantesh [133] use time and space dimensions to determine the relevance of a given historical instance. The idea is closely related to [18] just discussed and has the same limitations regarding following the current concept. The former work is four years later than the latter and worked out in conceptual level and context, while the latter mainly presented the idea.

Adaptive nearest neighbor classification [131, 200] is related to or approach.

Ueno et al [200] focus on the computational complexity issues in streaming kNN application, not on training set selection directly. Their idea is to introduce an order in which the comparison of the distances between the instances is processed, that is likely to give more accurate results than random order, if the comparison is stopped before the end of the historical data is reached. Law and Zaniolo [131] use exponential weighting of the instances in time. They use the grid to divide the space into neighborhood region and adapt only the grids, where the newly arrived instances belong. Building the neighborhood can be viewed as instance selection in space, but their approach is kNN classifier specific. The griding mechanism is explicitly oriented towards forming a single class cells, while generalization to different base classifiers would require an opposite strategy.

Finally, Black and Hickey [26] use the idea of augmenting the feature space by adding a time stamp feature. Then they use training window approaches, thus they do not employ space based selection. In principle the time feature can be integrated with space based instance selection. Augmenting the feature space and then measuring distances in the new space can be more flexible with respect to base classifier related adaptivity, than the combination we are taking. For instance, time related splits in a decision tree can be organized. We take the latter direction, mainly because the explicit combination of distances in time and space for training set selection can be easier to control and interpret.

We propose an approach for training set selection based on similarity to target instance. There are multiple classifier methods (e.g. [199]) employing similarity aspect but only in the classifier selection phase. However, the needed classifier might not be present among the ensemble members. From similarity in space perspective our approach is related to a lazy learning [9], but the main difference is that the latter does not construct explicit generalization and makes classification based on direct comparison of the target and training instances. Our approach can be used as a wrapper with different base classifiers. The closest approaches [18, 133] try to follow the concept changes, thus are not straightforward to generalize for reoccurring contexts.

4.5 FISH Algorithm Family

To support our approach (combining time and space similarity), we propose a family of algorithms called FISH (uni**F**ied **I**nstance **S**election algorit**H**m), which incorporate the presented approach. The algorithms systematically select training instances. Similarity in time and space is addressed. The method can be used with different base classifiers.

The family includes three modifications of the algorithm: FISH1, FISH2 and FISH3. In FISH1 the size of a training set is fixed and set in advance, the extension FISH2 operates using variable training set size. The size is determined at each time step (for each target instance) using cross validation on the historical data. In FISH2 the proportion of time and space distances (α_1 and α_2 in Equation 4.2) in the final similarity measure are fixed in advance as a design choice. We present a modification FISH3, where these proportions are trainable. They are determined by a cross validation at each time step.

We consider FISH2 to be the central algorithm in the family. We believe that in many cases the optimal proportions of time and space similarity are domain dependent and can be fixed offline (for instance, using an offline validation set). On the other hand, our intuition suggests that the drifts often might take non uniform speeds, thus online adjustable training set size is relevant.

4.5.1 The design choices

In Section 2.2 we formulated the four main design assumptions used when designing concept drift learners. The design choices for the FISH family are as follows.

A.1 Future assumption: estimation based on unlabeled $\mathbf{X}_{t+1}$. FISH selects training instances based on their similarity to the target observation.

A.2 Change type: gradual drift, generalizable to sudden drift and reoccurring contexts.

A.3 Learner adaptivity: training instance selection.

A.4 Model evaluation and selection: online cross validation.

Next we give pseudo code and explain the details and intuition for each of the three FISH algorithms.

4.5.2 The proposed algorithms

We present the algorithms as one time step of the sequential learning process, that means what would we do to predict a label for a given target instance $\mathbf{X}_{t+1}$. The process was described in Figure 2.2, we are now presenting an option for the 'black box'.

4.5.2.1 FISH1

We start with presenting FISH1 in Figure 4.6.

THE TRAINING SET SELECTION ALGORITHM (FISH2)

INPUT
Data: historical instances $\mathbf{X^H} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ with labels $\mathbf{y^H}$, target instance $\mathbf{X}_{t+1}$ without a label.
Parameters: training set size N, time/space similarity weight A (Equation (4.3)).
Base learner type: $\mathcal{L}$.
ALGORITHM

1. Calculate distances in time and space $\mathcal{D}_i^*$ (Equation (4.3)) *for* $i = 1 : t$.
2. Sort the distances from minimum to maximum $\mathcal{D}_{z1}^* < \mathcal{D}_{z2}^* < \ldots < \mathcal{D}_{zt}^*$. Indexes $z1, \ldots, zt$ define the permutation $(\mathbf{X}_1, \ldots, \mathbf{X}_t) \rightarrow (\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zt})$.
3. Pick N instances having the smallest distances $\mathcal{D}$.
4. Output the indexes $\{z1, \ldots, zN\}$.

OUTPUT
The indexes $\mathcal{I}_t = \{z1, \ldots, zN\}$ to form a training set $\mathbf{X_t^T} = (\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zN})$.

Figure 4.6: FISH1 algorithm for fixed size training set.

The algorithm ranks the historical instances (without their labels) according to their similarity to the target instance and picks N the most similar instances to form a training set. Since the size of the training set is fixed, if it happens to be to small, the instances from a single class might end up in a training set. To insure from this, we suggest selecting a stratified training set. It means that $\frac{N}{c}$ most similar instances are selected from each class, altogether forming a training set of size N.

4.5.2.2 FISH2

FISH1 uses fixed training set size N. FISH2 is an extension, where the training set size is learnable online. To implement a variable training size we incorporated the principles inspired by two windowing algorithms [110] (KLI) and [199] (TSY). FISH2 is presented in Figure 4.7.

We start by calculating the distances in time and space between the target instance $\mathbf{X}_{t+1}$ and every historical instance in $\mathbf{X^H}$ as in Equation (4.3). The the distances to each historical instance are sorted from minimum to maximum, as we would like to form a training set using the most similar (the closest) instances.

Next we decide *how many* of the most similar training instances to pick. For that we build a set of classifiers $(\mathcal{L}^1, \mathcal{L}^2, \ldots, \mathcal{L}^N)$ using different cut offs (different training set sizes). We test those classifiers using a validation set. The validation is formed using k historical instances, which were found to be the most similar to the target instance $\mathbf{X}_{t+1}$. We select the training size N^*, which has given the best accuracy on the validation set. The method works similarly to windowing in [110] (KLI). They use sequential instances in time to form the windows. We use combined time and space similarity.

Leave-one-out cross validation needs to be employed when using the validation set. It means that we repeat the validation process k times for every training set size N being checked. Each time we leave out one validation instance from the training set and then test on it. Without cross validation the training set of size k is likely to give the best accuracy, because in that case training set would be equal to the validation set.

THE TRAINING SET SELECTION ALGORITHM (FISH2)

INPUT
Data: $\mathbf{X^H}$, $\mathbf{y^H}$, $\mathbf{X}_{t+1}$.
Parameters: neighborhood size k, A . Base learner type: $\mathcal{L}$.
ALGORITHM

1. Calculate distances in time and space $\mathcal{D}_i^*$ (Equation (4.3)) *for* $i = 1 : t$.
2. Sort the distances from minimum to maximum $\mathcal{D}_{z1}^* < \mathcal{D}_{z2}^* < \ldots < \mathcal{D}_{zt}^*$.
3. *For* $N = k : step : t$ select the training set size
   (a) pick N instances having the smallest distances $\mathcal{D}$,
   (b) using cross-validation[a] build a classifier $\mathcal{L}^N$ using the instances $(\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zN})$ as the training set,
   (c) test $\mathcal{L}^N$ on the k nearest neighbors $(\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zk})$, record testing error e_N.
4. Find the minimum error classifier $\mathcal{L}^{N*}$, where $N^* = \arg\min_{N=k}^{t}(e_N)$.
5. Output the indexes $\{z1, \ldots, zN^*\}$.

OUTPUT
The indexes $\mathcal{I}_t = \{z1, \ldots, zN^*\}$ to form a training set $\mathbf{X_t^T}$.

[a] when test on the instance $\mathbf{X}_{zk}$, this instance is excluded from the validation set

Figure 4.7: FISH2 algorithm for variable training set selection.

The outcome of the algorithm is a set of N^* indexes $\mathcal{I}_t = \{z1, \ldots, zN^*\}$. These indexes indicate, which historical instances need to be picked as a training set $\mathbf{X_t^T} = (\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zN})$. Using the original instances $\mathbf{X_t^T}$ a classifier $\mathcal{L}_{\sqcup}^{\mathcal{N}*}$ is trained for the final prediction of the label y_{t+1} for the target instance $\mathbf{X}_{t+1}$. The method can be used plugging in various types of a base classifier $\mathcal{L}$.

FISH2 uses learnable training set size N^*, while FISH1 used prefixed size N.

4.5.2.3 FISH3

FISH3 is a extension of FISH2. FISH2 used a prefixed balance between time and space similarity A. FISH3 can learn the balance *online* using an additional

loop of cross validation. FISH3 is presented in Figure 4.8. Instead of fixing the proportion between time and space distances in $\mathcal{D}$ in Equation (4.2) we try a number of options and pick the learner which is the most accurate on the validation set, the same principle as in FISH2.

THE TRAINING SET SELECTION ALGORITHM (FISH2)

INPUT
Data: $\mathbf{X^H}$, $\mathbf{y^H}$, $\mathbf{X}_{t+1}$. Parameters: k. Base learner type: $\mathcal{L}$.
ALGORITHM

- *For* $j = 0 : step : 1$, $\alpha_1 = j$ $\alpha_2 = 1 - \alpha_1$ every time and space proportion
  1. calculate distances $\mathcal{D}_i^j = \alpha_1 d_{i,t+1}^{(S)} + \alpha_2 d_{i,t+1}^{(\mathcal{T})}$ (Equation (4.2)) *for* $i = 1 : t$.
  2. Sort the distances from minimum to maximum $\mathcal{D}_{jz1}^j < \mathcal{D}_{jz2}^j < \ldots < \mathcal{D}_{jzt}^j$.
  3. *For* $N = k : step2 : t$ select the training set size
     (a) pick N instances having the smallest distances $\mathcal{D}^j$,
     (b) using cross-validation[a] build a classifier $\mathcal{L}^{jN}$ using the instances $(\mathbf{X}_{jz1}, \ldots, \mathbf{X}_{jzN})$ as the training set,
     (c) test $\mathcal{L}^{jN}$ on the k nearest neighbors $(\mathbf{X}_{jz1}, \ldots, \mathbf{X}_{jzk})$, record testing error e_N^j.
- Find the minimum error classifier $\mathcal{L}^{jN*}$, where $jN^* = \arg\min_{j=0}^{1} \min_{N=k}^{t}(e_N^j)$.
- Output the indexes $\{jz1, \ldots, jzN^*\}$.

OUTPUT
The indexes $\mathcal{I}_t = \{jz1, \ldots, jzN^*\}$ to form a training set $\mathbf{X_t^T}$.

[a] when test on the instance $\mathbf{X}_{jzk}$, this instance is excluded from the validation set

Figure 4.8: FISH3 algorithm with learnable training set size and distance proportion.

4.6 Experimental Evaluation

In order to verify the properties of FISH algorithms we carry out extensive numerical experiments.

The main goal is to illustrate the advantage of combining similarity in time and space as compared to only time or only space similarity. We implement two peer algorithms and run them in parallel to FISH on six datasets. In order to minimize the bias of base classifier selection we run the experiments using four different base classifiers. We test the algorithms using two alternative distance in space measures.

4.6.1 The datasets

We use six data sets with potential gradual drift. Three datasets are real (Luxembourg, Ozone, Electricity), three other are real with artificially introduced drift (German, Vote2, Iono2). The expectations of the drift are related to the domain of the real data and the way artificial drift is introduced. The characteristics of the datasets are summarized in Table 4.1, more details can be found in Appendix C.

Table 4.1: Summary of the used datasets (all data is real).

Name	Dimensions	Size	Class	Type of drift
Luxembourg	31	1901	0.51:0.49	real
Ozone	72	2534	0.94:0.06	real
Electricity	6	2956	0.57:0.43	real
German	23	1000	0.70:0.30	simulated
Vote2	16	435	0.61:0.39	simulated
Iono2	43	435	0.61:0.39	simulated

We constructed Luxembourg dataset using social survey data. Each instance is a person. The task is to predict the if he or she is a heavy internet user. The task is relevant for marketing purposes. Ozone dataset consists of air measurements, the task is to predict ozone level eight hours ahead. Electricity data characterizes electricity demand in Australia, the task is to predict electricity market price.

German credit data consists of individual credit application records, the task is to predict bankruptcy. We introduced artificial drift by hiding the age feature. We do not introduce any synthetic drift in Iono and Vote datasets. However, we refer to the drift as artificial, since we assume the data is presented in a time

order, although it is not explicitly stated. How can we claim that there is a drift? If selective sampling gives better classification accuracy than a growing window in sequential learning process, this can be treated as the drift evidence.

In Figure 4.9 we visualize the six datasets on time and space axes. Note that the distances to one data point are visualized, which is rather a snapshot than a representation of the whole dataset. For illustration we use cosine distances in space.

4.6.2 Experimental scenario

We perform three series of experiments related to FISH1, FISH2 and FISH3 algorithms correspondingly.

4.6.2.1 FISH1 and fixed training set size

First we run controlled FISH1 experiments. We vary the proportion of time and space similarity to analyze the effect to the accuracy. We use fixed training set size N.The extreme $\alpha_1 = 0$ corresponds to a fixed training window. Contrary, $\alpha_2 = 0$ corresponds to only space based similarity.

We include a baseline ALL, which is using all the historical data as the training set. Thus it does not select training data, every time step the training set is growing. The classifier is retrained using all the past data. If the data happens to be stationary, ALL should be the most accurate.

The pseudo code for ALL is provided in Appendix B.

4.6.2.2 FISH2 and variable training set size

We present FIHS2 as a flagman in the FISH family and perform extensive experimental evaluation for it. We test FISH2 plugging in four alternative base classifiers: a parametric Nearest Mean classifier (NMC), non-parametric k Nearest Neighbors classifier (kNN) (for which we take $k = 7$), Parzen Window

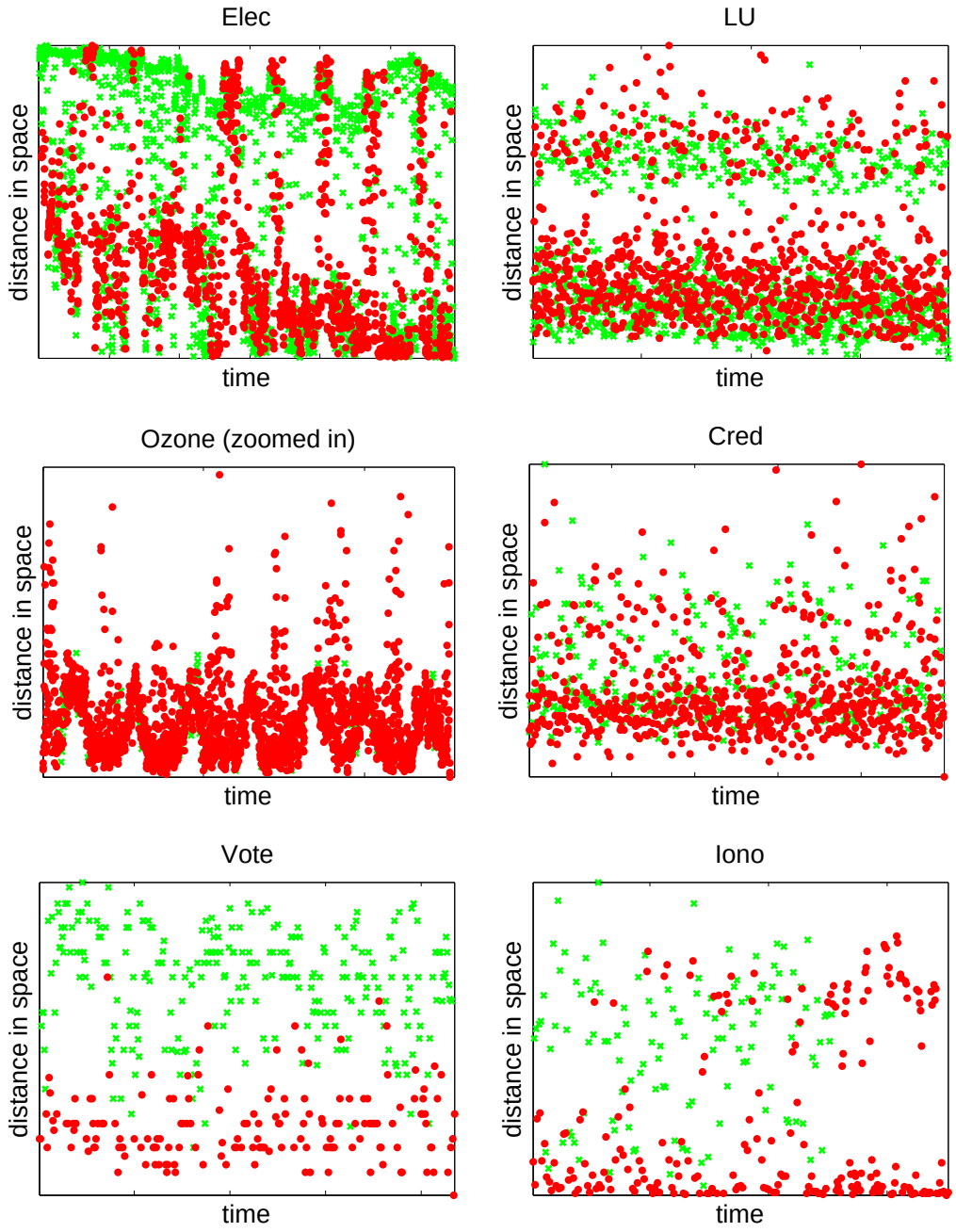


Figure 4.9: Visualization of the used real datasets.

classifier (PWC) and not pruned decision tree (tree), see e.g. [58] for details. In addition to different base classifiers, we run the tests using two alternative distance in space measures: the Euclidean distance and cosine (the details will follow in the next Section).

To support the viability of FISH2, we implement and run two peer algorithms for training set selection under concept drift: Klinkenberg and Joachims [110] (KLI) and Tsymbal et al. [199] (TSY). Both algorithms are not specific to a particular base classifier. KLI algorithm tries out a set of different training windows and selects the one which shows the best accuracy on the validation data. The most recent training data is chosen as a validation set. TSY algorithm builds a number of classifiers on different consecutive training subsets. The final classifier is also selected based on the performance on a validation set. Contrary to a time criterion, used in KLI, the latter algorithm employs similarity in space (to the target instance) criterion to select a validation set. Both algorithms use windowing to form the individual classifiers. In contrast, FISH builds individual classifiers using systematic instance selection based on time and space similarity. The summary of KLI and TSY with the options and interpretations chosen are presented in Appendix B Figures B.3 and B.4 respectively.

The motivation the motivation for choosing this peer group is to purify and highlight the effect of integrated instance selection (time and space) which is done in FISH. The chosen algorithms are able to determine straining set size using cross validation, they use no explicit change detection and are base classifier independent and they do not require complex parametrization.

We also include a baseline ALL, which uses all the historical data, which is available up to time $t + 1$. If the data happens to be stationary, ALL should be the most accurate.

4.6.2.3 FISH3 and variable time and space ratio

Finally, we compare the performances of FISH1,FISH2 and FISH3 to see what benefits in accuracy are brought by the increasing computational complexity (from FISH1 to FISH3).

We also analyze the progresses of the training set size and the proportion of time and space similarity in time.

4.6.3 Implementation details

For FISH, FISH2, FISH3 and TSY we use the Euclidean distance in space, which is

$$d^E(\mathbf{X}_j, \mathbf{X}_l) = \sqrt{\sum_{i=1}^{p} |\mathbf{x}_j^{(i)} - \mathbf{x}_l^{(i)}|^2}, \tag{4.4}$$

here $\mathbf{x}_j^{(i)}$ is the i^{th} feature of the instance $\mathbf{X}_j$ and p is the dimensionality.

We also test FISH2 using cosine distance in space, which is

$$d^C(\mathbf{X}_j, \mathbf{X}_l) = \cos(\mathbf{X}_j, \mathbf{X}_l) = \frac{\sum_{i=1}^{p} \mathbf{x}_j^{(i)} \mathbf{x}_l^{(i)}}{\sqrt{\sum_{i=1}^{p} (\mathbf{x}_j^{(i)})^2} \sqrt{\sum_{i=1}^{p} (\mathbf{x}_l^{(i)})^2}}. \tag{4.5}$$

The features are scaled to the interval $[0, 1]$ before calculating the distance in space. We use linear distance in time, as defined in Equation (4.1). Distances in time and space are scaled [2] to $d^{(S)}, d^{(\mathcal{T})} \in [0, 1]$ before calculating the proportion $\alpha_1 : \alpha_2$.

We use the following setting for the algorithms.

For FISH1 and TSY we use training set size $N = 40$, FISH2, FISH3 and KLI has adaptable set size, ALL has a growing set size.

KLI and TSY operate in batch mode, we use batch size 15 for both.

For TSY we use maximum ensemble size $= 7$, number of the nearest neighbors $= 7$ for error estimation.

The weights used for fixed time and space similarity for FISH1 and FISH2 were $\alpha_1 : \alpha_2 = 1 : 1$ for all the data. In the first series of experiments we used variable ration of $\alpha_1 : \alpha_2$.

[2] In [226] only features were scaled, thus the distance in space was in the interval $[0, p]$

For FISH2 and FISH3 we took training set sizes for cross validation with a step 5 to speed up the experiments [3]

If there are too few instances from one class in a formed sample, the label is assigned according to the major class [4].

For Ozone, Elec and LU data backward search for FISH2, FISH3, KLI and TSY was limited to 1000 instances to reduce the complexity of the experiment. For testing with the decision tree using Elec and Ozone data we subsampled taking every 5^{th} instance to speed up the experiments.

4.7 Algorithm Evaluation

We evaluate FISH2 performance based on the *testing error* and *complexity*. We also analyze the progress of the experiments to draw qualitative conclusions.

To evaluate the accuracy, we calculate the ranks of the peer methods. The best method for a given data set is ranked 1, the worst method is ranked 4. The ranks for each data set sum to 10. An average rank over all the datasets is calculated for each classifier and used as performance measure.

In order to estimate a statistical significance of the differences between the error rates of the methods, for real datasets we used the McNemar [123] paired test, which does not require assumption about i.i.d. origin of the data.

To evaluate the applicability, we calculate the worst case and the average complexity of the six peer methods. We count the number of data passes required to make a classification decision for one observation at time t. The results (approximations) are presented in Table 4.2. Granularity g is a step of the time and space proportion [5]. We also present the parameters that needed to be prefixed in advance for each algorithm.

[3] In [226] we used step of 2.

[4] In [226] it was assigned at random.

[5] the number of options tried out, we use 10: option 1 $\alpha_1 = 0, \alpha_2 = 1$, option 2 $\alpha_1 = 0.1, \alpha_2 = 0.9, \ldots$, option 10 $\alpha_1 = 1, \alpha_2 = 0$

Table 4.2: Algorithm complexities. b - batch size; M - ensemble size; k - testing neighborhood size; N - training set size; A - time/space weight; g - granularity of the time and space proportion; t - time since the start of the sequence.

Method	Worst case	Average	Hyperparameters
ALL	t	the same	$-$
KLI	$\frac{t^2}{2}+\frac{tb}{2}$	$\frac{t^2}{2b}+\frac{t}{2}$	b
TSY	$t(k+1)+N$	$t(k+1)+\frac{N}{b}$	b,M,k,N
FISH1	$t(N+2)+\frac{N(2-N)}{4}$	the same	A,N
FISH2	$\frac{t^2k}{2}+\frac{t(k+2)}{2}$	the same	A, k
FISH3	$g(\frac{t^2k}{2}+\frac{t(k+2)}{2})$	the same	k

The run time of FISH2 is reasonable for sequential data, for all five algorithms it takes up to 1 min for NMC, kNN and PWC and for the decision tree it is ~ 5 times longer to cast a classification decision for *one time observation* on a 1.46 GHz PC, 1GB RAM. For implementation MATLAB 7.5 is used.

Finance, biomedical applications are the domains where the data is scarce, imbalanced, while concept drift is very relevant. For example, in bankruptcy prediction an observation might be received once per day or even per week, while the model needs to be constantly updated and economic cycles imply concept drift. In supermarket stock management, stock quantity needs to be predicted once per week, thus only 52 observations are received per year. In such application cases even one hour of the algorithm run for the decision would not be an issue.

4.8 Results and Discussion

In this section we present and discuss experimental results. We start with FISH1.

4.8.1 FISH1 results

We run controlled experiments with FISH1 varying the proportion between time and space similarity. By controlled we mean that we fix the setting except

one parameter, which is the balance between time and space similarity. We investigate the effect of the balance to the testing accuracy on the six real dataset. We allow the balance $\alpha_1 : \alpha_2$ change from $0 : 1$ to $1 : 0$ with a step 0.01.

We use NMC as the base classifier to simplify the setup as much as possible to analyze the effect of time and space balance to the testing accuracy.

The testing results for each of the six datasets are provided in Figure 4.10. We plot the testing accuracy against the balance of time and space similarity. $\alpha_1 = 0$ means that only time similarity is used, which corresponds to a training window of a fixed size. $\alpha_1 = 1$ (implies $\alpha_2 = 0$) means that only similarity in space is used. All the values in between indicate different balance between time and space similarity in training set selection.

In Table 4.3 we provide the numerical results using a step 0.1 for α_1 values. Although the primary purpose of the experiment is to analyze the relation between the balance in time and space and accuracy, we also indicate statistical significance of the difference between FISH1 and the benchmark ALL using McNemar test.

The results both in Figure 4.10 and in the table show that even using a primitive fixed size technique the best accuracy is achieved in combination of time and space similarity. The minimum error is heavily shifted towards space similarity in most of the datasets ('Ozon','Cred','Iono' and 'Luxe'), which is explainable by data selection procedure. The datasets were picked expecting heterogeneous structure in space and also to have a temporal order. 'Luxe' data has it's minimum testing error at the very extreme space similarity (time similarity is 0), which we could observe in the data plot in Figure 4.9. Visually, the time order in 'Luxe' data is weak. On the contrary, 'Elec' data has visually strong time order and that correlates with the testing results. We can see in Figure 4.10 that 'the minimum testing error for 'Elec' is close to the time similarity extreme. It suggests the training based on windowing would be preferable.

The dashed lines in Figure 4.10 indicate the baseline testing error, which is achieved by using full history as a training set (ALL). The primitive FISH1 using a fixed training size already outperforms ALL in five out of six datasets. Absolute error in 'Ozon' is so high since the classes are heavily unbalanced

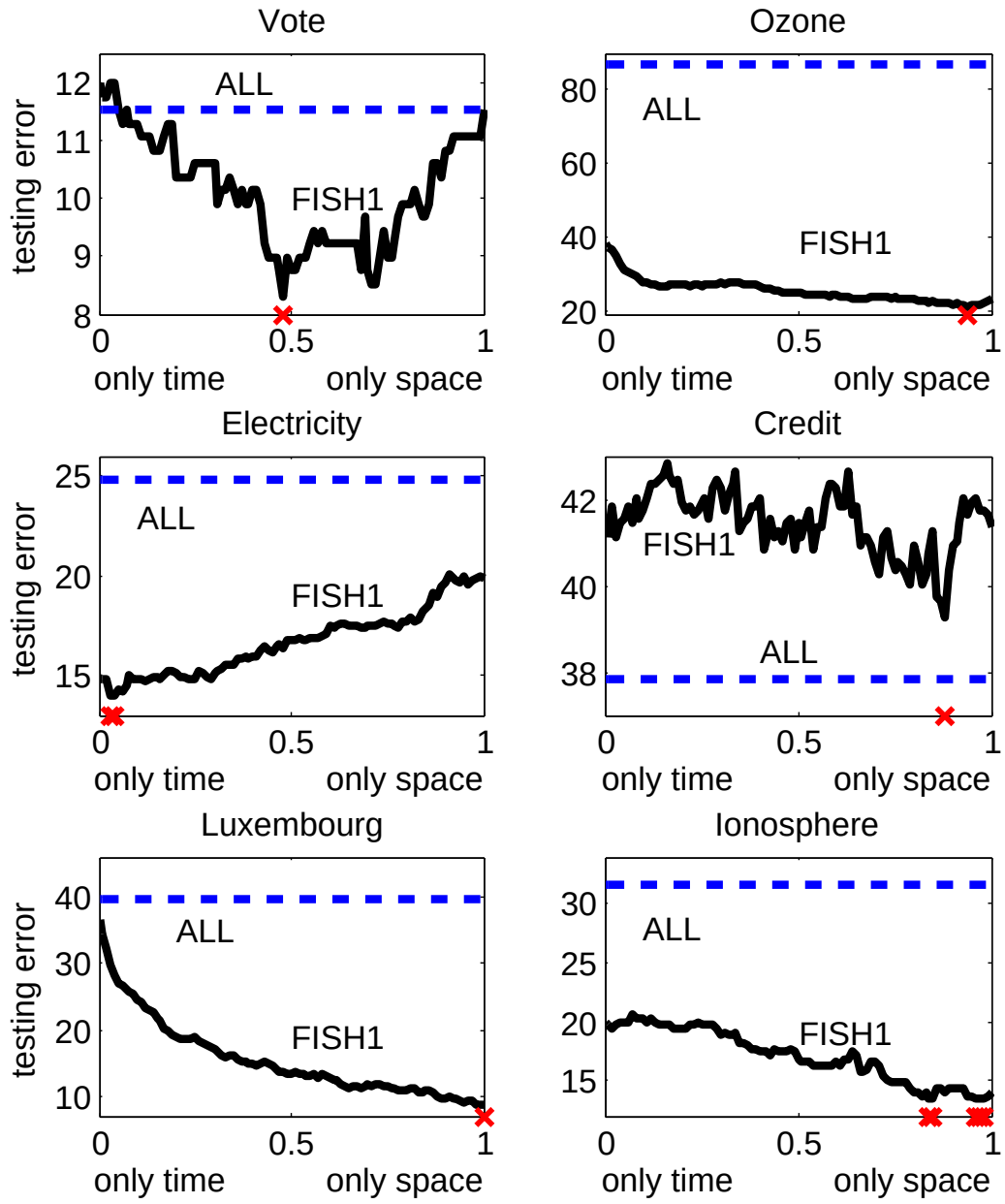


Figure 4.10: FISH1: testing errors. '×' symbol denotes minimum error.

(major class makes 94%). In 'Cred' FISH1 is worse than ALL in all the time and space proportions. It suggests that either the data is stationary, or the fixed size of the training set is very much non optimal. The training set size was chosen to be equal for all the data sets to keep the settings uniform and the results comparable. Let us look, what the results will be, when we introduce training

Table 4.3: Testing errors. The best accuracy for each column is underlined. Symbol '•' indicates that the algorithm performed significantly better than ALL, '∘' indicates that the algorithm performed significantly worse than ALL, and '−' indicates no difference (at $\alpha = 0.05$).

	weight							
	space	**time**	**Luxe**	**Ozon**	**Elec**	**Cred**	**Vote**	**Iono**
	α_1	α_2						
FISH1	0	1	36.53•	38.29•	14.86•	41.54∘	11.98−	20.00•
FISH1	0.1	0.9	24.42•	27.75•	<u>14.75</u>•	41.74∘	11.29−	20.29•
FISH1	0.2	0.8	19.00•	26.92•	15.13•	41.94∘	10.37−	19.43•
FISH1	0.3	0.7	17.21•	27.48•	15.13•	42.24∘	10.60−	18.86•
FISH1	0.4	0.6	14.84•	26.29•	15.91•	42.04∘	10.14−	17.43•
FISH1	0.5	0.5	13.42•	24.91•	16.75•	41.44∘	<u>8.76</u>•	16.86•
FISH1	0.6	0.4	12.79•	24.04•	17.46•	42.24∘	9.22−	16.57•
FISH1	0.7	0.3	11.74•	23.69•	17.53•	40.54∘	<u>8.76</u>−	16.57•
FISH1	0.8	0.2	10.95•	23.25•	17.70•	40.94−	9.91−	<u>14.00</u>•
FISH1	0.9	0.1	9.53•	<u>21.71</u>•	19.66•	40.94−	10.83−	14.29•
FISH1	1	0	<u>8.58</u>•	23.02•	19.83•	41.34∘	11.52−	<u>14.00</u>•
ALL			39.68	86.70	24.84	<u>37.84</u>	11.52	31.71

size to be learnable online.

4.8.2 FISH2 results

We test FISH2 along with two peer algorithms KLI and TSY as well as the baseline ALL. We test using four alternative base classifiers and two alternative distance in space measures for the six datasets. Thus all in all we run $4 \times 2 \times 6 = 48$ experiments for each of the algorithms. The results are provided in Tables 4.4 and 4.5. We use McNemar paired test to estimate the statistical significance of the difference between FISH2 and the peers.

The five methods were ranked as presented in Section 4.7 with respect to each data set, and the ranks were then averaged (last column in Table 4.4).

FISH2 has the best rank by a large margin with NMC and tree classifiers, for kNN and PWC either FISH2 or ALL prevails, depending on the distance measure. The final scores averaged over all four base classifiers and two alternative distance

Table 4.4: FISH2: testing errors, Euclidean distance in space. The best accuracy for each column is underlined. Symbol '•' indicates that the algorithm performed significantly worse than FISH2, '○' indicates that the algorithm performed significantly better than FISH2, and '−' indicates no difference (at $\alpha = 0.05$).

	base	Luxe	Ozon	Elec	Cred	Vote	Iono	RANK
FISH2	NMC	11.89	34.31	15.16	36.94	8.53	17.43	*1.33*
KLI		30.89•	22.90○	19.97•	36.24−	11.29•	21.71•	*2.08*
TSY		35.89•	37.23•	15.47−	40.64•	11.29•	20.57−	*2.75*
ALL		39.68•	86.70•	24.84•	37.84−	11.52•	31.71•	*3.83*
FISH2	kNN	14.63	7.03	15.06	30.13	8.76	22.00	*2.08*
KLI		15.74−	7.11−	18.98•	30.03−	9.68−	22.86−	*3.00*
TSY		28.79•	7.03−	13.16○	31.43−	10.60−	23.14−	*3.25*
ALL		11.84○	6.99−	19.86•	28.83−	8.29−	22.29−	*1.67*
FISH2	PWC	12.37	70.79	41.08	34.33	8.99	12.86	*1.75*
KLI		14.42•	38.81○	46.06•	34.63−	10.37−	15.14•	*3.00*
TSY		26.42•	54.72○	43.62•	36.54−	9.68−	19.71•	*3.25*
ALL		11.68−	84.88•	43.62•	34.43−	8.53−	12.86−	*2.00*
FISH2	tree	0.37	9.99	13.54	31.03	7.37	18.00	*1.42*
KLI		0.37−	11.69•	17.36•	36.34•	9.68−	20.86−	*3.25*
TSY		0.37−	12.63•	8.97○	37.04•	10.14−	20.29−	*3.08*
ALL		0.37−	10.50−	16.99•	32.83−	7.83−	18.57−	*2.25*

measures are: 1.68 for FISH2, 2.83 for KLI, 3.07 for TSY and 2.42 for ALL.

Using kNN, PWC and tree as a base classifier, ALL method outperform TSY and KLI according to the rank score. It implies, that under this setting there would be little point in employing those concept drift responsive methods and increasing complexity, as simple retraining (ALL) would do well. The results in favor of FISH2 are significant in about half of the cases. Some of the results indicate no statistical difference, the datasets are not large.

FISH2 method is designed to work where concept drift is not clearly expressed. These are the situations of gradual drift, reoccurring contexts. ALL method outperforms all the drift responsive methods but not FISH2 with kNN as a base classifier.

Windowing methods work well on Elec data, because the drifts in this data are more sudden. Elec data shows the biggest need for concept drift adaptive

Table 4.5: FISH2: testing errors, cosine distance in space. The best accuracy for each column is underlined. Symbol '•' indicates that the algorithm performed significantly worse than FISH2, '∘' indicates that the algorithm performed significantly better than FISH2, and '−' indicates no difference (at $\alpha = 0.05$).

	base	Luxe	Ozon	Elec	Cred	Vote	Iono	RANK
FISH2	NMC	12.68	35.25	15.57	38.14	8.76	16.86	*1.67*
KLI		30.89•	22.90∘	19.97•	36.24−	11.29•	21.71•	*2.08*
TSY		35.89•	37.23−	15.47−	40.64−	11.29•	20.57−	*2.58*
ALL		39.68•	86.70•	24.84•	37.84−	11.52•	31.71•	*3.67*
FISH2	kNN	14.79	6.99	15.19	29.93	8.53	21.71	*1.75*
KLI		15.74−	7.11−	18.98•	30.03−	9.68−	22.86−	*3.17*
TSY		28.79•	7.03−	13.16∘	31.43−	10.60•	23.14−	*3.33*
ALL		11.84∘	6.99−	19.86•	28.83−	8.29−	22.29−	*1.75*
FISH2	PWC	12.68	72.25	39.26	34.83	8.29	13.34	*2.00*
KLI		14.42•	38.81∘	46.06•	34.63−	10.37•	15.14−	*2.83*
TSY		26.42•	54.72∘	43.62•	36.54−	9.68−	19.71•	*3.25*
ALL		11.68∘	84.88•	43.62•	34.43−	8.53−	12.86−	*1.92*
FISH2	tree	0.37	10.03	12.79	31.34	7.60−	17.71	*1.71*
KLI		0.37−	11.69•	17.36•	36.34•	9.68−	20.86−	*2.83*
TSY		0.37−	12.63•	8.97∘	37.04•	10.14−	20.29−	*3.06*
ALL		0.37−	10.50−	16.99•	32.83−	7.83−	18.59−	*2.40*

methods, because for these datasets ALL method performs relatively the worst from the peer group.

The credits for FISH2 performance in the peer group shall be given to similarity based training set selection. KLI addresses only similarity in time (training window). TSY uses only similarity in time for classifier training, but then they use similarity in space for classifier selection. We employ a combination of time and space similarity already in classifier building phase.

Why we outperform KLI and TSY even in time minimum? Because as compared to KLI we use adaptive validation set, as compared to TSY we use variable training set size.

In FISH2 experiments we fixed equal share of time and space similarity $\alpha_1 : \alpha_2 = 1 : 1$, in order to have uniform comparable setups for all the datasets. Let us look if we can improve FISH2 results by allowing the time and space proportion to be learnable online.

Table 4.6: FISH3: variable proportion of time and space. The best accuracy for each column is underlined.

	Luxe	**Ozon**	**Elec**	**Cred**	**Vote**	**Iono**
FISH1	14.63	<u>26.49</u>	15.74	41.44	8.76	16.86
FISH2	11.89	34.31	15.16	36.94	<u>8.53</u>	17.43
FISH3	<u>10.53</u>	37.47	<u>13.77</u>	<u>36.34</u>	<u>8.53</u>	<u>15.43</u>
mean α_1	0.77	0.62	0.41	0.46	0.32	0.52
ALL	39.68	86.70	24.84	37.84	11.52	31.71

4.8.3 FISH3 results

FISH3 algorithm has variable training set size and variable time and space similarity proportion, both are learnable online. Recall, that we had different time and space proportions in FISH1 experiments. However, in FISH1 the balance fixed for all the experiment. In FISH3 we have a variable balance for every time step and it is learnable online.

In Table 4.6 we compare the accuracies of the three FISH algorithms using simple settings: NMC classifier and Euclidean distance in space. We use the same fixed proportion of time and space as before for both FISH1 and FISH2 ($\alpha_1 : \alpha_2 = 1 : 1$).

FISH3 has the best accuracy in all cases except for Ozone data, which is very highly imbalanced. We include a benchmark 'ALL' to verify if our concept drift responsive methods make sense. The differences between 'ALL' and FISH accuracies are statistically significant everywhere except in Credit data.

It might be argued that improvement in accuracy shown by FISH3 as compared to FISH2 is marginal. In fact the differences between FISH2 and FISH3 are statistically significant in three out of six datasets (Luxembourg, Ozone and Electricity) which are more than twice longer than the remaining ones, besides they have a natural temporal order, while the remaining three have assumed temporal order.

Let us look at the time and space proportion. In Table 4.6 we provide averaged space proportion (mean α_1). Luxembourg and Ozone datasets are inclined towards space similarity, while Vote and Electricity data shows inclination

towards time similarity. That is not fully consistent with the observations in FISH1 experiments, Section 4.8.1. Note that in FISH1 experiments the time and space proportion was fixed for all the run on a dataset, while here we allowed the proportion to vary every time step. In Figure 4.11 we plot the progress of the time and space proportion for all six datasets. The line is smoothed using a moving average of 5 to emphasize the tendencies against individual peaks.

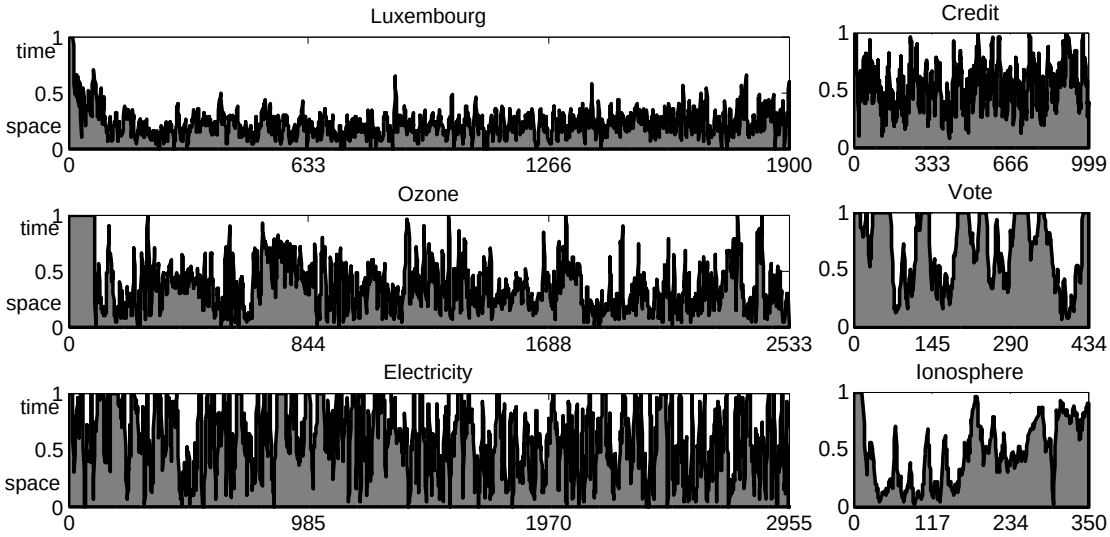


Figure 4.11: Progress of the time and space proportions in FISH3.

It can be concluded that if the domain allows increased complexity variable training set size and variable time/space proportions are worth applying to gradually drifting datasets.

4.8.4 Discussion

The FISH family of algorithms should be regarded not as a competitor but as an extension to existing techniques. It emphasizes that time and space relations are not discrete, but can be viewed in a continuous space.

4.9 Conclusion

In this chapter we addressed the training set selection problem under gradual concept drift, which happens when one data generating source is gradually taken over by another and for some time period both sources are active. We demonstrated that under these settings training window strategy in time is not enough, in addition historical data needs to be filtered in the feature space in order to select a relevant training set, since the data from different sources is mixed together.

The second research question (RQ2) consisted of two parts. (1) How can a training set selection method be developed, which would unify two selection criteria: similarity in time and feature space? (2) What effect the integration of the two criteria has to the accuracy of an adaptive learner?

We formulated a new concept of similarity in both time and feature space and the distance metric for adaptive training set selection, which integrates the two. An integration of time and space similarity is a conceptual contribution to learning under concept drift. It leads to a range of training set selection strategies including a training window and instance selection in space as boundary cases.

Based on the formulated concept and distance measure we developed a family of algorithms for training set selection under concept drift. FISH1 uses a preset balance between time and space similarity and preset training set size. FISH2 learns the training set size online at every time step using cross validation on the historical data. FISH3 learns online both the training set size and the balance between time and space similarity.

With FISH1 we demonstrate that for a gradually drifting data combination of time and space similarity can lead to a better classification accuracy than using a single technique. FISH2 algorithm shows the best accuracy in the peer group on the datasets exhibiting gradual drifts and a mixture of several concepts. FISH3 algorithm demonstrates that the balance between time and space similarity is learnable online. The computational complexity is reasonable for the field applications as compared to the peer instance selection algorithms.

The extensive numerical experiments using four alternative base classifiers and

two alternative distance in space measures on six real datasets demonstrated statistically significant improvement in classification accuracy as compared to two existing adaptive algorithms, which use only time or only space criterion.

Integration of similarity in time and feature space when selecting training set allows to improve generalization performance as compared to using only time or only space criterion under gradual concept drift.

Combining time and space similarity for training instance selection contributes to improvement of classifier generalization performance under gradual concept drift, since this way heterogeneous nature of the drifting data can be captured.

Chapter 5

Reoccurring Concepts: Context Awareness

In this chapter we address the issue of reoccurring concepts. That is the concept drift configuration where previously seen patterns reoccur, but it is not certain when exactly and in what form they will repeat.

We address recurring concepts using sales prediction task, where recurrences are particularly relevant. The task is to predict the quantity of the products to be sold next week based on historical sales data and external information. In this chapter we design a context aware prediction approach.

We select the training data and the base learner for each product depending on the structural properties of the historical sales data. First we present an approach where product categorization is done offline and at real time predictions the category is fixed. In the second part of the chapter we present an approach where the categories can be reassigned online.

The study includes extensive numerical experiments with a real dataset from a food wholesaler. The experiments indicate that there exist product subsets on which, using product categorization strategy, it is possible to outperform naive methods. Combining instance selection with product categorization leads to improved prediction accuracy as compared to the baseline methods.

The chapter is based on our publications [234, 236] and includes a material from

our paper [229].

The chapter is organized as follows. In Section 5.1 we give an introduction to the problem and highlight the main challenges related to food sales prediction. In Section 5.2 we present the principles of the context aware sales prediction approach (CAPA) and position it within the related work in Section 5.3. In Section 5.4 we implement CAPA using the food wholesales case. An experimental work is carried out in Section 5.5. Here the predictions are output online, the product categorization is done offline. In Section 5.6 we extend the approach to online categorization and present corresponding experiments in Section 5.7. In Section 5.8 we analyze, what products are predictable and what are their properties. Finally, Section 5.9 discusses future prospects and concludes.

5.1 Motivation

Demand prediction is an important part of business planning. In this study we address food wholesales prediction problem. However, observations and methods discussed here can be generalized to other sales prediction.

An accurate and timely sales prediction is essential for stock management. The stock includes large variety of goods, some of them require special storage conditions, some are quickly perishable. Thus, careful planning of stock is of crucial importance for food sales profitability. Planning cannot be done without estimating future sales.

Weekly predictions are essential for the stock management in food wholesales, a large part of which are fast moving and perishable goods. Thus we focus on the weekly predictions, aiming to spot medium term seasonal patterns.

There are general and product specific causes of the demand fluctuation. Variations in consumer demand may result from changes in price, promotions, weather changes or in longer run changing consumer preferences [202]. Furthermore, a large share of the products sold in that market is sensitive to some form of a seasonal change. Seasonal changes occur due to different cultural habits, religious holidays, fasting. All these factors imply that some types of products

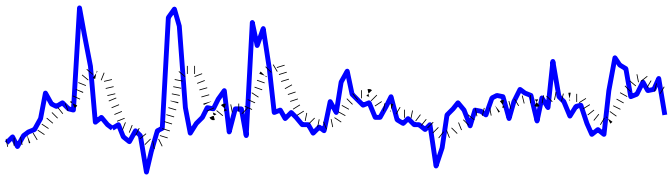


Figure 5.1: Predicting with a moving average as a baseline. Blue solid line represents the original series, black dashed line illustrates six weeks moving average (the baseline).

have high sales during a limited period of time.

Seasonal patterns can be expected, but the predictive features, which define those seasons, are not always directly observed. Therefore, fluctuations in sales which are accommodated by the changing seasons are often difficult to predict. Besides, the historical data is often highly imbalanced. For example, occasion specific seasonal products would have only a few weeks of the sales peaks per year.

Variation in sales figures can be classified into short term fluctuations (e.g. a buyer went to a pub today, thus will buy groceries tomorrow), medium term seasonal patterns (e.g. June vacations) and long term trends (e.g. changing economic situation, developing new eating habits). The baseline approach to prediction is to use a moving average[1] and manually adjust it based on the domain knowledge about the particular products.

Predictions based on moving averages may work reasonably well when demand is flat. When demand follows trend or seasonal patterns the reaction to the changes is slow, see Figure 5.1. Each time the sales start to rise the prediction follows up a couple of weeks later. The most important points to predict in terms of stock management are the rise at the beginning of the season and the decrease at the end of the season. Due to these drawbacks of the moving average method the predictions are overridden by human experts.

A manager needs to keep a number of triggers and reminders about the coming (school, national or religious) holidays, (warm or cold, sunny or rainy) weather and other factors that influence demand of particular food products. The results

[1]The average of the last L weeks, where L is the lag.

of such prediction highly depend on human factors like frustration, overload of information, lacking expertise (especially with a new personnel or for a new set of products). Simply forgetfulness may result in mistaken predictions and poor decision making. Managers often try to improve the performance in seasonal peak periods by increasing the number of safety days which results in a higher than necessary stock.

The discussed effects are related to seasonality (or reoccurring contexts [211]). Seasonality does not necessarily imply strict periodicity. We define seasonality as external time dependent factors, implying deviations of the sales patterns.

The goal of this study is to contribute to automation of sales predictions using intelligent data mining techniques. We present a context aware approach for sales prediction. For each product we classify it to one of the predefined categories based on structural properties of the sales time series. Then we apply a category specific predictor to each product.

5.2 Context Aware Sales Prediction Approach

In this section we present context aware sales prediction approach (CAPA). By context awareness we mean categorization of individual time series in space and assigning a specific training set formation strategy based on the category. First we give the general overview and then explain individual parts of the approach.

5.2.1 Intuition

The main idea of the context aware approach is to select the predictor based on the structural properties of time series. Different products have different sales behavior and different seasonality. If we can identify and extract distinct categories of products, specific input data construction procedures and specific predictors could be employed for each category.

One could argue, that an ensemble approach does that automatically. All possible input features can be collected and then apply rigorous feature selection and predictor selection from an ensemble. Ensemble approach has limitations with respect to a given food sales prediction problem. First of all, the data can be noisy and relatively short. For instance, a wholesaler keeps only two years record in their transactional database. If a product is seasonal and peaks only once per year for a particular event, a designer would have only one or two positive examples in the historical data for training. By defining the context, we filter out a part of noise.

Secondly, some series share common patterns. For example, New Year peaks are common for a large subset of products. By categorizing the time series based on their structural properties, we narrow down the job for the particular predictor, allowing to focus on the peculiarities of a particular series.

Another related approach would be a multitask learning ([17]). Yet instead of taking a 'black box' approach we use explainable judgment to filter out relevant part of the task.

5.2.2 Online operation of CAPA

CAPA is a collection of steps to output the prediction for one product at one time step. To obtain the outputs for all the time steps an sequential learning framework shall be followed (Section 2.1).

CAPA consists of two blocks - training (offline) and operational (online). We start with presenting operational online part and then describe, how the model is trained and parameterized offline. For now let us assume that the model has already been trained offline. That means:

- the product categories have been defined $(C_1, C_2, \ldots, C_c)$;
- mapping of the historical sales series to the categories is established $C = \mathcal{F}(\mathbf{y}_1, \ldots, \mathbf{y}_t)$;
- a base predictor has been assigned for each category $C \rightarrow \mathcal{G}$.

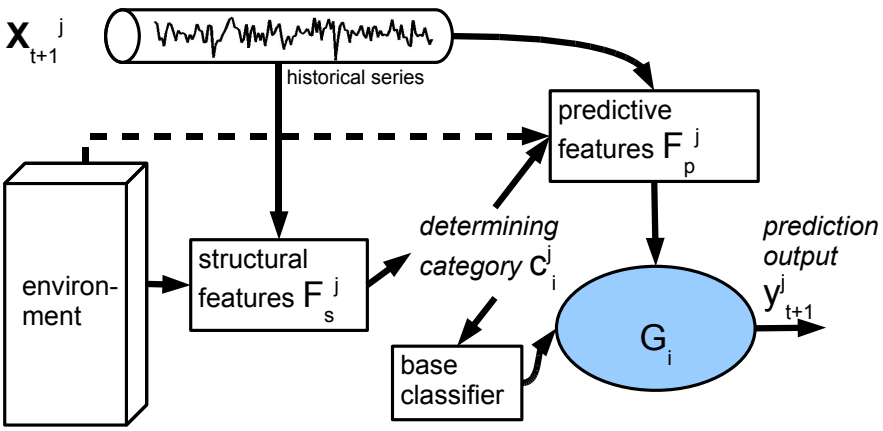


Figure 5.2: Online operation of CAPA.

Figure 5.2 presents online operation of CAPA.

To output a prediction for a given product at time t the following steps need to be taken. The historical sales data $(\mathbf{y}_1, \ldots, \mathbf{y}_t)$ is available.

1. We extract structural features from the original sales time series, $F_s = \mathcal{F}_s(\mathbf{y}_1, \ldots, \mathbf{y}_t)$. Structural features characterize time series as a whole. For example, mean of the series is a structural feature.

2. We assign the product to one of the categories $F_s \rightarrow C$. The mapping function has been trained offline. Generally we can assign a product to more than one category and use voting, but to keep the focus we delimit the mapping to one category.

3. We pick a base predictor specific to a particular category $\mathcal{G} \rightarrow C$ and select input features F_p relevant to a particular category. For example, if $\mathcal{G}$ is a moving average predictor, only historical sales are relevant as input features. While if $\mathcal{G}$ is a decision tree, we could form a larger feature space, e.g. (historical sales, weekday, holiday, temperature). The mapping from product categories to base predictors is the context aware part of the approach. The contexts are specified by predefined categories.

4. Having the original series, the base learner and the input features we can cast the prediction $\hat{\mathbf{y}}_{t+1} = \mathcal{G}((\mathbf{y}_1, \ldots, \mathbf{y}_t), F_p)$. Note, that the predictor

does not use the structural features F_s which were used for assigning the product to a category.

The prediction output now can be used for a decision making in a business process.

5.2.3 CAPA training

CAPA training can be divided into two parts: online and offline. Offline training basically defined the architecture and the parameters of the approach. Online training is repeated at every time step t, it was presented in the previous section. Now we discuss the offline part.

The two main tasks of the offline training are

1. to define the mapping between the product categories and the base predictors $(C_1, \ldots, C_c) \leftrightarrow (\mathcal{G}_1, \ldots, \mathcal{G}_m)$, and
2. to learn how to assign a product j to one of the categories, say C_j.

Then we would know what base predictor $\mathcal{G}_j$ to use to output the prediction online.

Mapping the product categories to the base predictors. A limited set of base predictors $(\mathcal{G}_1, \ldots, \mathcal{G}_m)$ needs to be preselected based on domain knowledge and expectations in order to delimit full state space search. Then each predictor is trained and tested using a reserved training set. A training set includes all the products. Thus for each product j we get m results (prediction accuracies). Next, each product is assigned to a category, based on the best performing predictor. We get all the products organized into up to m categories. Each obtained category serves as a basis for constructing categorization rules.

Categorization rules. The goal of the training process is to learn how to assign a product j to one of the categories, having only a fragment of the time series. Categorization rules define what structural features F_s shall be extracted from a given time series and how to map these structural features to the product categories $F_s \rightarrow C$. Categorization rules can be fixed based on a domain knowledge or trainable. Trainable categorization rules can be seen as a supervised learning task, where F_s are the inputs, $(C_1, \ldots, C_c)$ is a set of labels and we aim to learn a mapping $C = \mathcal{F}(F_s)$. The 'ground truth' is obtained from the previous training part, where all the products were organized into m categories.

When we establish the categorization rules and the mapping from a category to a base predictor, an unseen product can be processed online as described in Figure 5.2.

We presented the basic principles of the context aware sales prediction approach. The specific design elements shall be selected having in mind the problem and data at hand. CAPA is presented as sales prediction approach, we believe it is generalizable to other types of drifting time series prediction (e.g. consumption, production, even prices).

5.3 Related Work

The approach we present has connections with metal learning [203] and time series similarity [5] . Meta learning is concerned with extracting information from the input data as a group, in order to select a suitable classifier. CAPA is not truly meta learning in a way that it has separate feature construction for predictor selection and for prediction output itself. CAPA is related to time series similarity works, which often address the problem of time series querying. The relation is in feature reduction step. For that a series needs to be transformed into a distance representation either directly or after reducing the number of features. CAPA extracts structural features, which can be seen as feature reduction. Again, these extracted features in CAPA are used in categorization, but not used in a final prediction step.

In many real-world domains the situations seen in the past might partially repeat.

The situation is often referred as reoccurring contexts [210]. The phenomenon is related to seasonality; however, it should be emphasized that it is not known with certainty, when the patterns will reoccur.

Reoccurring contexts is often discussed in relation with hidden context [211]. Hidden context is information which is not present in the feature space in a learning task, however, the changes in that information affect the label. For instance, in ice cream sales prediction task temperature can be viewed as a hidden context if not included into a feature space. If it is raining the sales will drop, even the other observed variables will stay at the same level. Hidden context may not be observed, it may also be not possible to measure. In principle, concept drift might occur due to changes in hidden context or observed context. Reoccurring contexts is a subset of a concept drift problem.

Reoccurring contexts in supervised learning are mainly handled by trying to define and recognize the contexts. There are two main types of these methods: the ones implicitly assuming observed context and meta-learning related approaches.

The first assume that it is possible to recognize the context on an instance level. The methods which can handle reoccurring contexts can store concept descriptions [210], employ instance [49, 60] or batch [86, 103] selection to look for concepts.

The metal-learning related approaches assume that it is possible to extract hidden contexts from a group as a whole. A two level learning model is presented by Widmer [209] perceived context changes are used to focus the learner specifically on the information relevant to the current context. Klinkenberg [109] developed an approach, where at each time step not only the training window but also the type of base learner and its parametrization is selected from a fixed set of learners, using cross validation. Chen et al. [40] present higher order models, which in principle use meta-learning approach. They define several prototypes (learning them) and then introducing a switch mechanism. Mandl et al [141] detects a change in hidden context and expands the feature space adding a context feature.

The third type of methods capable of handling reoccurring contexts, are ensem-

ble methods which do not delete old classifiers, expecting that they might be useful in the future, e.g. [168]. There is no explicit context descriptions, the adaptivity is achieved via ensemble weights. See Section 2.3.1.1 in Chapter 2 for more details on adaptive ensembles.

One of the demand categorization approaches could be incorporated into our framework [114], which is left as a future work.

CAPA approach belongs to meta-learning related group. It is specific in a way that we incorporate external information and behavioral observations, based on the domain knowledge into categorization (context). We also manipulate the feature space (add relevant external features) based on the observed categories.

5.4 CAPA Implementation for Sales Prediction

In Section 5.2 we presented a general view of the context aware sales prediction approach. In this section we present CAPA implementation for food sales prediction case. In particular, we discuss the three design elements: how to obtain structural features F_s, how to construct input feature space F_p and how to learn the categorization rules $F_s \rightarrow C \rightarrow \mathcal{G}$.

5.4.1 Structural features

Structural features F_s should represent information about historical sales of a given product. They need to be independent from the length of the series in order to be applied online when different histories are available. In addition, we want the structural features to be related to observable seasonal patterns.

Using external knowledge and visual observations, we can categorize the series using two dimensions: seasonality and deviations (see Figure 5.3).

For example, bread sales can behave like 'flat' series. 'Frequent' series represent the products, which are bought in large quantities, following no particular seasonality. 'Occasional' product sales increase sharply in relation to particular

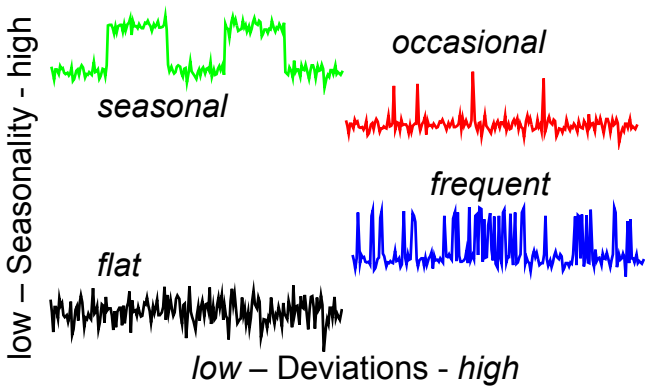


Figure 5.3: Structural categories of the product sales.

occasions, like eggs for Easter. Ice cream is a 'seasonal' products with respect to weather. The series in Figure 5.3 are artificial, for illustration purposes.

In order to capture seasonality and deviations we arbitrary define the following structural features:

- F_{s1-s3} |mean - median|, standard deviation, shift;
- F_{s4-s8} threshold h crossing ratios;
- F_{s9-s10} normalized power of the frequency p in the frequency spectrum;
- $F_{s11-s12}$ local variation features: interquartile range and unequal neighbors.

We normalize the values of the series to be in a range $(0, 1)$ before defining structural features.

The intuition behind these structural features is related to the categories of the series we defined. 'Flat' and 'seasonal' series are expected to have similar mean and median, while 'occasional' and 'frequent' series are expected to have upward deviations and thus larger difference between the mean and the median. 'Seasonal' and 'occasional' series are expected to have fewer crossings of the upper threshold that the other two.

The features F_{s1-s3} capture global characteristics of the series $\mathbf{y}$. Shift is the mean value of points for which $\mathbf{y}_t < \mu$ minus the median value of points for which $\mathbf{y}_t < \mu$, where μ is the mean of the time series.

In order to capture the behavior of the time series, we define a number of threshold values and note the number of times the signal crosses these thresholds (features F_{s4-s8}). This is done for the threshold values $h = 0.3, 0.4, 0.6, 0.7, 0.8$. This number is then scaled to the total length of the signal to obtain the ratio. This ratio indicates the structural nature of the signal w.r.t. seasonal recurrences.

Seasonal patterns could manifest themselves as, for instance, yearly or bi-yearly changes. This information should appear in the frequency spectrum of the time series as a relative high power in the frequencies $p = 1/52$ and $2/52$. In order to obtain these features we apply fast Fourier transformation (Cooley-Tukey implementation [43]) and extract the corresponding frequencies (features F_{s9-s10}).

We aim to capture local variation using features $F_{s11-s12}$. Unequal neighbors is the mean number of times $\mathbf{y}_t \neq \mathbf{y}_{t-1}$. The interquartile range of $\mathbf{y}_t - \mathbf{y}_{t-1}$ is a robust measure for the spread of variation in the signal. Any outliers that fall in the upper or lower 25% of the difference distribution will not affect it.

5.4.2 Constructing the input space

In this section we discuss the input space. These features F_p are used by the prediction models, while previously discussed structural features F_s are used for product categorization (see Figure 5.2).

The input feature space is formed using internal an external data. Internal data comes from a food wholesales company database, where historical sales are stored. External data (holidays, temperature, seasons) is formed using information from the ministry of culture, meteorological institute and common knowledge. The input feature set is specified in Figure 5.4.

The internal features are interrelated. Moving average (F_{p2}) is calculated using (F_{p1}). Last years moving average (F_{p3}) is formed taking the values (F_{p1}) and using one year lag. Cumulative sales (F_{p4} and F_{p5}) include the sales of all the products in quantity terms. We checked corresponding series in monetary sales terms to verify that there is no significant distortion due to different quantity

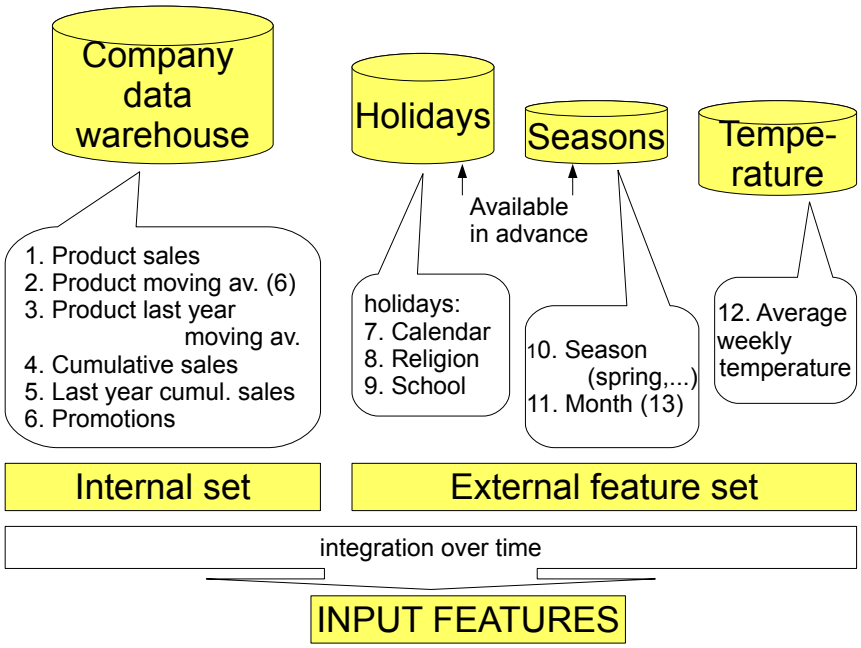


Figure 5.4: Formation of the Feature Space.

measures. The company organizes promotions (F_{p6}) for selected products, this can be known in advance.

The external features (F_{p7-p11}) are available in advance. Average weekly temperature (F_{p12}) can be predicted sufficiently accurately one-two weeks in advance. The external features add on much to dimensionality. Holidays (F_{p7-p9}) are described in 16 binary features. Seasons (F_{p10}) are described in 4 binary features, and months (F_{p11}) are described in 13 binary features. Temperature (F_{p12}) is described in a single numerical dimension.

5.4.3 Learning to categorize

We would like to learn categorization rules, which would assign a given product to one of the defined categories based on it's structural features. This learning is done offline using a reserved training set.

We present two approaches *trainable* and *domain based*. In the first approach we use the training accuracies to label the training products and using these

labels try to learn categorization in a supervised manner. In the second case we visually pick a set of representations from the four categories ('flat', 'frequent', 'occasional' and 'seasonal') defined earlier and use them as prototypes to learn the categorization.

5.4.3.1 Trainable categorization

We aim to learn a mapping from the structural features to categories $C = \mathcal{F}(F_s)$. The 'ground truth' labels to be used for learning are obtained all the base predictors $(\mathcal{G}_1, \ldots, \mathcal{G}_m)$ on all the products. A product gets the label, corresponding to the number of the most accurate base predictor.

Training. We want to be able to interpret the decision boundaries, used for product categorization. Thus we select a decision tree as trainable categorization rule $\mathcal{F}$. We use the 12 structural features F_s described in Section 5.4.1. Since it is unknown a priori which features will result in the best classification, we perform a full search in the structural feature space. That means that for every possible subset of features a decision tree is trained and evaluated. Now we have a pool of decision trees.

Validation. The final decision tree is selected based on the accuracy of the final base predictor $\mathcal{G}_j$, which was selected based on the output C_j of the decision tree. Recall (Section 5.2), that a mapping $C_j \rightarrow \mathcal{G}_j$ is prefixed. We use cross validation on the training set to select the categorization rule.

5.4.3.2 Domain based categorization

A categorization rule can be selected based on domain expertise. We visually pick a number of products to represent a category. Then we extract structural features F_s from each of the picked series. We average the structural features within each of the four categories and the averages serve as the four prototypes. Finally, we cluster the products to the four categories, using prototypes as fixed cluster centers.

5.5 Experimental Evaluation of CAPA, part I

In this experimental setting we aim to test if the base predictor, which is selected based on time series category, consistently outperforms alternative models in terms of prediction accuracy. We aim to predict one week ahead the quantity of weekly product sales, aggregated over all the locations of the food wholesales company.

5.5.1 Data

We use the sales dataset from Sligro Food Group N.V. (SLIGRO). The company is engaged into food wholesales. SLIGRO works with corporate clients, mainly food retail and food service companies (restaurants), although there are some direct consumers as well. SLIGRO has around 40 outlets. The group pursues a multi-channel strategy, covering various forms of sales and distribution (cash-and-carry and delivery service) and using several different distribution channels (retail and wholesale). SLIGRO trades about 60000 products.

Our experimental field consists of 538 product sales quantities over two years period (from July 2006 to October 2008). The sales are aggregated on weekly basis, thus each series is of 119 weeks length. Each series represent the sales of one product aggregated over all outlets.

We use a regression as the base predictor, equipped with feature selection mechanism. Although we partially preselect the input features, still further selection needs to be carried out. For instance, we have 15 calendar events but only a few of them might be relevant for a particular product. We run Principal Component Analysis and keep the new features, which explain at least 70% of the data variance.

We use discretized labels to achieve comparability between different products. We discretize the outputs to 8 classes from very low sales (1) to very high (8), using Symbolic Aggregate Approximation (SAX) [138].

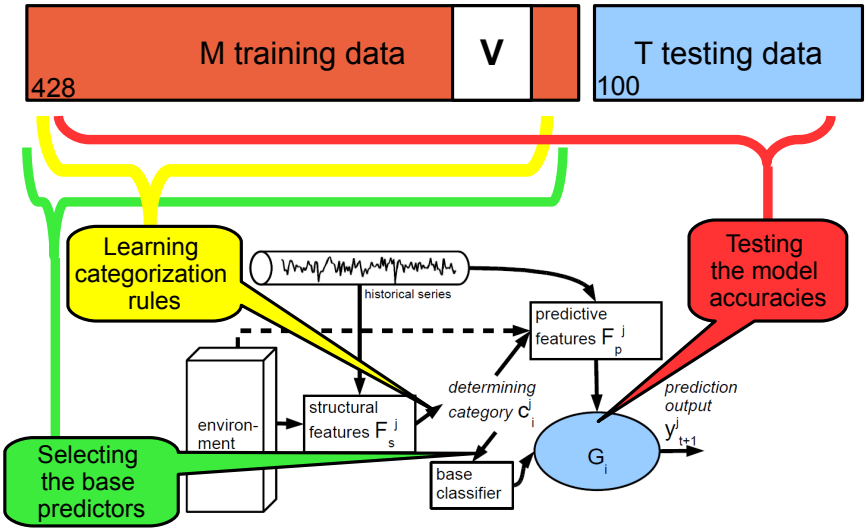


Figure 5.5: Experimental scenario depicted on the CAPA model presented in Figure 5.2.

5.5.2 Experimental setup

Experimental scenario consists of three parts: selection the base predictors, learning categorization rules and testing final model accuracies. In the first part we select the base predictors and obtain 'true' labels. We show show that there exist product subsets on which it is *possible* to outperform baseline predictor. In the second part we aim to learn the dependencies between product categorization accuracies and sales prediction accuracies applying two approaches *trainable* and *domain based* categorization. In the third part we test the final model accuracies. The experimental scenario and its position in CAPA is sketched in Figure 5.5.

To quantify the results, we perform controlled experiments using 538 product sales history. We take out at random 100 series from the dataset and reserve them for final *model testing*. We call this set **T**. We develop the model using the remaining 438 series. We call the remaining set **M**.

We employ sequential learning framework, i.e. we test the prediction accuracy sequentially. That means at time t we have all the historical sales $\mathbf{y}_1, \ldots, \mathbf{y}_t$ available and want to predict sales level $\hat{\mathbf{y}}_{t+1}$. We use discretized outputs, but real valued inputs, normalized to $(0, 1)$ before regressing them. We rerun discretization procedure after including each new value into historical set.

For model evaluation we use *mean absolute scaled error* (MASE) [94]:

$$MASE = \frac{1}{n}\Sigma_{t=1}^{n}|\frac{e_t}{MAE(Baseline)}|, \tag{5.1}$$

where e_t is the prediction error at time t, $MAE(Baseline)$ is the mean absolute error of the baseline method. We use naive one step ahead prediction as the baseline (the prediction for the next week is equal to the factual sales this week). This is a special case of moving average, when lag is equal to one.

Having in mind that we use MASE as the accuracy measurement, we implicitly compare the intelligent classifiers to the naive method.

5.5.3 Selecting the base predictors

Having observed four different categories of the series (Figure 5.3) we narrow down selection of the base predictors to a regression with different sets of input features. The input features F_p are listed in Table 5.1. The index in superscript indicates the time from which the features are taken, assuming we are now at time t and predict the sales for time $t+1$. For example $F_{p9}^{(t+1)}$ means school holidays next week and $F_{p9}^{(t)}$ means this week's school holidays.

Table 5.1: Base predictor selection

	Base	Input features
G_1	MA(1)	$F_{p1}^{(t)}$
G_2	MA(3)	$F_{p1}^{(t-5...t)}$
G_3	MA(6)	$F_{p1}^{(t-5...t)}$
G_4	reg.	$F_{p1}^{(t-5...t)}$, $F_{p6}^{(t+1)}$
G_5	reg.	$F_{p1}^{(t-5...t)}$, $F_{p2..p5}^{(t)}$,$F_{p6,p11}^{(t+1)}$,
G_6	reg.	$F_{p1}^{(t-5...t)}$, $F_{p2}^{(t)}$, $F_{p6,p10}^{(t+1)}$, $F_{p7..p9,p12}^{(t,t+1,t+2)}$
G_7	reg.	$F_{p1}^{(t-5...t)}$, $F_{p2..p5}^{(t)}$, $F_{p6,p11,p12}^{(t+1)}$, $F_{p9}^{(t,t+1,t+2)}$

- G_1, G_2, G_3: moving average with lags of $1, 3$ and 6 weeks. These are expected to work well on 'flat' type of products with no particular relation

to seasonality.

- G_4: a linear regression trained on subsequences of 6 weeks. It is expected to work well on 'frequent' or 'flat' series, where there is no particular relation to calendar events or seasonality.
- G_5: a linear regression trained on subsequences of 6 weeks, present and last years moving averages, present and last years cumulative sales, months. This classifier is designed for periodic products.
- G_6: a linear regression trained on subsequences of 6 weeks and a range of calendar, temperature and seasonality related features. This method is expected to pick up specific events information for 'occasional' products.
- G_7: a linear regression trained on subsequences of 6 weeks and a range of annual patterns related features. This method includes aggregated sales, month indications, calendar events, expecting that there is somewhat annual periodicity in the 'seasonal' series.

We run all the predictors on 438 series. We split the series into two parts 57 weeks used as a 'warm up' and then the remaining 61 weeks are used for sequential testing. We obtain 438×7 matrix of scaled accuracies (MASEs). Then we group the products based on the top ranking predictor. This way we get the 'ground true' categories for the given settings. In Table 5.2 the average MASEs for each of the 'ground true' categories are provided.

Table 5.2: Average MASEs for the 'true categories'.

	Size	G_1	G_2	G_3	G_4	G_5	G_6	G_7
C_1	200	**1.00**	1.31	1.63	1.63	1.89	1.91	1.93
C_2	68	1.00	**0.89**	1.02	1.25	1.56	1.53	1.62
C_3	36	1.00	0.92	**0.85**	1.04	1.21	1.21	1.22
C_4	35	1.00	1.04	1.09	**0.85**	1.00	1.01	1.03
C_5	19	1.00	1.06	1.17	0.96	**0.86**	0.99	0.93
C_6	43	1.00	1.02	1.11	0.97	0.98	**0.85**	0.94
C_7	37	1.05	1.08	1.18	1.02	0.98	0.98	**0.90**

The best results appear on the diagonal in bold. The results below *one* mean that *it is possible to outperform moving average* if we do the correct categorization online. Furthermore, there are more than a single base predictor per line, which outperform the baseline predictor. It means that there are alternative methods which can outperform the baseline predictor within the defined products categories.

SLIGRO is currently using 6 weeks moving average, which is G_3. G_3 is on average worse than the 'intelligent' methods G_4, G_5, G_6 and G_7 in the last four categories which represent the 'intelligent' methods.

Let us look at the pool of the predictors from another angle. For each predictor we count the number of products which gave MASE below *one*. That is, we count the number of wins against baseline predictor. If there is no method on a given product to outperform the baseline predictor, G_1 gets the score. The results are in Table 5.3.

Table 5.3: Performance counts for the 'true categories'.

	G_1	G_2	G_3	G_4	G_5	G_6	G_7
C_1	**200**	0	0	0	0	0	0
C_2	3	**65**	28	16	13	9	13
C_3	1	28	**35**	20	13	17	13
C_4	0	12	16	**35**	21	17	18
C_5	0	8	6	12	**19**	9	14
C_6	2	17	17	24	26	**41**	32
C_7	2	15	11	21	28	25	**35**

The first three elements of the diagonal sum to $2/3$ of the total training set and these are the moving averages. That means $1/3$ of the products in this set could be better predicted using intelligent methods with additional features.

5.5.4 Learning the categorization rules

We test the two categorization approaches *trainable* and *domain based*, which were described in Section 5.4.3. We delimit the task to four categories, which we expect to correspond to the product categories presented in Figure 5.3. Thus

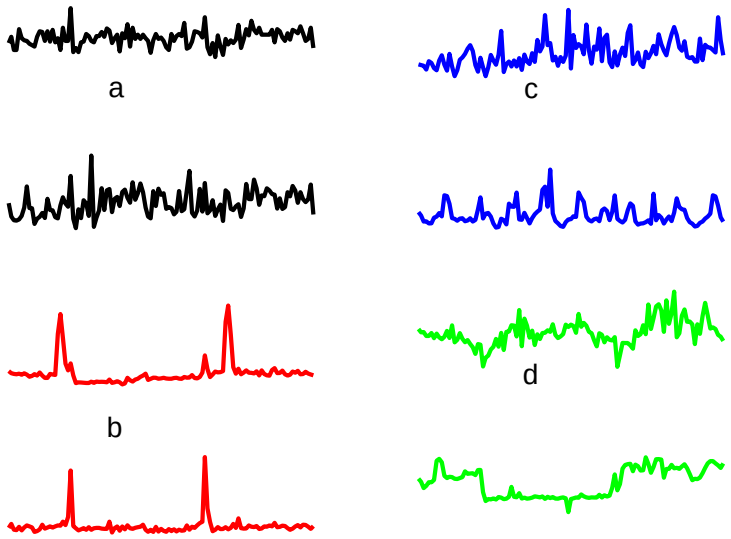


Figure 5.6: Examples of the prototype products (a) 'flat', (b) 'frequent', (c) 'occasional', (d) 'seasonal'.

we group the three moving averages (G_1, G_2, G_3) into a single category C_1. A basic regression (G_4) forms the second category C_2. The third category C_3 is the calendar based regression (G_6). And the fourth includes both annual patterns related regressions (G_5 and G_7).

In both cases we use the full length of the series (119 weeks), normalized to fit the range $(0, 1)$.

In *trainable* approach we use the training set **M** for obtaining the 'true' labels and learning the categorization tree.

For *domain based* approach we visually pick four products for each category (see Figure 5.6 for examples), sixteen products all in all.

Figure 5.7 shows the averages of the structural features for the resulting categories and the prototypes. there is a clear distinction between the categories in prototypes (b), however looking at all the products (a) the distinction between the categories is not so clearly expressed. This suggests that there are mixed series within the product pool, or categories are changing in time.

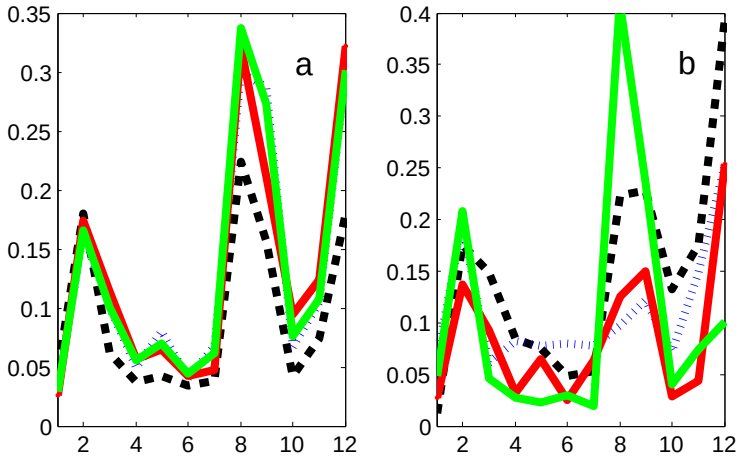


Figure 5.7: Structural features of (a) the obtained categories, (b) the prototypes.

We present the results of the categorization procedure in the next section together with the final accuracies.

5.5.5 Prediction accuracies

We develop and validate CAPA model using the training set **M**. Then we test the model assuming online settings on the testing set **T**. We are interested in MASEs within each product category, assigned using a categorization rule.

The testing protocol is as follows. We split each series into 'warm up' (57 weeks) and testing (61 weeks). For the dataset **M** we use the categorization, which was obtained during the training phase. We categorize the products from the dataset **T** using only 'warm up' part and the categorization rules obtained in the training phase.

We run sequential testing of the four models G_1, G_4, G_6 and G_7, which we assume to correspond to each of the four product categories. We run the four models on each of the series from both datasets **M** and **T**. We compare the four accuracies for each product series. We aim to minimize the MASE for each pair of model-category.

First, we present the accuracies of the training data **M**, which corresponds to

offline settings. In Table 5.4 average MASEs for the obtained categories are listed. It can be seen that in training *bottom up* approach (a) the selected base predictors G_4, G_6 and G_7 outperform the baseline predictor in the corresponding categories C_2, C_3 and C_4 .

Finding a categorization rule that results in a classification with clear dominance of a particular method is not hard. Out of 4096 total classifiers (i.e. feature combinations) 3541 manage to find a classification that meets this requirement. We do 10-fold cross validation on the training data to pick a single classifier. The features selected to for a decision tree were: $2, 5, 8, 10, 12$ (see Section 5.4.1). We get 81% categorization accuracy on the training set.

To validate the results, we do the following experiment. We assign each product to a random category and then calculate average MASEs correspondingly. We present the results in the same Table 5.4 (c). It can be seen from the accuracies above 1 that in random categorization case the baseline predictor prevails. The *top down* categorization method (b) does not outperform the baseline predictor, however it gives better than random results, leaving prototyping approach as promising future direction. The poor performance of the method can be attributed to categorization accuracy, which is only 43% on the training data. However, random categorization would be only 25%. Thus we are better than random, but not enough to beat the final accuracies of the baseline predictor.

Let us check, how accurate should be the categorization, so that the baseline predictor is still outperformed. In Figure 5.8 we depict MASE of each of the four categories as a function of categorization accuracy. 100% corresponds to the 'true categories', 0% corresponds to random categorization. We fill in the figure by increasing share of random categorization. For example, 80% means that we randomly select 20% of the products and assign random categories, while the remaining part is assigned the 'true categories'.

The area within the ellipse in the Figure 5.8 is the region of interest. This is the area where the three categories (C_2, C_3, C_4) outperform the baseline predictor in terms of MASE. This shows, that application of CAPA makes sense if we are able to assign the products to the specified categories with the accuracy higher that 85%.

Table 5.4: Average accuracies for the training data **M** (a) trained categorization *bottom up*, (b) trained categorization *top down*, (c) random categorization.

		Size	G_1	G_4	G_6	G_7
(a)	C_1	328	**1.01**	1.43	1.68	1.71
	C_2	34	1.00	**0.89**	0.97	1.00
	C_3	31	1.00	1.00	**0.94**	0.97
	C_4	45	1.00	0.98	0.95	**0.91**
(b)	C_1	158	**1.01**	1.69	2.13	2.20
	C_2	32	1.00	**1.34**	1.40	1.40
	C_3	7	1.00	0.95	**0.94**	0.92
	C_4	241	1.00	1.07	1.12	**1.12**
(c)	C_1	100	**1.01**	1.36	1.59	1.54
	C_2	108	1.00	**1.35**	1.66	1.59
	C_3	117	1.01	1.28	**1.60**	1.51
	C_4	113	1.00	1.35	1.63	**1.56**

Now let us have a look how the categorization works on the testing data for online settings. Note, that we use only 'warm up' part of the data here for categorization.

Table 5.5 presents average MASEs of the obtained categories on the dataset **T** for *bottom up* (a) and *top down* (b) categorization approaches. Along we present the results of random categorization (c). The results do not show MASE below 1 for the target categories. This is due to not sufficient categorization accuracy, which is 47% for (a) and 43% for (b). However, random categorization would give only 25% accuracy. Thus we managed to learn some categorization and these are promising results.

In Table 5.6 the performance counts for the test data **T** are listed. For example, the cell (C_1, G_4) means that in the category 1 there were 18 cases when the predictor G_4 outperformed the baseline predictor.

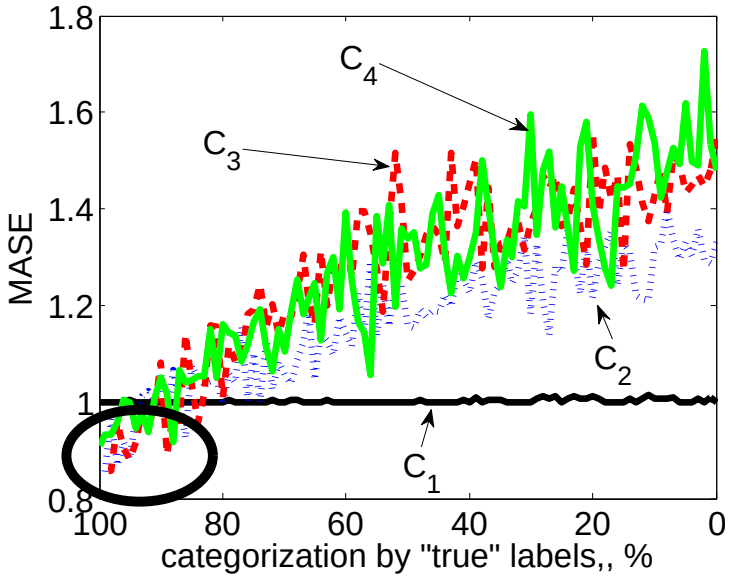


Figure 5.8: MASE as a function of categorization accuracy.

5.6 CAPA with Online categorization

CAPA provides online sales predictions, but in previous experiments product assignment to the four predefined categories is done once at the beginning offline. In the remaining part of the chapter we implement online categorization. The intuition behind CAPA remains the same, the approach is used to output sales prediction one week ahead. But instead of offline product categorization we allow reassignment of the product category at each time step. This serves for two purposes.

- It is possible that a product changes it's category. For instance, a competitor beer is removed from the shelves, thus the sales of another beer jump to become seasonal from previously very small and flat.
- Sequential learning online allows to accumulate larger history, which sometimes improves the categorization accuracy.

To purify the effect of online categorization we aimed to simplify the setup as much as possible. We made several modifications with respect to the original CAPA setup. The modifications and motivation are the following.

Table 5.5: Average accuracies for the testing data **T** (a) trainable categorization, (b) domain based categorization, (c) random categorization.

		Size	G_1	G_4	G_6	G_7
(a)	C_1	69	**1.00**	1.37	1.57	1.65
	C_2	3	1.00	**1.41**	1.63	1.60
	C_3	20	1.00	1.62	**1.84**	1.85
	C_4	8	1.00	1.17	1.26	**1.28**
(b)	C_1	39	**1.00**	1.29	1.37	1.43
	C_2	4	1.00	**1.22**	1.39	1.42
	C_3	1	1.00	1.21	**1.23**	1.36
	C_4	56	1.02	1.32	1.67	**1.70**
(c)	C_1	24	**1.00**	1.30	1.50	1.59
	C_2	33	1.00	**1.37**	1.57	1.63
	C_3	15	1.00	1.72	**1.92**	1.93
	C_4	28	1.00	1.39	1.58	**1.63**

Table 5.6: Performance counts for **T**.

(a)	G_1	G_4	G_6	G_7
C_1	45	18	17	19
C_2	2	1	1	1
C_3	15	4	2	2
C_4	4	3	4	3

Product categories. Product categorization is changed from 4 to 2 classes: 'predictable' and 'unpredictable'.

Classifier type for product categorization. The Linear Discriminant Classifier is used for product categorization. The reasons for picking this classifier are:

1. interpretable weight vector (importance of features),
2. unique model for a given dataset,
3. easy to manipulate prior probabilities and costs of misclassification.

We train the model assuming equal prior probabilities, but in fact we have about 4 times more of 'unpredictable' products. Thus we include a bias of the discrimination threshold towards 'unpredictable' class, using a threshold -0.002 which was learned experimenting with the training data to match the prior probabilities of the classes.

Normalization and discretization of the series. We changed normalization and discretization of the time series to more domain driven approach. Previously the normalization and discretization was done w.r.t. minimum and maximum value of the series. There was no relation to absolute volume of sales. For example, two series $\mathbf{y}_a = (2323)$ and $\mathbf{y}_b = (102103102103)$ would be equal to $\mathbf{y}^* = (0101)$ after normalization. However, the change in $\mathbf{y}_a$ in absolute value is much more significant than in $\mathbf{y}_b$.

Now normalization works as follows:

$$\mathbf{x}_t^* = \frac{\mathbf{x}_t}{\text{mean}(\mathbf{x}_1, \ldots, \mathbf{x}_T)}, \tag{5.2}$$

here $\mathbf{x}_t$ is the original signal at time t and $\mathbf{x}_T$ is the last instance of the training data.

Change detection. We introduce change detection to handle the series with significant jumps or drops in the middle (see Figure 5.9). These jumps correspond to the the new year dates. We guess that they renegotiate contracts with the suppliers. Thus we take a naive approach to cut the training history at those points and start a new training history.

Change detection criterion is the following:

$$K = \frac{2|m_1 - m_2|}{m_1 + m_2}, \tag{5.3}$$

$$\text{if } K > 1.0 \text{ then a change is detected}, \tag{5.4}$$

here m_1 and m_2 are the means of the training data coming from different years. The detection threshold is set based on visual inspection, selecting the changing

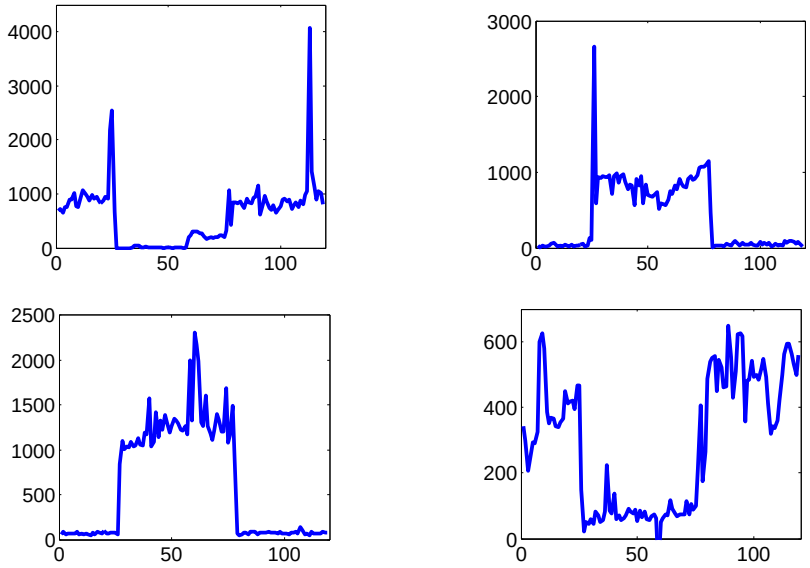


Figure 5.9: Examples of new year changes in product sales, x axis: time (weeks), y axis quantity (units).

products from the training data. For now we detect only the fist change and automatically assume

In the next section we evaluate the predictions under the modified setup.

5.7 Experimental Evaluation of CAPA, part II (online categorization)

In this section we present two groups of experiments. The purpose of the first group of experiments is to show the predictability of a category under new setup. The second group of experiments evaluates the idea of dynamic category selection online as compared to static categorization.

We use the same structural features, the same product set and the same split into training and testing as before. The categorization rule (linear discriminant classifier) is trained and selected using 5 fold cross validation on the training data.

Table 5.7: Predictability: scaled mean absolute errors.

	naive	6 weeks MA	intelligent	average size
Training data				
'unpredictable'	1.000	1.341	1.730	323
'predictable'	1.000	0.987	**0.940**	115
Testing data				
'unpredictable'	1.000	1.383	1.762	80
'predictable'	1.000	0.970	**0.950**	20
Random clustering				
'unpredictable'	1.000	1.306	1.604	50
'predictable'	1.000	1.296	**1.593**	50

5.7.1 Predictability of a category

For predictability of a category we present three result tables. The first one shows the obtained accuracies for training data. The second shows the obtained accuracies for the testing data. The third table shows the accuracies for the testing data after categories were assigned at random.

The final results are averaged over 50 runs. That means the experiment is repeated 50 times, each time random assigning into 5 cross validation set is different, therefore we come up with slightly differently parameterized classifiers.

We include 6 weeks moving average as a benchmark to see how the current company method would perform. 'naive' means 1 week moving average, the same as in previous setup.

The results are provided in Table 5.7.

We train the linear discriminant on the structural features derived from a full length of the training data. The main delimitation in this setup is that we also use the full length to derive the structural features of the testing data.

In 'predictable' class the intelligent method outperforms both naive predictors when we use trainable categorization. While at random categorization the

	Testing result
Scenario 1, static	1.0136
Scenario 2, dynamic, fixed window	0.9952
Scenario 3, dynamic, incremental window	0.9953

Table 5.8: Online categorization: scaled mean absolute errors.

intelligent method gives the worst result. The results suggest that learnable categorization is meaningful.

5.7.2 Dynamic category selection

In this experiment we train the linear discriminant developing the structural features of the products only from one year sales history to match the testing setup and the reality more closely. For the linear discriminant again 5 fold cross validation is used. Here we increase the threshold of the decision linear discriminant to -0.004 to reduce the number of false positives.

We test three scenarios on the testing data:

1. The product category is assigned once and forever based on the structural features developed from the first year of the sales history.
2. The product category is reassigned every week based on a moving window (one year size) of sales history.
3. The product category is reassigned every week incrementally. Every week a new sales data is added, thus the history window is expanding.

The dominance of scenario 2 would mean that the category is indeed changing during this small period of 2 years. The dominance of scenario 3 would mean that the accuracy of categorization increases when we have more historical samples available.

The results are provided in Table 5.8. The results are averaged over 100 runs.

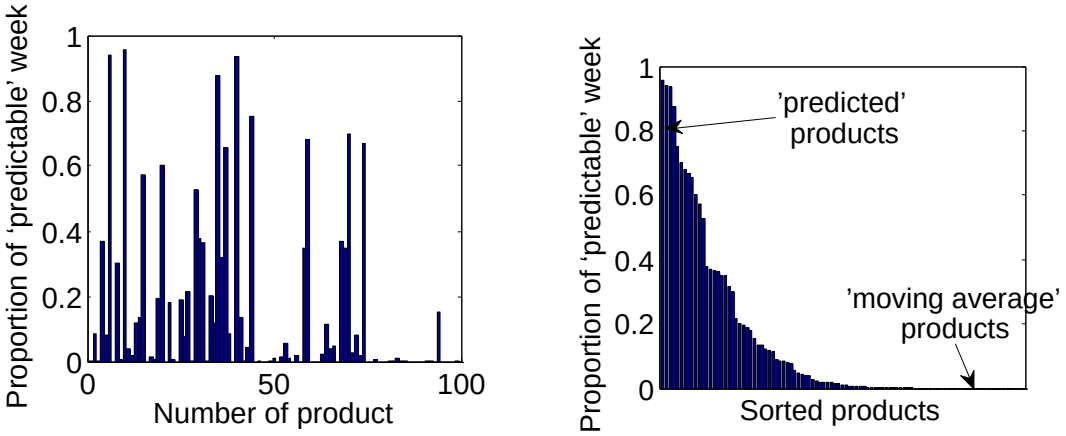


Figure 5.10: Online categories of the testing products: (a) in order, (b) sorted.

Scenario 2 and 3 clearly outperform scenario 1, that implies that online assignment of categories is beneficial. Scenario 2 (windowing) is on average slightly better than scenario 3, that suggests possible concept drift within a given product.

Now let us look how many products have were migrating from one category to the other online. In Figure 5.10 we plot the bars of predictability for testing products (100 products). On y axes '1' means that the product was assigned to a 'predictable' category, '0' means 'unpredictable'. We average the assignments for a given product over all the 60 testing weeks and then average over 100 runs. Note that the graphs are based on the linear discriminant outputs, not on the 'ground truth'.

The results show that there are many heterogeneous products thus the system accuracy benefits from online categorization.

5.8 What Products Are Predictable?

In this section we look, what products are predictable and what are their properties. In Figure 5.11 we plot the six most accurately predictable products by the intelligent methods. The scaled mean absolute errors are correspondingly

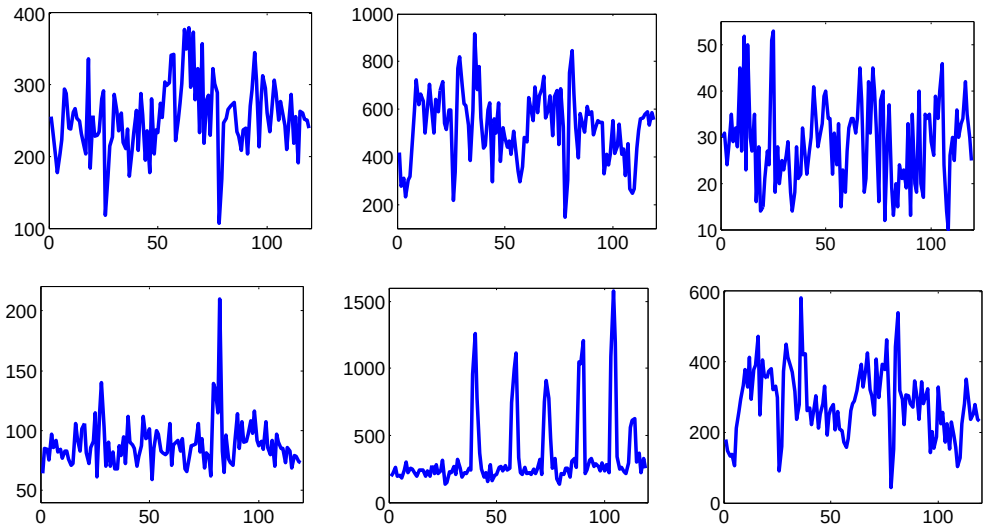


Figure 5.11: The top 6 *predictable* products.

$0.61, 0.64, 0.72, 0.78, 0.83, 0.87$. The top predictable products are: meat, croissants, coffee, anchovies, toilet paper.

Most of the predictable products seem chaotic from visual inspection (except maybe the 5^{th} one, which is rather periodic). However, all the top products seem visually alike in their behavior. Visual inspection suggests, that the historical sales of the predictable products share some common structural features. Thus there should be a room for intelligent data mining methods.

Now let as look what are the least predictable products, i.e. where the intelligent classifier makes the largest error. The six worst are depicted in Figure 5.12. The errors are correspondingly $8.50, 4.11, 4.09, 3.53, 3.50, 3.42$. The worst predictable products are: another brand of toilet paper, prepared salad, shampoo, meat, coffee, cheese. The list is similar to the best predictable products. Thus it can be concluded that the predictability does not highly depend on the type of product, but rather on the brand.

It can be seen that most of the 'troublesome' products have a hill pattern at the new year. More sophisticated change detection could contribute to dealing with those.

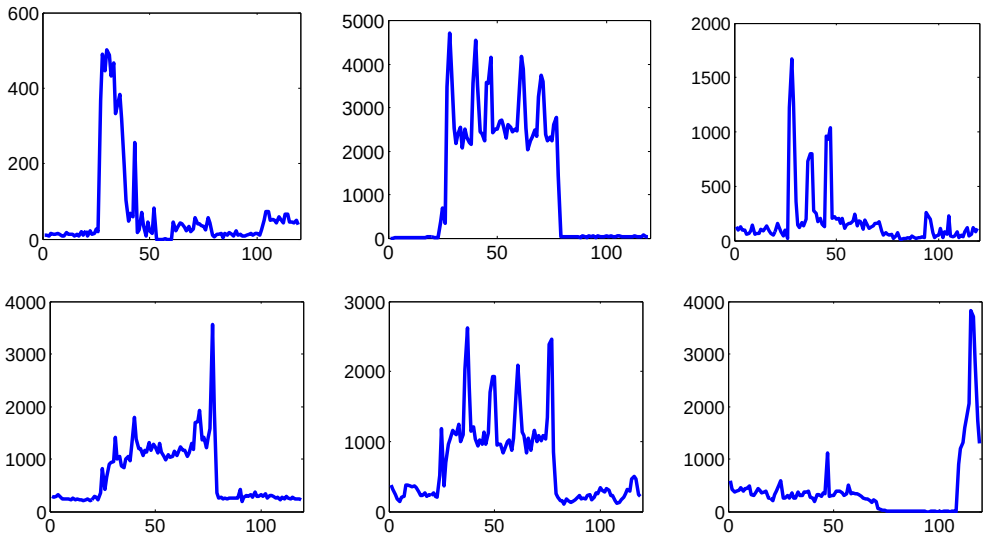


Figure 5.12: The 6 most *unpredictable* products.

5.8.1 Implications for business decision making

We illustrated CAPA using a case of SLIGRO. In their business process the predictions are made using moving average but human corrections are made on top of the obtained predictions. Human factor is a volatile part of the decision making. Thus we aimed to summarize the seasonality related factors to build intelligent models with integrated context knowledge.

We presented a set of controlled experiments to justify our claims regarding context aware prediction. The data represent a static snapshot of the database. In reality the company would have a moving window of the sales history, i.e. all two years data could be used to cast the prediction for the next week. After the true sales values for the predicted weeks are obtained, it will be possible to include it into the training data. This way it would be possible to reassign product category online. Next cycle would be performed on an updated database.

CAPA is generic in a sense that we can reassign the product to another category at each time step based on recent development of sales. If a concept drift is happening domain expert could interfere using the feedback loop.

5.9 Conclusion

In this chapter we addressed the training set formation problem under recurring concepts, which happens when several data generating sources are expected to switch over time at irregular time intervals. We addressed a problem of recurring concepts using a case of food sales prediction, where recurring concepts are often expected. Sales prediction is a complex task due to a number of hidden factors influencing the sales and there are different seasonality patterns across the product assortment. Identifying the past concepts and using this information while forming the training set is expected to contribute to the final prediction accuracy.

The third research question (RQ3) consisted of two parts. (1) How can a contextual learning method be developed, which would relate training set formation to the context (the types of historical data 'behavior')? (2) To what extent such method can increase the accuracy of food sales prediction, where recurring concepts are often expected?

We demonstrated that identifying and learning to recognize the types of historical behavior allows to form a training set in a way that this historical information contributes to the present prediction accuracy.

We developed context aware prediction approach, which forms a training set interactively, based on the type of historical behavior, which is determined employing structural features. Training set formation based on structural features is novel in the field. Structural features describe a set or a sequence of instances.

Using a real data from the food wholesales company (a database of 538 product sales over a period of two years), we experimentally validated CAPA. We demonstrated that distinct behavioral categories exist and how it is possible to learn product categorization rules. We showed that online reassignment of the categories helps to increase the prediction accuracy. Therefore, it is likely that each product is not homogeneous and it would be worth exploring the regions of predictability within the product to employ online classifier switching. The experiments demonstrated 5% improvement in the testing prediction accuracy as compared to the baseline prediction, which is relevant for field applications.

Contextual training set formation, while connecting the types of historical sales with the training set formation strategies and learning to recognize the types online using structural features, allows to improve generalization performance in food sales prediction task, where reoccurring concepts are expected.

Chapter 6

Industrial Case Study: Handling Concept Drift in Boiler Operation

In this chapter we address a problem of mass flow estimation for a burning process. We analyze the case of circulating fluidized bed (CFB) boilers, which use fuel as input to produce heat. When the fuel input and tuning are done manually, the process is subject to a concept drift. The first goal is given a stream of operational data to estimate a mass flow signal *online*, which is a challenging task due to the presence of noise and outliers. The second goal is to signal *online* the start and the end of fuel feeding stage. Accurate estimates of fuel consumption are needed for control systems.

In this case study we use real data collected from an experimental CFB boiler. We develop a generic approach for online estimation of the mass flow under concept drift assumption using sensor measurements. We address the problems of labeling the data offline, detecting a sudden concept drift which corresponds to the start and end of a feeding stage, and learning an accurate signal estimator for *online* operation. Last but not least we emphasize the importance of having the domain knowledge concerning the considered case.

The chapter is based on our publications [12, 235] and includes a material from our paper [13].

The chapter is organized as follows. We start Section 6.1 giving motivation and

overview of the problem of a mass flow prediction in CFB boiler. In Section 6.2 we present the model we propose for online mass flow estimation. Section 6.3 contains the experimental evaluation and discussion of the results. We conclude and point out open problems in Section 6.4.

6.1 Motivation

Mechanical devices, like CFB boiler, typically are comprised of moving parts. The movements cause interference in the observed sensor data, to which we will refer as *noise*. Filtering out the true signal from the measured noise is a challenging task. Accurate estimates of the true signal are needed for control systems.

6.1.1 Origin of the signal

CFB is a technology used in power plants, oriented towards efficient use of fuel and emission control. A simplified process is illustrated in Figure 6.1 (a). Heat is produced as a result of burning a mix of fuels with added air and limestone. The heat is used to make steam, which is the output of a system. As the boiler burns fuel to produce energy, the amount of fuel in the container is decreasing. At some point in time a new portion of fuel is added to the container resulting to an increase in fuel mass.

CFB boilers are flexible on fuel types. Coal and biomass are the most popular fuels. The fuel can be very non homogeneous (especially biomass as a mix of wood particles, peat, plants and other materials), which raises a challenge to model burning as a physical process. A mix of fuels can also be used.

The fuel input is not a continuous process due to inhomogeneity of the particles. In addition, in small boilers fuel input process can be manual, like in our case study. It means, that a human operator inputs fuel into the container. The human operator can regulate the mixture of fuel using a mixing screw and burning speed using the feeding screw. Different amounts of fuel can be added to the boiler at irregular time intervals resulting in sudden drifts in a signal.

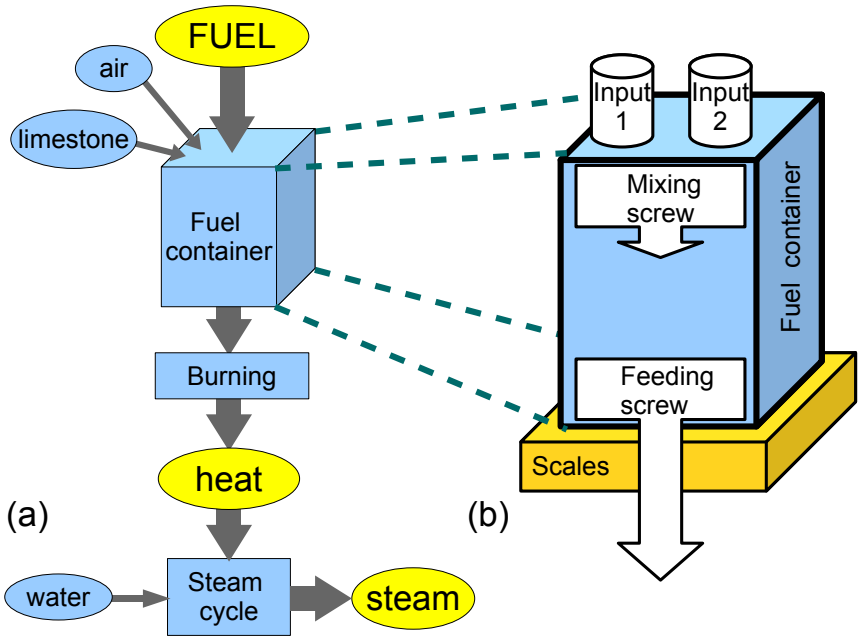


Figure 6.1: (a) operation of CFB boiler; (b) fuel container of CFB boiler.

The mass of the fuel inside the container is measured by a mechanical scale, see Figure 6.1 (b). During the burning stage the mass of fuel inside the container decreases. As new fuel is added to the container (the burning process continues), the fuel feeding stage starts that is reflected by a rapid mass increase. The automatically available mass signal is noisy, as depicted in Figure 6.2. The start and the end time of the fuel input process is not available from the sensors as a direct measurement.

6.1.2 Properties of the signal

As a result of the described processes, the fuel mass signal has the following characteristics:

1. There are two types of change points: an abrupt change from consumption to fuel input and slower but still abrupt change from input to consumption.

2. There are asymmetric outliers, oriented upwards. In online settings the outliers can be easily mixed with the changes from burning to feeding.

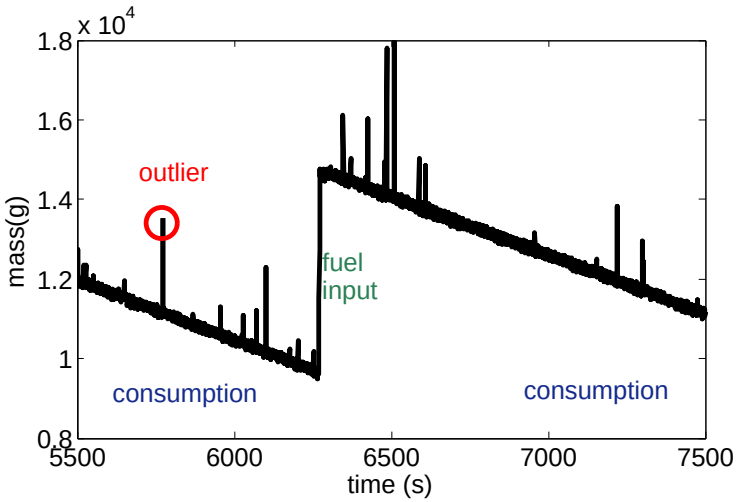


Figure 6.2: Noisy fuel mass signal, measured by the scales.

3. There is a symmetric high frequency signal noise.

Correspondingly, there are three main sources of changes in the signal.

First, fuel feeding is a manual and non standardized process, which is not necessarily smooth, it can have short interruptions (see Figure 6.3). Each operator can have different habits. Besides, the feeding speed depends on the type of fuel.

Second, the feeding screw rotation adds noise to the measured signal. Besides, fuel particle jamming often happens, slowing down the screw for some seconds and distorting the signal estimate. Therefore, the reported mass inside the container is not accurate, the signal contains extreme upward outliers in the original signal, that can be seen in Figure 6.2.

Third, there is a low amplitude rather periodic noise, which is caused by the mechanical rotation of the system parts. These amplitudes may become higher depending on the burning setup.

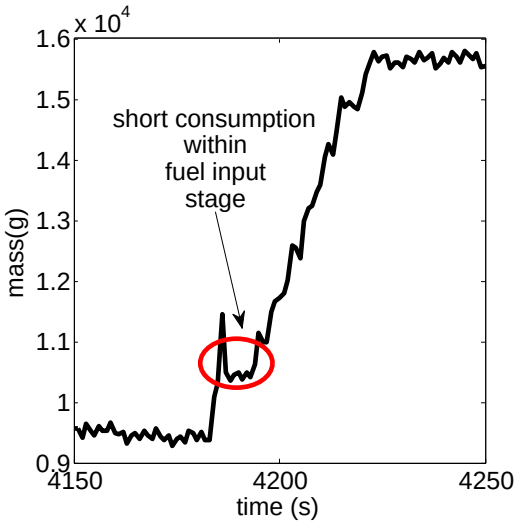


Figure 6.3: An example of a short consumption within the fuel input

6.1.3 Challenges of the signal estimation

Algorithmic change detection and signal estimation is not trivial as it might seem from the visual inspection of the signal.

The outliers are asymmetric (oriented only upwards), that raises two challenges. Firstly, if averaged directly, the signal would have a positive bias. The noise and outliers do not sum to zero with respect to the *true* signal. Secondly, an outliers can be mixed with the start of fuel input stage when observing online, before a few more seconds of signal are seen.

Besides, there are short consumption periods within the fuel input stages, due to possible pauses in an input process, which depend on human operator behavior. These interruptions can vary from 5 to 20 seconds are difficult to identify.

For evaluation of the performance of signal estimators labeled data is needed. There is no hard evaluation method for the actual amount of fuel present. It could be generated by the domain experts. It is not trivial to extract the actual signal even *offline*, since the data includes the effects of external influences.

Our ultimate goal is to learn an accurate online estimation of the signal given that we can

1. catch and handle the changing behavior due to a process change, and
2. ignore the noisy patterns generated by anomalous behavior or the influence of moving parts.

6.1.4 Related work

Change detection [14] and outlier detection [37] are well established research areas. However, the boiler problem exposes specific combination of change points and outliers at which existing change detection methods may fail. Statistical change detection methods, which are based on comparing pieces of raw data (e.g. [21]) do not take signal trends into account, which contain significant part of discriminatory information in the boiler problem. The noise and outliers are not normally distributed making it hard to use statistical methods that assume a particular distribution of the data [12]. Model output based change detection (e.g. [74]) is not directly suitable for this problem due to the nature of the signal: noise, trends and specific outliers. Consumption and fuel input stages, which are observed in the fuel mass signal, are very different in nature and timing.

Data mining approaches can be used to develop better understanding of the underlying processes in CFB boilers, or learning a model to optimize its efficiency [162]. Fundamental studies develop mathematical models for boiler operation [92, 126, 158, 182], incorporating operational parameters in the models.

A straightforward approach to non stationary time series prediction would be SARIMA model [39]. Seasonal periodicity is expected there, but in case of boiler mass flow prediction fuel feeding periods are not regular. The patterns might differ in every feeding round as well as the periods between two feedings. Our preliminary experiments confirmed that SARIMA indeed did not give satisfactory results.

In this Chapter we present an online signal estimation method, which takes into account the properties of mass flow signal (noise, trends, specific outliers, switch between operational stages). We take a data driven approach, we use no additional input data from the boiler except the noisy signal itself. The method is equipped with a change detection, tailored for trendy noisy drifting signal.

Change detection is needed to drop out the old signal from the training sample of the predictor. Experiments with two different fuel types are carried out.

6.2 The Model for Online Mass Flow Estimation (OMFP)

Given the challenges discussed in previous section, we develop a method for online estimation of mass flow (OMFP). The model has three main design elements: base learner, outlier elimination mechanism and change detection mechanism. Base learner models the signal. Outlier elimination is needed for cleaning the historical data. Change detection signals the need for model retraining.

The model needs to be trained and evaluated before employing online. For that we need to construct the 'ground truth' of a signal, since the 'true truth' is not available. In fact if it was technologically available, there would be no mass flow estimation task. However, when constructing the 'ground truth' we can make use of the training data *offline*.

6.2.1 Setup

Let's have the original signal $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_n)$, $\mathbf{x} \in \Re^1$. Having $\mathbf{x}$ as input we want to obtain the actual mass flow signal $\mathbf{y}$ that can be achieved by learning a functional mapping of noisy sensor measurements to the true actual signal, so that $\mathbf{y} = \mathcal{F}(\mathbf{x})$.

Concept drift in a signal is expected, since it is not known exactly when fuel input starts, what exact proportions of fuel are used and what are the positions of mechanical screws. Once a change in the system stage happens the functional mapping $\mathcal{F}$ might become outdated and needs to be retrained. The boiler data exhibits sudden changes. Thus change detection and retraining of the base model, after a change is detected, can be relevant.

The intuition behind the model is the following. At each point in time t we fit a model $\mathcal{F}(\mathbf{x})$, using all or a subset of the historical data $\mathbf{x}$. The historical data is cleaned before use (elimination of outliers). If a change is detected in a stream, the old portion of the historical data is dropped out and new model is trained using the data after the change.

A simplified estimation procedure is presented in Figure 6.4, the steps are explained in more detail in the following section.

ONLINE MASS FLOW PREDICTION (OMFP) FOR A TIME POINT $t+1$

input: historical signal $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$.

1. Eliminate the outliers from $\mathbf{x}$ to obtain $\mathbf{x}^* = \{\mathbf{x}_1, \ldots, \mathbf{x}^*_{oi}, \ldots, \mathbf{x}^*_{oj}, \ldots, \mathbf{x}_t\}$, where $\mathbf{x}^*_{oi} = \frac{\mathbf{x}_{i-1}+\mathbf{x}_{i+1}}{2}$ is a replacement for an outlier.
2. Find the last change point κ.
3. Learn the model $\mathcal{F}(\mathbf{x})$, using $\{\mathbf{x}_\kappa, \ldots, \mathbf{x}_t\}$ as a training sample.
4. Cast the prediction $\hat{\mathbf{y}}_{t+1} = \mathcal{F}(\mathbf{x}_{t+1})$.

output: $\hat{\mathbf{y}}_{t+1}$.

Figure 6.4: Online Mass Flow Estimation.

6.2.2 OMPF design elements

OMPF has three main design elements, which correspond to the steps presented in Figure 6.4: elimination of outliers, change detection and base learner.

6.2.2.1 Elimination of Outliers

We have the domain knowledge that the outliers are asymmetric (oriented upwards). Based on visual inspection they exceed the moving average of a signal significantly. This suggests, that a simple outlier detection based on signal comparison to a threshold Tr_{out}, would work well. Since the signal is noisy, we use a moving average $\bar{\mathbf{x}}_t^{\,'Lout}$ instead of a pure signal $\mathbf{x}_t$ for comparison to a threshold. L_{out} is a lag for a moving average.

A threshold Tr_{out} and and the lag L_{out} for calculating a moving average need to be predefined by a designer using domain expertise or trained offline in advance.

Note that in an online setting the nearest neighbors are available only from the past, but not from the future. Thus, a moving average is asymmetric with respect to time t:

$$\bar{\mathbf{x}}_t^{'Lout} = \frac{\mathbf{x}_{t-Lout+1} + \mathbf{x}_{t-Lout} + \ldots + \mathbf{x}_t}{L_{out}}. \tag{6.1}$$

The start of a fuel input stage, can be easily mixed with an outlier. Thus an outlier can be confirmed only after some time has passed since it occurred.

When outliers are detected, we replace them with an average of the two available nearest neighbors to clean the historical signal: $\mathbf{x}_t^* = (\mathbf{x}_{t-1} + \mathbf{x}_{t+1})/2$.

6.2.2.2 Change Detection

The data exhibits trends, therefore change detection based on comparison of raw data subsets fails. Our task is to detect the start and the end of fuel input stages. They are characterized by a steep increase in the signal value. An intuitive solution to deal with a trend is to use the first order differences of the signal $\Delta_t^{(1)} = \mathbf{x}_t - \mathbf{x}_{t-1}$ and threshold these values. If $\Delta_t^{(1)} > Tr_{ch}$ the system is at fuel input stage, if $\Delta_t^{(1)} < -Tr_{ch}$ the system is at consumption stage. In a noise free environment $Tr_{ch} = 0$.

Unfortunately, due to signal noise, the stages are undistinguishable directly (see Figure 6.5 (a)). Obvious solution would be to replace the original signal with a moving average, before calculating the first order differences. This already makes fuel input regions apparent, but still noisy (see Figure 6.5 (b)).

Thus we propose using l^{th} order differences of the moving averages.

$$\Delta_t^{(l)} = \bar{\mathbf{x}}_t^{'Lch} - \bar{\mathbf{x}}_{t-l}^{'Lch}, \tag{6.2}$$

here $\bar{\mathbf{x}}^{'Lch}$ is an asymmetric moving average with a lag L_{ch}.

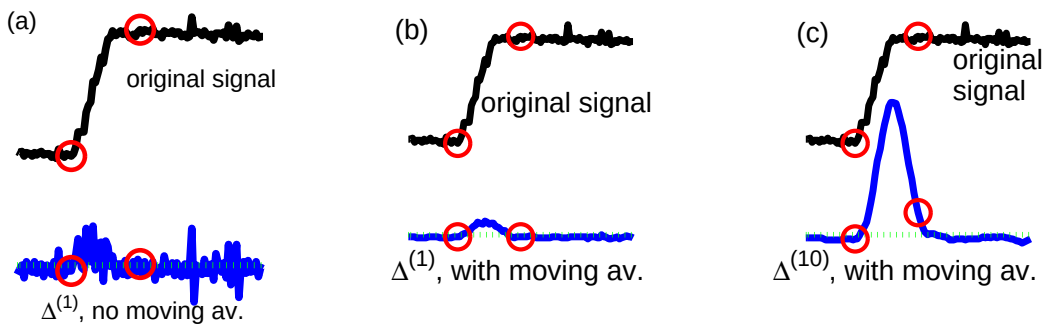


Figure 6.5: Change detection. Dashed line (green) is the threshold for a change. Circles indicate estimated true change.

The more noisy is the signal, the larger lag is needed. Both lags the difference lag l and the moving average lag L_{ch} as well as the detection threshold Tr_{ch} are to be preset by a designer using a domain knowledge or they can be learned offline in advance. In this case study we use $l = 10$, the transformed signal is depicted in Figure 6.5 (c).

Tr_{ch} for changes is not to be mixed with Tr_{out} for outliers. The first is applied to a differentiated signal, while the second is applied to raw data.

Change detection might be equipped with a prior probability of switching the stages, based on the total amount of mass present in the container.

6.2.2.3 The Base Learner

The functional mapping $\mathcal{F}$ is designed as follows.

We assume that mass flow signal has a nonzero second derivative. It implies that the speed of the mass change depends on the amount of fuel in the container. The more fuel is in the container, the higher is the acceleration, thus the more fuel gets into the screw. The weight of the fuel at higher levels of the tank compresses the fuel in the lower levels and in the screw, the fuel density is increased. Besides, compression and thus the burning speed depends on the type and quality of the fuel.

The signal at time t can be modeled using the following equation:

$$\mathbf{y}_t = \frac{a \cdot t^2}{2} + v_0 \cdot t + m_0 + A \cdot \sin(\omega_{feed} \cdot t + \alpha_{feed}) + B \cdot \sin(\omega_{mix} \cdot t + \alpha_{mix}) + e(t), \quad (6.3)$$

where $\mathbf{y}_t$ denotes the output of the scales at time t, a is acceleration of the mass change, v_0 stands for the speed of the mass change at time t_0, m_0 is the initial mass at time t_0; A and B, ω_{feed} and ω_{mix}, α_{feed} and α_{mix} are amplitude, frequency and phase of the fluctuations caused by feeding and mixing screws, respectively; $e(t)$ denotes the random peaked high amplitude noise caused by the jamming of the fuel particle at time t. We assume $t_0 = \kappa$ was the time of switch in the stages.

Since we are not interested in estimating the signal generated by the oscillations of the screw and the noise signal, we make a simplifying assumption that these parts can be treated as a signal noise. Thus we simplify to the following:

$$\hat{\mathbf{y}}_t = \frac{a \cdot t^2}{2} + v_0 \cdot t + m_0 + E(t), \quad (6.4)$$

where $E(t)$ is the aggregated noise component and the other terms are as in (6.3).

In our estimator we use a linear regression approach with respect to the second order polynomial given by (6.4). The model is inspired by the domain knowledge of the underlying process in the boiler, therefore seem more reasonable choice than alternative autoregressive models.

6.2.3 Training the model

Two parts of training OMPF can be distinguished: online and offline. Online training refers to retraining the base learner at each time point t, which is shown in Equation (6.4). That follows the sequential learning framework, which was discussed in 2.1. Offline training refers to setting the model parameters $Tr_{out}, L_{out}, Tr_{ch}, L_{ch}, l$. They can be chosen using the domain knowledge or learned on an offline training set.

6.2.3.1 Online retraining

The Equation (6.4) gives an intuition for the base learner. The actual model for time t is

$$\hat{\mathbf{y}}_t = a_t^{(2)}\mathbf{x}_t^2 + a_t^{(1)}\mathbf{x}_t + a_t^{(0)}, \tag{6.5}$$

here $a_t^{(2)}, a_t^{(1)}, a_t^{(0)}$ are the polynomial coefficients to be learned using historical data. Thus in order to estimate a signal value $\hat{\mathbf{y}}_{t+1}$ we learn the model coefficients using the historical signal which is cleaned (outliers) and cut (change). The training data is $\mathbf{x}^* = \{\mathbf{x}_\kappa^*, \ldots, \mathbf{x}_t^*\}$, where $*$ means that the outliers were removed and κ is the last detected change point. The actual fitting of the data is done in least squares.

6.2.3.2 Setting the Parameters

In this case study we prefixed the offline parameters using an allocated training set, which will be presented in Section 6.3 along with the experiments.

6.2.4 Verifying the model

The mass flow prediction is an unsupervised learning task in a way that the need for prediction arises from the fact that there is no method to measure *the ground truth*. However, to verify the validity of the model we still need a benchmark.

To obtain an approximation to *the ground truth* we use all the data set at once offline. The procedure is structurally similar to online learning, presented in Figure 6.4. We identify and remove the outliers, detect the change points and then fit the base model.

Outlier removal procedure is the same as in an online case (Section 6.2.2.1), but now we can use symmetric moving averages w.r.t. to time t, since the 'future data' is available. The symmetric moving average is defined as follows:

$$\bar{\mathbf{x}}_t^{'Lout} = \frac{\mathbf{x}_{t-\frac{Lout-1}{2}} + \ldots + \mathbf{x}_t + \ldots + \mathbf{x}_{t+\frac{Lout+1}{2}}}{L_{out}}. \tag{6.6}$$

For offline change detection employ different approach than online, since here we can employ the 'future data'. We use ADWIN method [21], which showed to be robust to false positives in semi-online settings [12]. We can not use it in online settings, because the lag needed to detect the change after it happened is too large.

Given a sequence of signals, ADWIN checks whether there are statistically significant differences between the means of each possible split of the sequence. If statistically significant difference is found, the oldest portion of the data backwards from the detected point is dropped and the splitting procedure is repeated recursively until there are no significant differences in any possible split of the sequence. More formally, suppose m_1 and m_2 are the means of the two subsequences as a result of a split. Then the criterion for a change detection is $|m_1 - m_2| > \epsilon_{cut}$, where

$$\epsilon_{cut} = \sqrt{\frac{1}{2m} \log \frac{4n}{\delta}}, \tag{6.7}$$

here m is the harmonic mean of the windows $m = \frac{1}{\frac{1}{n_1}+\frac{1}{n_2}}$, n is total size of the sequence, while n_1 and n_2 are sizes of the subsequences respectively. Note that $n = n_1 + n_2$. $\delta \in (0, 1)$ is a hyper-parameter of the model. In our experiments we used $\delta = 0.3$, $n = 200$ which were set during the preliminary experiments using the training data.

ADWIN identifies change points approximately. To get the exact change points we sequentially search the neighborhood maximum and minimum values.

We approximate the 'ground truth' which we associate as y using a symmetric moving average of a cleaned signal. We stop the moving average at the identified change points, to keep them originally sharp. The moving average lag parameter L_{gr} is chosen by a visual inspection of several options. We validate the obtained approximation to the 'ground truth' by visual inspection of a domain expert.

Table 6.1: Characteristics of the data sets.

Name	Size of feeds	Number	Fuel
A	50 977	24	bio
B	25 197	9	bio
C	25 197	6	coal

6.3 Experimental Evaluation

We conduct numerical experiments to test for prediction accuracy and for change detection accuracy. We chose an asymmetric moving average prediction as a 'naive' online method to 'benchmark the performance.

6.3.1 Data sets

We use three mass signal data sets referred as A,B and C. The data was obtained from a pilot boiler operated in VTT Technical Research Center of Finland. A summary of the data sets is provided in Table 6.1 and they are plotted in Figure 6.6. Different composition of fuel was used in the three data sets. The total length of A is different from B and C.

Data set A is used for training the model and selecting the model parameters. Data sets B and C are used as testing sets, the model trained on A with the same set of parameters is applied. Note that the level of noise and outliers in the data sets are different. B and C represent two fuel tanks, operating in parallel, therefore there are nearly twice as much of noise sources as in A.

Using the training data set A we construct a representation of an average fuel input stage pattern, which is depicted in Figure 6.7. This pattern is obtained by separating the approximated 'ground truth' data into sections at the change points. Then the sections are matched at the fuel input points. The pattern shows an average fuel input stage, where the start point is rather sharp, while the end point (switch to a consumption stage) is smooth and not so clearly

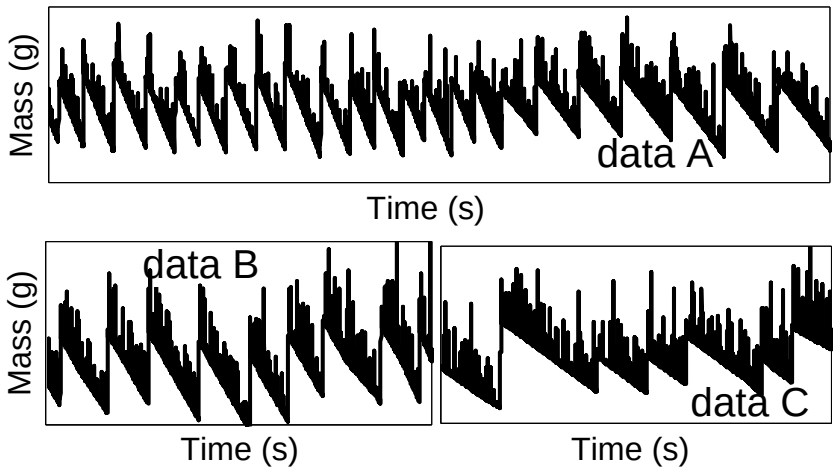


Figure 6.6: The complete data sets A, B, and C used in the experiments.

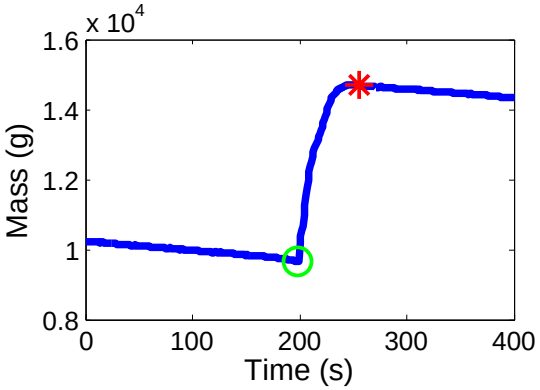


Figure 6.7: An average pattern of the fuel input stage.

6.3.2 Experimental setup

We run experiments using different delays. Delay is the time when the estimation is output as compared to the true signal time. For example, a signal $\mathbf{x}_t$ is measured at time t, but the estimation $\mathbf{y}_t$ is output at time $t + 2$, then the delay is $D = 2$. Longer delay allows more accurate estimate, since more symmetric moving averages instead of asymmetric can be used, the change detection can be verified using the next signals. The change is verified if it remains for not less than D time steps.

$D = 0$ means no delay, the estimation is output right away without seeing any more signal values. We also run a group of experiments for signal prediction. That means $D = -1$. We train the model to output $\mathbf{y}_{t+1}$ given the signal value $\mathbf{x}_t$. D is not to be mixed with l, which is a lag used by change detection method itself (Section 6.2.2.2). The domain experts suggested that maximum possible delay (D) in prediction could be 10 sec.

Once a change is detected, old portion of the data is dropped out of the training sample. We do not use the 2^{nd} order polynomial model before we accumulate a certain number of training samples. The following heuristics is employed. For the first 2 samples we use simple moving average of 1 rule: $\mathbf{x}_{t+1} = \mathbf{x}_t + s$, where s is a linear intercept term obtained using an average feeding stage pattern of the training data (A), which is presented in Figure 6.7. For the consumption stage $s_c = -2$, for the fuel input stage $s_f = 81$. If $2 - 10$ historical data points are available after a change, we fit the 1^{st} order polynomial model, since the 2^{nd} order approximation is too noisy with this few amount of points. Having over 10 data points we start using the 2^{nd} order polynomial.

The set of parameters is the the following.

- $Tr_{out} = 400, L_{out} = 9, Tr_{ch} = 100, L_{ch} = 8, l = 10$ were selected offline, using training data set A.
- $a_t^{(2)}, a_t^{(1)}, a_t^{(0)}$ are learned online every time step using the available historical data.
- A lag L_{gr} for approximation to the 'ground truth' was set on training data set A. The same parameters $wTr_{out} = 400, L_{out} = 9, Tr_{ch} = 100, L_{ch} = 8, l = 10$ were used.

6.3.3 Results of the online estimation

We evaluate the prediction accuracy comparing the mean absolute errors (MAE) with respect to the approximated 'ground truth' (described in Section 6.2.4). The results are listed in Table 6.2. We present MAE for the whole data sets and then present MAE's for the fuel input and the consumption stages separately.

Delay 'p' means prediction of change one second ahead, 0 means real time signal estimation and $2, 4$ and 9 means estimation with the respective delay in seconds.

We compare the accuracy of seven alternative methods or parameterizations. 'MA3', 'MA5' and 'MA10' stand for simple estimation by asymmetric moving averages, the number indicates the lag. 'win50' uses the 2^{nd} order polynomial presented in Section 6.2.2, but instead of change detection a simple moving window of the 50 last instances is used for the model training at each time step. 'all' uses the 2^{nd} order polynomial with no change detection at all, it retrains the model at every time step. Finally we include a benchmark of the 2^{nd} order model assuming known change points ('known'). We assume with this method that the change detection is 100% accurate.

MAE in 'overall performance' is rather close to MAE in the 'consumption stages' and very different from the 'fuel input stages'. This is because of uneven distribution of the stages in the data. The 'consumption stages' comprise less than 2% of the data.

6.3.4 Results of the change detection

We report the performance of the change detection in online settings in Table 6.3. For each method we present confusion matrices of detecting sudden changes in the fuel input (INPU) and consumption (CONS) stages and detecting of outliers (OUTL). For change detection we allow 10 sec. deviation. If a change is detected within the allowed region it is considered as identified correctly. We require the outlier detection to be precise.

We visualize change and outlier detection in Figure 6.8. The solid (blue) lines represent the true positives (TP) divided by the actual number of changes, the dashed (red) lines represent the number of false positives (FP) divided by the actual number of changes. The dotted black lines show the level of true changes (i. e. 24 change points for data set A, 9 for B, 6 for C).

The number of FP is decreasing along with the increase in allowed prediction delay. A delay allows to inspect the following signal values after the detected change and if necessary cancel the alarm within the delay period.

Table 6.2: Mean average prediction accuracies. The best accuracies for each delay are **bold**; the best overall accuracy over a single experiment is underlined.

Delay	p	0	2	4	9	P	0	2	4	9	P	0	2	4	9
Data			A					B					C		
Overall performance															
OMFP	**34.1**	**29.4**	**27.8**	**27.6**	**29.0**	**23.8**	**20.9**	**16.6**	**16.3**	**31.4**	**12.9**	**13.0**	**10.3**	**10.1**	16.3
MA3	64.0	64.0	66.4			48.5	47.2	46.9			36.3	35.6	35.2		
MA5	63.1	51.9		39.9		49.7	45.3		41.7		35.9	33.9		32.5	
MA10	59.1	54.8			33.2	58.1	53.7			34.9	39.3	37.2			28.5
win50	53.0	45.0	44.4	44.4	44.4	40.3	34.3	32.0	32.0	32.0	19.7	16.7	15.2	15.2	**15.2**
all	1271	1269	1267	1265	1261	1313	1310	1308	1306	1301	1022	1021	1019	1019	1016
known	34.8	32.0	30.6	31.3	41.8	50.7	47.9	45.1	44.6	65.5	18.0	16.5	15.7	16.3	22.0
Fuel input stages															
OMFP	463	325	229	231	321	1531	952	601	682	968	519	334	182	180	325
MA3	**308**	**181**	**115**			**733**	**510**	**434**			**260**	**163**	**119**		
MA5	438	294		**77**		1118	713		**359**		374	236		**105**	
MA10	781	640			**60**	2081	1714			**171**	714	578			**61**
win50	751	646	577	577	577	1867	1602	1248	1248	1248	860	735	645	645	645
all	1757	1753	1748	1744	1731	3259	3253	3248	3242	3225	2264	2259	2255	2250	2237
known	441	315	236	244	290	1493	924	561	594	752	464	306	249	269	296
Consumption stages															
OMFP	30.0	28.7	28.5	**28.2**	**28.1**	**22.1**	**19.7**	**16.9**	**17.2**	**34.9**	**10.7**	**11.4**	**10.5**	**11.3**	19.1
MA3	60.8	62.4	65.8			48.1	46.9	46.8			35.6	35.0	34.8		
MA5	57.7	48.4		39.4		48.3	44.4		41.5		34.5	32.8		32.1	
MA10	48.3	45.9			32.8	54.6	50.6			**34.9**	36.0	34.3			28.3
win50	42.8	37.0	37.1	37.7	39.1	39.4	33.1	32.4	33.3	35.5	15.6	12.9	12.8	13.8	**16.3**
all	1264	1262	1261	1260	1257	1320	1317	1315	1314	1311	1015	1013	1013	1013	1013
known	**29.1**	**28.0**	**28.1**	29.3	40.7	50.6	47.7	45.7	46.0	69.1	16.2	15.0	15.5	16.9	25.1

Table 6.3: Confusion matrices of detecting changes to fuel input (INPU) and consumption (CONS) stages and outlier detection (OUTL). P - positive, N - negative, T - true, F - false.

Training data set A									
delay	**INPU**	**P**	**N**	**CONS**	**P**	**N**	**OUTL**	**P**	**N**
pred.	T	24	50946	T	12	50934	T	659	49784
	F	26	0	F	38	12	F	543	10
0	T	24	50946	T	12	50934	T	659	49784
	F	26	0	F	38	12	F	543	10
2	T	24	50967	T	10	50953	T	659	49783
	F	5	0	F	19	14	F	544	10
4	T	24	50969	T	10	50955	T	658	49783
	F	3	0	F	17	14	F	544	11
9	T	24	50972	T	8	50956	T	660	49782
	F	0	0	F	16	16	F	545	9
Testing data set B									
pred.	T	6	25162	T	2	25158	T	475	24597
	F	26	3	F	30	7	F	104	21
0	T	6	25162	T	2	25158	T	475	24597
	F	26	3	F	30	7	F	104	21
2	T	6	25165	T	2	25161	T	477	24597
	F	23	3	F	27	7	F	104	19
4	T	6	25165	T	2	25161	T	477	24597
	F	23	3	F	27	7	F	104	19
9	T	6	25183	T	2	25179	T	489	24594
	F	5	3	F	9	7	F	107	7
Testing data set C									
pred.	T	6	25176	T	2	25172	T	362	24750
	F	15	0	F	19	4	F	75	10
0	T	6	25176	T	2	25172	T	362	24750
	F	15	0	F	19	4	F	75	10
2	T	6	25177	T	1	25172	T	364	24750
	F	14	0	F	19	5	F	75	8
4	T	6	25177	T	1	25172	T	364	24750
	F	14	0	F	19	5	F	75	8
9	T	6	25191	T	1	25186	T	372	24746
	F	0	0	F	5	5	F	79	0

The number of false negatives (FN) is relatively large. However, this does not mean that the changes from feed to burn were not detected at all. In this setting it means that they were not detected in time (within 10 sec. interval).

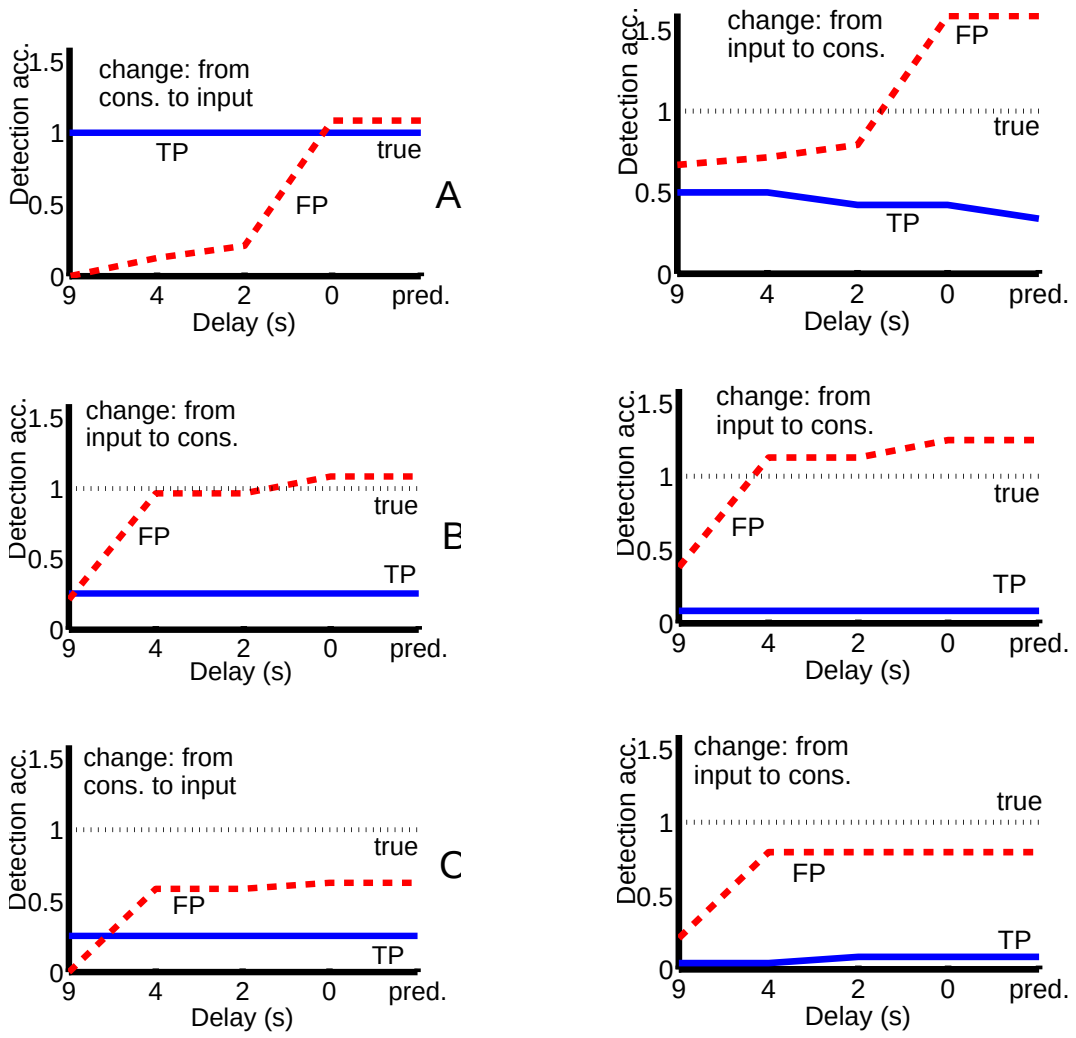


Figure 6.8: Change detection accuracy as a function of the prediction delay for A, B and C data sets.

6.3.5 Discussion

OMFP outperforms the competitive methods in terms of overall accuracy. However, for the fuel input stage, a simple moving average is the most accurate. Note that the approximation to the 'ground truth' was constructed using moving averages, thus it could be expected that moving average performs well in this test setup.

OMFP method performance gets worse having a very large delay in predictions. This is likely due to a fixed lag size for moving average calculations, we are

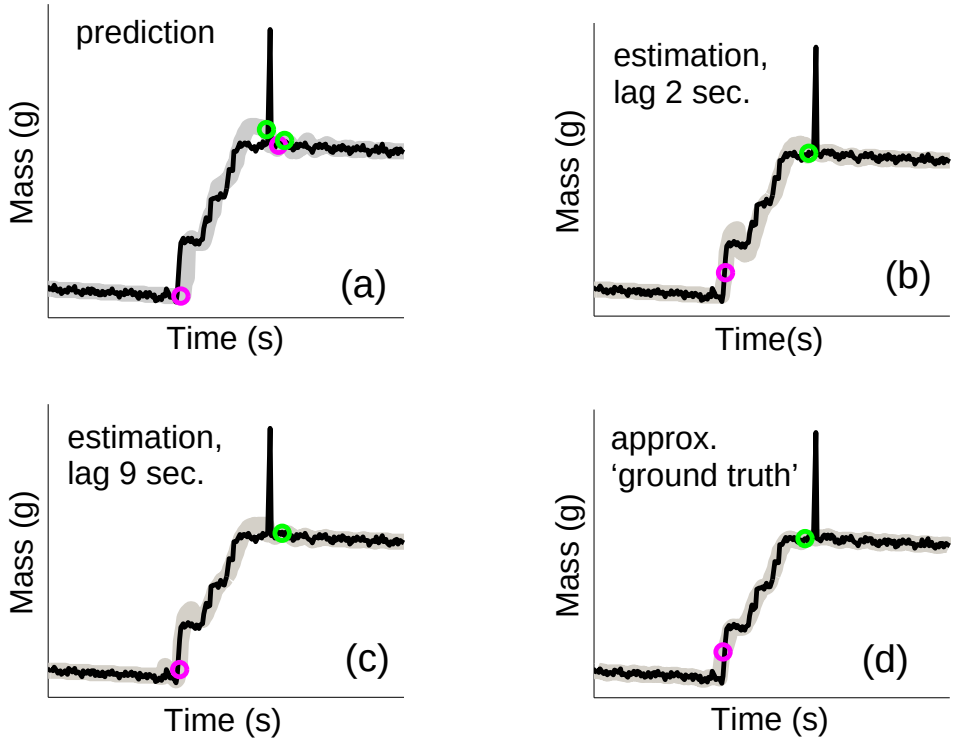


Figure 6.9: The original signal (black) and OMFP estimation (gray).

using the same parameter settings for all the experiments.

Degradation of OMFP performance along with the increase in prediction delay also suggests, that there might be more accurate cutting points than just the change points themselves. Note that having a delay we allow canceling the detected changes.

In Figure 6.9 an extraction from the prediction outputs is provided. In (a) the prediction follows previous points almost as a straight line. It is reasonable to expect, since in the fitted function $a^{(2)}$ coefficient is mostly 0. Estimation with a delay $D = 2$ (b) is more curvy than $D = 9$ (c) likely due to more change points identified and therefore more cuts in history.

Overall accuracy indicates that OMFP method is more accurate than moving average or alternative methods in the consumption stage, while separate handling of prediction in the fuel input and consumption stages might be advantageous.

6.4 Conclusion

The fourth research question (RQ4) asked, how the considered adaptive training methods can be extended to develop a tailored training set selection method for a selected industrial application (industrial boiler)?

We developed and experimentally evaluated an online method for mass flow estimation during the boiler operation. We evaluated the performance of the method on three real data sets, including two distinct fuel types and two distinct operating stages (single vs multiple fuel).

One of the challenges in this task is coming up with approximation for constructing the 'ground truth' signal, which we handle by a combination of moving average and responding for change and outlier points. We use this approximation to evaluate the performance of the online predictors.

Change detection is sufficiently accurate in transition from the burning to the feeding stage, where the incline in signal is rather sharp. The reverse detection accuracy is reasonable for the final prediction accuracy. OMFP outperforms the comparative methods in terms of overall accuracy.

The developed adaptive method for online estimation of the mass flow for an industrial boiler, which operates using a changing mix of fuel and changing input styles, allows to achieve more accurate estimates than using no adaptivity to changes and this way allows to improve the control system of the boiler.

Chapter 7

Conclusion and Future Research

> I can hold a note for a long time - actually, I can hold a note forever - but, eventually, that's just noise. It's the change we're listening for, the note coming after, and the one after that. That's what makes it music.
>
> Lorne the Host ('Angel', 1999)

We addressed a problem of supervised learning over time when the data is changing (concept drift). We focused on adaptive training methods, which are based on selective training set formation. In the stationary setting the more data is at hand, the more accurate model can be trained. In the changing environment an old data decreases the accuracy. In such a case only a subset of the historical data might be selected to form a training set.

In the problem statement we asked: how to advance adaptive supervised learning methods and develop training set formation algorithms based on these methods, in order to improve the accuracy of classification and prediction models under concept drift. We addressed the problem following the three main drift types: sudden, gradual and recurring. Sudden drift happens when one data source is instantly replaced by another. The data generated by the new

source is scarce immediately after the drift. Gradual drift happens when one data generating source is gradually taken over by another and for some time period both sources are active. Finally, reoccurring concepts is a pattern of concept drift where several sources are switching and the past sources might reoccur, but it is not certain when exactly and in what form they will repeat.

The thesis improves the training strategies under sudden, gradual and recurring concept drifts. Four adaptive training set formation algorithms have been developed and experimentally validated, which allow to increase the generalization performance of the base models under each of the three concept drift types. Experimental evaluation using generated and real data confirms improvement of the classification and prediction accuracies as compared to using all the historical data as well as the selected existing adaptive learning algorithms from the recent literature. A tailored method for an industrial boiler application, which unifies several drift types, has been developed.

Next we provide the four main conclusions of the thesis, corresponding to the four research questions stated in Introduction 1. We finish the thesis by outlining related future work perspectives.

7.1 Conclusion

(1) There was no explicit distinction between the change point and the start of the training window in supervised learning under concept drift. The historical data was dropped as soon as a sudden change was detected. In Chapter 3 we made an explicit theoretical distinction between the sudden change point and the training window. We demonstrated that the impact of taking the difference into account to the classification accuracy is increasing along with the more complex data. Based on the theoretical distinction we developed a training window resizing algorithm WR* and demonstrated an improvement in classification accuracy as compared to the existing algorithms for variable window size, which do not make this distinction. **Theoretical distinction between the training window and the change point when determining a variable window size allows to improve generalization performance under sudden concept drift.**

(2) So far either temporal instance selection (training windows) or instance selection in feature space was used for learning under concept drift. In Chapter 4 we developed a new distance measure unifying the distances in time and feature space for training set selection. We argued that both criteria are relevant under gradual concept drift and demonstrated this on real datasets from six domains. Using the new distance measure we developed a family of training set selection algorithms FISH. The three algorithms FISH1, FISH2 and FISH3 differ in determining the training set size and the proportion of time and space in the developed distance measure. In FISH2 only set size is learnable online, while in FISH3 both set size and the proportion of time and space are learnable online. The extensive numerical experiments using four alternative base classifiers and two alternative distance in space measures on six real datasets demonstrated statistically significant improvement in the classification accuracy as compared to the two existing adaptive algorithms, which use only time and only space criterion. **Integration of similarity in time and feature space when selecting training set allows to improve generalization performance as compared to using only time or only space criterion under gradual concept drift.**

(3) Using a real problem of food sales prediction, for which recurring concepts are relevant, we developed and experimentally validated a contextual method CAPA for training set formation in Chapter 5. We demonstrated that identifying and learning to recognize the types of historical behavior allows to form a training set in a way that this historical information contributes to the present prediction accuracy. CAPA forms a training set interactively, based on the type of historical behavior, which is determined employing structural features. We showed that online reassignment of the categories increases the prediction accuracy. The experiments demonstrated 5% improvement in the testing prediction accuracy as compared to the baseline prediction, which is relevant for the field applications. **Contextual training set formation, while connecting the types of historical sales with the training set formation strategies and learning to recognize the types online using structural features, allows to improve generalization performance in food sales prediction task, where reoccurring concepts are expected.**

(4) We developed a mass flow estimation method OMFP for an industrial boiler in Chapter 6. OMFP takes into account concept drifts using a tailored train-

ing window strategy. **The developed adaptive method for online estimation of the mass flow for an industrial boiler, which operates using a changing mix of fuel and changing input styles, allows to achieve more accurate estimates than using no adaptivity to changes and this way allows to improve the control system of the boiler.**

The developed algorithms WR* and FISH are relevant for supervised learning tasks from different domains, where correspondingly sudden and gradual drifts are expected over time. Sudden drift is particularly relevant for network intrusion detection, financial fraud monitoring, navigation, demand prediction tasks. Gradual drift is particularly relevant for marketing, recommender systems, online shops, adaptive tutoring systems. CAPA is not specific to food sales, it is applicable for prediction in other domains, for instance other sales, demand prediction, geographic crime maps, bus travel time prediction. OMPF is tailored to a boiler case and needed for its control systems. However, OMFP can be adapted to different tasks related to burning or fuel consumption processes, for example monitoring of the fuel consumption for cars, based on traffic, fuel type.

The thesis contributes to the understanding concept drift problem in general and training set selection under concept drift in particular.

7.2 Future Research

We investigated adaptive training set formation following a range of drift types: starting from sudden via gradual to reoccurring concepts. In this final section we highlight prospective directions for further research, related to the thesis topic and findings.

The thesis work was concentrated on training set formation via instance selection. In Chapter 5 we considered training set formation in feature space together with an instance selection. However, dynamic feature space formation over time we left out of the scope of the thesis work. There are a few works on dynamic feature space formation [101, 208]. Concept drift might be local not only w.r.t. instance space [199], but also w.r.t. feature space. For instance, in textual data, structured and unstructured documents, web content analysis changes might

affect only a part of the feature space, related to the changed vocabulary. We believe that dynamic approach to training feature space would help to preserve historical information which is stable and discard irrelevant information more selectively and this way increase flexibility in adaptation to drifts. Thus we see this direction as interesting and promising for potential extension of the thesis work.

Another promising direction for future research would be instance manipulation for adaptive training set formation. The thesis work was limited to the training instance selection not altering the instances themselves. By instance manipulation we mean forming artificial instances from existing historical data, for example, using noise injection [189], hybrid instances merging or swapping historical instances in the feature space. We expect such strategies could increase flexibility in adaptation to drifts in cases of small sample size, also in cases of gradual drifts where it is hard to distinguish between the sources in the original feature space. In addition, instance formation would be very promising in a light of privacy preserving data mining, where instances correspond to humans and disclosure of data is sensitive.

The third promising direction following from the thesis work is context aware learning, which selects decision making models from a pool, based on the observed context (environmental state). In Chapter 5 we developed a method which forms training set based on the observed historical behavior. This allows to utilize the relevant past information after the concept has changed. The two main challenges in building such context aware learners are: how to define the contextual categories and how to link those categories with the individual learning models. This type of approach can be extended beyond the problem of concept drift, for instance, taking into account geographic context, social context, occupation (work or leisure). Incorporating the contextual information into the learning model is expected to reduce uncertainty in the decision making phase.

Finally, concept drift problems are heterogeneous from the application perspective. In Chapter 2 Section 2.5 we categorized applications in four main categories and identified the desiderata for algorithm performance. We believe that the future research on adaptivity to concept drift has prospects and demand to come closer to specializing in application groups.

Appendix A

Derivations and Computation Details

A.1 The Case of a Sudden Concept Drift

Following Raudys' methodology [170] we derive the expression for the generalization error of the Nearest Mean classifier $C^{(M)}$ trained on a mixture of data generated by a source S_1 (before the drift) and S_2 (after the drift). The classifier is applied for classification of data coming from the source S_2. In the source S_1 the two classes are distributed as $\mathbf{X} \sim \mathcal{N}(\mu_i^{(1)}, I)$, $\mathbf{X} \in \Re^p$, $i = 1, 2$, in the source S_2 the distribution is as $\mathbf{X} \sim \mathcal{N}(\mu_i^{(2)}, I)$, $\mathbf{X} \in \Re^p$, $i = 1, 2$. Assuming equal prior probabilities of the classes, the discriminant function of NMC is

$$q(\mathbf{X}) = \mathbf{X}^{\mathbb{T}} w^E + w_0. \tag{A.1}$$

Here $w^E = \mu_2^{(M)} - \mu_2^{(M)}$ and $w_0 = -\frac{1}{2}(\mu_2^{(M)} + \mu_2^{(M)})^{\mathbb{T}} w^E$, where $\mu_i^{(M)}$ are the means of the classes ω_i, $i = 1, 2$ generated by the sources S_1 and S_2 and $\mathbb{T}$ denotes transpose.

The classification decision is made according to the sign of $q(\mathbf{X})$. The probability that $C^{(M)}$ will misclassify a given data point from the source S_2 is

$$E_M^{(2)} = \frac{1}{2} P\{q(\mathbf{X}) < 0 | \widehat{w}_0, \widehat{w}^E, \mathbf{X} \in \omega_1\} + \frac{1}{2} P\{q(\mathbf{X}) \geq 0 | \widehat{w}_0, \widehat{w}^E, \mathbf{X} \in \omega_2\}. \tag{A.2}$$

Since $\mathbf{X}^{(i)} \sim N(\mu_i, \Sigma)$, the discriminant function as linear form of normally distributed random variables will have normal distribution with:

$$\begin{aligned}\mathcal{E}(q(\mathbf{X})|\widehat{w}_0^{(M)}, \widehat{w}^{E(M)}, \mathbf{X} \in \omega_i) &= \mu_i^{(2)\mathbb{T}}\widehat{w}^{E(M)} + \widehat{w}_0^{(M)} \\ &= (\mu_i^{(2)} - \frac{1}{2}(\mu_2^{(M)} - \mu_2^{(M)}))^{\mathbb{T}}(\mu_2^{(M)} - \mu_2^{(M)}); \\ Var(q(\mathbf{X})|\widehat{w}_0^{(M)}, \widehat{w}^{E(M)}, \mathbf{X} \in \omega_i) & \\ &= \widehat{w}^{E(M)\mathbb{T}}\widehat{w}^{E(M)} \\ &= (\mu_2^{(M)} - \mu_2^{(M)})^{\mathbb{T}}(\mu_2^{(M)} - \mu_2^{(M)}).\end{aligned}$$

Then

$$E_M^{(2)} = \frac{1}{2}\sum_{i=1}^{2}\Phi\left\{(-1)^i\frac{\mathcal{E}(q(\mathbf{X})|\widehat{w}_0^{(M)}, \widehat{w}^{E(M)}, \mathbf{X} \in \omega_i)}{\sqrt{Var(q(\mathbf{X})|\widehat{w}_0^{(M)}, \widehat{w}^{E(M)}, \mathbf{X} \in \omega_i)}}\right\}, \tag{A.3}$$

where Φ is the standard normal cdf.

Having total sample size $2N$ and β proportion of the data from the source S_2 in it, we use the following estimates of the variables: $\mu_i^{(2)} \leftarrow \overline{\mathbf{X}}_{\mathbf{i}}^{(\mathbf{2})} \sim \mathcal{N}(\mu_i^{(2)}, \frac{\mathbf{I}}{\beta N})$, $\mu_i^{(M)} \leftarrow \overline{\mathbf{X}}_{\mathbf{i}}^{(\mathbf{M})} \sim \mathcal{N}(\mu_i^{(M)}, \frac{1}{N}\Sigma^{(x)})$. We showed in Section 3.2 that

$$\begin{aligned}\mu_1^{(2)} &= (\frac{\delta}{2}\cos\varphi, \frac{\delta}{2}\sin\varphi, 0, \ldots, 0)^{\mathbb{T}} = (m_1^{(2)}, m_2^{(2)}, 0, \ldots, 0)^{\mathbb{T}}; \mu_2^{(2)} = -\mu_1^{(2)}, \\ \mu_2^{(M)} &= (\frac{\delta}{2}(1-\beta+\beta\cos\varphi), \frac{\delta}{2}\beta\sin\varphi, 0, \ldots, 0)^{\mathbb{T}}; \mu_2^{(M)} = -\mu_2^{(M)}\end{aligned}$$

.

We assumed the covariance matrix does not change and it is common for both classes. However when the data generated before and after the drift is mixed in a training set, the covariance matrix is no longer identity:

$$\Sigma^{(M)} = (1-\beta)\Sigma^{(1)} + \beta\Sigma^{(2)} + (1-\beta)\mu_1^{(1)}\mu_1^{(1)\mathbb{T}} + \beta\mu_1^{(2)}\mu_1^{(2)\mathbb{T}} - \mu_2^{(M)}\mu_1^{(M)\mathbb{T}} = \mathbf{I} + \mathbf{R}. \tag{A.4}$$

Here $\mathbf{R}$ is a $p \times p$ matrix the following non-zero elements:

$$
\begin{aligned}
r_{11} &= \frac{\delta^2}{4}\beta(1-\beta)(1-\cos\varphi)^2; \\
r_{12} = r_{21} &= \frac{\delta^2}{4}\beta(1-\beta)\sin\varphi(\cos\varphi - 1); \\
r_{22} &= \frac{\delta^2}{4}\beta(1-\beta)\sin^2\varphi
\end{aligned}
$$

We introduce the following notations:

$$
\begin{aligned}
U &= 2\mu_2^{(M)} = (u_1, u_2, 0, \ldots, 0), \qquad \text{(A.5)} \\
Y &= \overline{\mathbf{X}}_{\mathbf{1}}^{(\mathbf{M})} + \overline{\mathbf{X}}_{\mathbf{2}}^{(\mathbf{M})}, Y \sim N\left(\mathbf{0}, \frac{2}{N}\Sigma^{(M)}\right), \\
Z &= \overline{\mathbf{X}}_{\mathbf{1}}^{(\mathbf{M})} - \overline{\mathbf{X}}_{\mathbf{2}}^{(\mathbf{M})}, Z \sim N\left(U, \frac{2}{N}\Sigma^{(M)}\right). \qquad \text{(A.6)}
\end{aligned}
$$

Z, Y are independent as $cov(Z, Y) = \mathbf{0}$. Then,

$$
E_M^{(2)} = \frac{1}{2}\sum_{i=1}^{2}\Phi\left\{-\frac{\mu_1^{(2)\mathbb{T}}Z + (-1)^i Y^{\mathbb{T}}Z}{\sqrt{Z^{\mathbb{T}}Z}}\right\}. \qquad \text{(A.7)}
$$

We can find a random orthogonal $G_{p\times p}$, where $g_{1i} = \frac{z_i}{\sqrt{Z^{\mathbb{T}}Z}}$, $i = 1, \ldots, p$, such that

$$
\begin{aligned}
GZ &= \left(\sqrt{Z^{\mathbb{T}}Z}, 0, \ldots, 0\right)^{\mathbb{T}}, \\
GY &= (y_{11}, \ldots, y_{1p})^{\mathbb{T}} \sim N\left(0, \frac{2}{\beta N}\mathbf{I}\right), \qquad \text{(A.8)} \\
Y^{\mathbb{T}}Z &= Y^{\mathbb{T}}G^{\mathbb{T}}GZ = y_{11}\sqrt{Z^{\mathbb{T}}Z}. \qquad \text{(A.9)}
\end{aligned}
$$

Having the notations, (A.7) can be simplified to

$$
\begin{aligned}
E_M^{(2)} = & \quad \frac{1}{2}\sum_{i=1}^{2}\Phi\left\{-\frac{m_1 z_1 + m_2 z_2}{\sqrt{\sum_{j=1}^{p} z_j^2}} + (-1)^i\frac{y_{11}}{2}\right\} \qquad \text{(A.10)} \\
= & \quad \frac{1}{2}\sum_{i=1}^{2}\Phi\left\{-\frac{m_1 z_1 + m_2 z_2}{\sqrt{z_1^2 + z_2^2 + \sum_{j=3}^{p} z_j^2}} + (-1)^i\frac{y_{11}}{2}\right\}. \qquad \text{(A.11)}
\end{aligned}
$$

Note that $y_{11} \sim N\left(0, \frac{2}{\beta N}\right)$. To simplify the notation we introduce $\Omega_i = \frac{2(r_{ii}+1)}{N}$.

Denote, $\xi_0 = \frac{y_{11}}{\sqrt{\Omega_1}}$, $\xi_0 \sim N(0,1)$, then $y_{11} = \xi_0\sqrt{\Omega_1}$.
Denote $\xi_i = \frac{(z_i - u_i)}{\sqrt{\Omega_i}}$ for $i = 1, 2$, $\xi_i \sim N(0,1)$, then $z_i = \xi_i\sqrt{\Omega_i} + u_i$.
And denote $\xi_j = z_j\sqrt{\frac{N}{2}}$ for $j = 3, \ldots, p$, $\xi_j \sim N(0,1)$, then $z_j = \xi_j\sqrt{\frac{2}{N}}$ and $\sum_{j=3}^{p} \xi_j^{s2} \sim \chi^2_{p-2}$.

Inserting the defined variables into (A.11) we get:

$$
\begin{aligned}
E_M^{(2)} &= \frac{1}{2}\sum_{i=1}^{2} \Phi\left\{ -\frac{m_1(\xi_1\sqrt{\Omega_1} + u_1) + m_2(\xi_2\sqrt{\Omega_2} + u_2)}{\sqrt{(\xi_1\sqrt{\Omega_1} + u_1)^2 + (\xi_2\sqrt{\Omega_2} + u_2)^2 + \frac{2}{N}\sum_{j=3}^{p}\xi_j^2}} + (-1)^i \frac{\xi_0\sqrt{\Omega_1}}{2} \right\} \\
&= \frac{1}{2}\sum_{i=1}^{2} \Phi\left\{ -\frac{m_1\xi_1\sqrt{\Omega_1} + m_1u_1 + m_2\xi_2\sqrt{\Omega_2} + m_2u_2)}{\sqrt{\xi_1^2\Omega_1 + 2u_1\xi_1\sqrt{\Omega_1} + u_1^2 + \xi_2^2\Omega_2 + 2u_2\xi_2\sqrt{\Omega_2} + u_2^2 + \frac{2}{N}\sum_{j=3}^{p}\xi_j^2}} \right. \\
&\quad \left. + (-1)^i \frac{\xi_0\sqrt{\Omega_1}}{2} \right\}.
\end{aligned}
$$

When $N \to \infty, p \to \infty, \frac{N}{p} = const$, we get the following expression for the generalization error:

$$
\begin{aligned}
E_M^{(2)} &= \Phi\left\{ -\frac{m_1u_1 + m_2u_2}{\sqrt{(\Omega_1 + u_1^2 + \Omega_2 + u_2^2 + \frac{2}{N}(p-2))}} \right\} \\
&= \Phi\left\{ -\frac{\delta^2 K}{2\sqrt{(\frac{2}{N}(p + r_{11} + r_{22}) + \delta^2(1-2L))}} \right\} \\
&= \Phi\left\{ (-1)^i \frac{\delta}{2\sqrt{\frac{2p}{N\delta^2K^2} + \frac{L}{NK^2} + \frac{1-2L}{K^2}}} \right\}
\end{aligned} \tag{A.12}
$$

If $K >= 0$ then $i = 1$, otherwise $i = 2$.

Here $K = (1-\beta)\cos\varphi + \beta$, $L = \beta(1-\beta)(1-\cos\varphi)$, N is the number of training instances in *one class*, p is the dimensionality.

A.2 The Case of Incremental Concept Drift

Similarly to the sudden drift case, we derive the expression for the generalization error of the Nearest Mean classifier $C^{(Mk)}$ trained on the mixture of data coming from the sources $S_1, S_2, \ldots, S_k$. The classifier is applied for classification of the data coming from the source S_k. In the source S_j the two classes are distributed as $\mathbf{X} \sim \mathcal{N}(\mu_i^{(j)}, I)$, $\mathbf{X} \in \Re^p$, $i = 1, 2, j = 1, \ldots, k$. We assume equal prior probabilities of the classes.

We analyze the model of two Gaussian classes, rotating counterclockwise V instances per 1^o. The rotation speed would be $v = \frac{1}{V}$. The modeled data stream starts at time t_1. We are interested in finding the generalization error of NMC trained on all the data available from time period $[t_1, t]$, coming from k concepts. At time t we would have $V(k-1)$ data instances from the old sources and $t2 = t \mod V$ instances from the source S_k.

Having the same S_1 as in stationary case, the means of the data coming from the source S_k would be:

$$\mu_1^{(k)} = (\frac{\delta}{2}\cos\gamma_{k-1}, \frac{\delta}{2}\sin\gamma_{k-1}, 0, \ldots, 0)^{\mathbb{T}}; \mu_2^{(k)} = -\mu_1^{(k)}, \tag{A.13}$$

where $\gamma_i = \frac{i\pi}{180}$.

At time t the classifier would be trained on the mixture of the data coming from $S_1, S_2, \ldots, S_k$. The means of the mixed sample would be:

$$\mu_2^{(M)} = (\frac{\delta}{2K}(1 + \sum_{i=1}^{k-2}\cos\gamma_i + \frac{t_2}{V}\cos\gamma_{k-1}), \frac{\delta}{2K}(\sum_{j=1}^{k-2}\sin\gamma_j + \frac{t_2}{V}\sin\gamma_{k-1}), 0, \ldots, 0)^{\mathbb{T}};$$
$$\mu_2^{(M)} = -\mu_2^{(M)},$$

here $K = k - 1 + \frac{t_2}{V}$.

The data sample coming from the mixture of distributions $S_1, S_2, \ldots, S_k$ will no

longer have $\mathbf{I}$ covariance matrix. The covariance of the mixture would be:

$$\begin{aligned}\Sigma^{(M)} &= \mathbf{I} - \frac{1}{K}(\sum_{s=1}^{k-1}\mu_1^{(s)}\mu_1^{(s)\mathbb{T}} - \frac{t_2}{V}\mu_1^{(k)}\mu_1^{(k)\mathbb{T}}) + \mu_2^{(M)}\mu_1^{(M)\mathbb{T}} \\ &= \mathbf{I} + \mathbf{R}.\end{aligned}$$

Here $\mathbf{R}$ is a $p \times p$ matrix the non-zero elements $r_{11}, r_{12}, r_{21}, r_{22}$, where:

$$\begin{aligned}r_{11} &= \frac{\delta^2}{4K}(1 + \sum_{i=1}^{k-2}\cos^2\gamma_i + \frac{t_2}{V}\cos^2\gamma_{k-1} - \frac{1}{K}(1 + \sum_{i=1}^{k-2}\cos\gamma_i + \frac{t_2}{V}\cos\gamma_{k-1})^2); \\ r_{22} &= \frac{\delta^2}{4K}(\sum_{j=1}^{k-2}\sin^2\gamma_j + \frac{t_2}{V}\sin^2\gamma_{k-1} - \frac{1}{K}(\sum_{j=1}^{k-2}\sin\gamma_j + \frac{t_2}{V}\sin\gamma_{k-1})^2).\end{aligned}$$

We would estimate the means from the sample, the estimates would be: $\mu_i^{(k)} \leftarrow \overline{\mathbf{X}}_\mathbf{i}^{(\mathbf{k})} \sim \mathcal{N}(\mu_i^{(k)}, \frac{2\mathbf{I}}{t_2})$, $\mu_i^{(M)} \leftarrow \overline{\mathbf{X}}_\mathbf{i}^{(\mathbf{M})} \sim \mathcal{N}(\mu_i^{(M)}, \frac{4\Sigma^{(M)}}{VK})$.

Using the same reasoning as in Section A.1 we come up with the expression for the generalization error of the mixed NMC. When $(VK) \to \infty, p \to \infty, \frac{VK}{p} = const$:

$$E_M^{(k)} = \Phi\left\{-\frac{m_1u_1 + m_2u_2}{\sqrt{(\Omega_1 + u_1^2 + \Omega_2 + u_2^2 + \frac{4(p-2)}{VK})}}\right\}, \tag{A.14}$$

here $\mu_1^{(k)} = (m_1, m_2, 0, \ldots, 0)^\mathbb{T}$, $2\mu_2^{(M)} = (u_1, u_2, 0, \ldots, 0)^\mathbb{T}$, $\Omega_i = \frac{4(r_{ii}+1)}{VK}$. Then

$$\begin{aligned}E_M^{(k)} &= \Phi\left\{-\frac{\delta^2(T_C\cos\gamma_{k-1} + T_S\sin\gamma_{k-1})}{2K\sqrt{\frac{4p}{VK} + \frac{\delta^2}{VK^2}(K - \frac{1}{K}(T_C^2 + T_S^2)) + \frac{\delta^2}{K^2}(T_C^2 + T_S^2)}}\right\} \\ &= \Phi\left\{-\frac{\delta(T_C\cos\gamma_{k-1} + T_S\sin\gamma_{k-1})}{2\sqrt{\frac{4pKv}{\delta^2} + vK + (K-v)(T_C^2 + T_S^2)}}\right\},\end{aligned}$$

here $T_C = 1 + \sum_{i=1}^{k-2}\cos\gamma_i + t_2v\cos\gamma_{k-1}, T_S = \sum_{j=1}^{k-2}\sin\gamma_j + t_2v\sin\gamma_{k-1}$.

A.3 Theoretical Accuracy of LDC after a Change

Following Raudys' derivation of the classification error [170], here we derive the expression for the error of the linear discriminant classifier C trained on source S_1 and applied to the probability distributions in source S_2 (notation $E_2(C_1)$). In source S_1, the two classes are distributed as $\mathbf{x} \sim \mathcal{N}(\mu_i^{(1)}, \Sigma)$, $\mathbf{x} \in \Re^n$, $i = 1, 2$. Superscript in parentheses will be used to indicate the source. We assume that the prior probabilities are $P(\omega_1) = P(\omega_2) = 1/2$. The discriminant function of LDC is

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \tag{A.15}$$

$$= \left(\mu_1^{(1)} - \mu_2^{(1)}\right)^T \Sigma^{-1}\mathbf{x} + \frac{1}{2}\left[\left(\mu_2^{(1)}\right)^T \Sigma^{-1}\mu_2^{(1)} - \left(\mu_1^{(1)}\right)^T \Sigma^{-1}\mu_1^{(1)}\right] \tag{A.16}$$

If $g(\mathbf{x}) \geq 0$ then $\mathbf{x}$ is assigned to class ω_1, otherwise, to class ω_2. The probability that C will make an error is

$$E(C) = \frac{1}{2}\big(Pr(g(\mathbf{x}) < 0|\omega_1) + Pr(g(\mathbf{x}) \geq 0|\omega_2)\big). \tag{A.17}$$

Conditioned by either ω_1 or ω_2, the discriminant function $g(\mathbf{x})$ becomes a linear combination of normally distributed variables, and is normally distributed itself. The error can be calculated using the cumulative distribution function Φ of the standard normal distribution. For this, we need the expectation and the standard deviation of $g(\mathbf{x})$, conditioned by the respective class label. For source S_2, class ω_1 has mean $\mu_1^{(2)} = \mu_1^{(1)} + \Delta_1$. Therefore the expectation of $g(\mathbf{x})$ for the "new" ω_1 is

$$\mathcal{E}_1^{(2)}[g(\mathbf{x})] = \mathbf{w}^T\mathcal{E}_1^{(2)}[\mathbf{x}] + w_0 = \mathbf{w}^T(\mu_1^{(1)} + \Delta_1) + w_0. \tag{A.18}$$

Expanding $\mathbf{w}$ and w_0 leads to

$$\mathcal{E}_1^{(2)}[g(\mathbf{x})] = \frac{1}{2}\left(\delta^{(1)}\right)^2 + \mathbf{w}^T\Delta_1, \tag{A.19}$$

where $\delta^{(1)} = \left(\mu_1^{(1)} - \mu_2^{(1)}\right)^T \Sigma^{-1}\left(\mu_1^{(1)} - \mu_2^{(1)}\right)$ is the Mahalanobis distance between the class means in source S_1. In the same way we arrive at the expectation

of $g(\mathbf{x})$ conditioned by the new class ω_2

$$\mathcal{E}_2^{(2)}[g(\mathbf{x})] = -\frac{1}{2}\left(\delta^{(1)}\right)^2 + \mathbf{w}^T\Delta_2. \tag{A.20}$$

The variance of $g(\mathbf{x})$ is the same for both classes, and is given by

$$\begin{aligned}\mathcal{V}[g(\mathbf{x})] &= \mathcal{V}[\mathbf{w}^T\mathbf{x} + w_0] = \mathbf{w}^T\Sigma\mathbf{w} &\text{(A.21)}\\ &= \left(\mu_1^{(1)} - \mu_2^{(1)}\right)^T \Sigma^{-1}\,\Sigma\,\Sigma^{-1}\left(\mu_1^{(1)} - \mu_2^{(1)}\right) = \left(\delta^{(1)}\right)^2. &\text{(A.22)}\end{aligned}$$

Then

$$Pr(g(\mathbf{x}) < 0|\omega_1) = \Phi\left(-\frac{\mathbf{w}^T\Delta_1}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right) \tag{A.23}$$

and

$$Pr(g(\mathbf{x}) \geq 0|\omega_2) = 1 - \Phi\left(-\frac{\mathbf{w}^T\Delta_2}{\delta^{(1)}} + \frac{\delta^{(1)}}{2}\right) = \Phi\left(\frac{\mathbf{w}^T\Delta_2}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right) \tag{A.24}$$

By substituting the conditional probabilities (A.23) and (A.24) back in (A.17), we arrive at the error on S_2 of LDC trained on S_1

$$E_2(C_1) = \frac{1}{2}\left\{\Phi\left(-\frac{\mathbf{w}^T\Delta_1}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right) + \Phi\left(\frac{\mathbf{w}^T\Delta_2}{\delta^{(1)}} - \frac{\delta^{(1)}}{2}\right)\right\}. \tag{A.25}$$

A.4 A Correction for the Estimate of the Common Covariance Matrix

In the derivation of (3.9) we assumed that the common covariance matrix for the classes can be calculated even with one observation. In practice, to have a non-singular covariance matrix for one class, we need at least $p + 1$ different observations from that class, where p is the dimensionality of the problem [90]. Numerous regularization and 'shrinkage' methods have been proposed for obtaining a non-singular estimate of the covariance matrix [70, 90, 188]. For the online classification considered here, we adopt a simpler approach. We use the nearest mean classifier (NMC), assuming that Σ is the identity matrix, until the data estimate of Σ becomes non-singular. From this point on, we continue with LDC. In order to incorporate the correction in the expression for N^*, we shall

assume that the point of reverting from NMC to LDC comes after we accumulate $p+1$ points from one of the classes. With i.i.d data, the number of samples $\mathbf{X}$ until $p+1$ samples from class ω_i apprear has a negative binomial distribution with mean $(p+1)/P(\omega_i)$. In the case of two equiprobable classes the expected number of samples until $p+1$ samples appear from either of the two classes is $2(p+1)$.

Equation (3.9) is applicable for NMC with $\delta^{(2)}$ calculated as the Euclidean distance between the two class means instead of the Mahalanobis distance. Therefore the switch strategy should be as follows

1. Calculate N^*_{NMC} using (3.9) with $\delta^{(2)} = \left(\mu_1^{(2)} - \mu_2^{(2)}\right)^T \left(\mu_1^{(2)} - \mu_2^{(2)}\right)$.
2. Calculate N^*_{LDC} using (3.9) with $\delta^{(2)} = \left(\mu_1^{(2)} - \mu_2^{(2)}\right)^T \Sigma^{-1} \left(\mu_1^{(2)} - \mu_2^{(2)}\right)$.
3. Assuming that $N^*_{\text{LDC}} \leq N^*_{\text{NMC}}$ (because LDC learns faster), consider the following three cases
   (a) $2(p+1) \leq N^*_{\text{LDC}} \leq N^*_{\text{NMC}}$. In this case use the old LDC until $N^* = N^*_{\text{LDC}}$, then switch to the new LDC.
   (b) $N^*_{\text{LDC}} \leq 2(p+1) \leq N^*_{\text{NMC}}$. In this case use the old LDC until $N^* = 2(p+1)$, then switch to the new LDC.
   (c) $N^*_{\text{LDC}} \leq N^*_{\text{NMC}} \leq 2(p+1)$. In this case use the old LDC until $N^*_1 = N^*_{\text{NMC}}$ and then switch to the new NMC. When the window size reaches $N^*_2 = 2(p+1)$, switch to the new LDC.

A.5 Correction for δ^2

In this appendix we derive an expression for the unbiased estimate of the Euclidean distance δ between two class means.

Consider two samples: a sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_{N_x}\} \subset \Re^n$ of i.i.d. data coming from a distribution with mean μ_x, and a sample $\{\mathbf{y}_1, \ldots, \mathbf{y}_{N_y}\}$ of i.i.d. data coming

from a distribution with mean μ_y. Let $\bar{\mathbf{x}}_{N_x}$ and $\bar{\mathbf{y}}_{N_y}$ be the two sample means. The expected squared Euclidean distance calculated from $\bar{\mathbf{x}}_{N_x}$ and $\bar{\mathbf{y}}_{N_y}$ is

$$\begin{aligned} E[\hat{\delta}^2] &= E[(\bar{\mathbf{x}}_{N_x} - \bar{\mathbf{y}}_{N_y})^T(\bar{\mathbf{x}}_{N_x} - \bar{\mathbf{y}}_{N_y})] && \text{(A.26)} \\ &= E[\bar{\mathbf{x}}_{N_x}^T\bar{\mathbf{x}}_{N_x}] - 2E[\bar{\mathbf{x}}_{N_x}^T\bar{\mathbf{y}}_{N_y}] + E[\bar{\mathbf{y}}_{N_y}^T\bar{\mathbf{y}}_{N_y}] && \text{(A.27)} \end{aligned}$$

Consider first the second term of (A.27).

$$E[\bar{\mathbf{x}}_{N_x}^T\bar{\mathbf{y}}_{N_y}] = E\left[\left(\frac{1}{N_x}\sum_{i=1}^{N_x}\mathbf{x}_i\right)^T\left(\frac{1}{N_y}\sum_{j=1}^{N_y}\mathbf{y}_j\right)\right] \quad \text{(A.28)}$$

$$= E\left[\left(\frac{1}{N_x}\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x) + \mu_x\right)^T\left(\frac{1}{N_y}\sum_{j=1}^{N_y}(\mathbf{y}_j - \mu_y) + \mu_y\right)\right] \quad \text{(A.29)}$$

$$\begin{aligned} = E\Bigg[&\frac{1}{N_xN_y}\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x)^T\sum_{j=1}^{N_y}(\mathbf{y}_j - \mu_y) + \frac{\mu_x^T}{N_x}\sum_{j=1}^{N_y}(\mathbf{y}_j - \mu_y) \\ &+ \frac{\mu_y^T}{N_y}\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x) + \mu_x^T\mu_y\Bigg]. \end{aligned} \quad \text{(A.30)}$$

Since $\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x) = 0$ and $\sum_{j=1}^{N_y}(\mathbf{y}_j - \mu_y) = 0$, the second and the third term are both zeros. The observations from the two samples are unrelated, hence $E[(\mathbf{x}_i - \mu_x)^T(\mathbf{y}_j - \mu_y)] = 0$. Therefore,

$$E[\bar{\mathbf{x}}_{N_x}^T\bar{\mathbf{y}}_{N_y}] = \mu_x^T\mu_y. \quad \text{(A.31)}$$

For the first term of (A.27),

$$E(\bar{\mathbf{x}}_{N_x}^T\bar{\mathbf{x}}_{N_x}) = E\left[\left(\frac{1}{N_x}\sum_{i=1}^{N_x}\mathbf{x}_i\right)^T\left(\frac{1}{N_x}\sum_{j=1}^{N_x}\mathbf{x}_j\right)\right] \quad \text{(A.32)}$$

$$= E\left[\left(\frac{1}{N_x}\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x) + \mu_x\right)^T \left(\frac{1}{N_x}\sum_{j=1}^{N_x}(\mathbf{x}_j - \mu_x) + \mu_x\right)\right] \tag{A.33}$$

$$= E\left[\frac{1}{N_x^2}\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x)^T \sum_{j=1}^{N_x}(\mathbf{x}_j - \mu_x) + 2\frac{\mu_x^T}{N_x}\sum_{j=1}^{N_x}(\mathbf{x}_j - \mu_x) + \mu_x^T\mu_x\right]. \tag{A.34}$$

Again, the second term becomes zero because $\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x) = 0$. The terms in the double summation are $E[(\mathbf{x}_i - \mu_x)^T(\mathbf{x}_j - \mu_x)] = 0$ for $i \neq j$. For $i = j$, $E[(\mathbf{x}_i - \mu_x)^T(\mathbf{x}_i - \mu_x)]$ is the sum of the variances of the n features, i.e., the trace of the covariance matrix of the first distribution. If the covariance matrix is the identity matrix, its trace is n, the dimensionality of the data. There are N_x such terms in the double summation, hence $\sum_{i=1}^{N_x}(\mathbf{x}_i - \mu_x)^T \sum_{j=1}^{N_x}(\mathbf{x}_j - \mu_x) = N_x \times n$. Therefore

$$E[\bar{\mathbf{x}}_{N_x}^T \bar{\mathbf{x}}_{N_x}] = \frac{n}{N_x} + \mu_x^T\mu_x. \tag{A.35}$$

Similarly,

$$E[\bar{\mathbf{y}}_{N_y}^T \bar{\mathbf{y}}_{N_y}] = \frac{n}{N_y} + \mu_y^T\mu_y. \tag{A.36}$$

Finally, (A.27) becomes

$$E[\hat{\delta}^2] = \frac{n}{N_x} + \mu_x^T\mu_x + \frac{n}{N_y} + \mu_y^T\mu_y - 2\mu_x^T\mu_y \tag{A.37}$$

$$= \underbrace{(\mu_x - \mu_y)^T(\mu_x - \mu_y)}_{\text{true } \delta^2} + \frac{n}{N_x} + \frac{n}{N_y} = \delta^2 + \frac{n}{N_x} + \frac{n}{N_y}. \tag{A.38}$$

To eliminate the bias, we use the corrected δ^2

$$\delta^2_{\text{corrected}} = \hat{\delta}^2_{\text{estimated}} - \frac{n}{N_x} - \frac{n}{N_y}. \tag{A.39}$$

A.6 Change Detection Using Hotelling T-test

Suppose that we have the labeled sequence of observations labeled in c classes. To estimate the likelihood of a change at time d, where $1 \leq d \leq t$, we assume that the class means migrate independently of one another. Then the probability

that there is a change at time d is

$$P(\text{change}|d) = 1 - \prod_{k=1}^{c} P(\text{no change in } \mu_k|d),$$

where μ_k is the mean for class k, $k = 1, \ldots, c$. Given that the data lives in $\Re^n$, the value of $P(\text{no change in } \mu_k|d)$ can be estimated using the p-value of the Hotelling multivariate T^2-test. This test compares the means for class k before and after the hypothetical change at d.

The mean before the change, $\mu_k^{(1)}$, is estimated from the streaming data from 1 to d labeled in class k; the mean after the change, $\mu_k^{(2)}$, is estimated from the streaming data from $d+1$ to t labeled in class k. Let $\hat{\Sigma}$ be the unbiased pooled covariance matrix estimated from the data and N_1 and N_2 be the respective sizes of the two samples. The T^2 statistic is

$$T^2 = \frac{(N_1 + N_2 - n - 1)}{n(N_1 + N_2 - 2)} \frac{N_1 N_2}{(N_1 + N_2)} \hat{\delta}^2 \tag{A.40}$$

where $\hat{\delta^2}$ is the squared Euclidean distance $\hat{\delta^2} = \left(\hat{\mu}_k^{(1)} - \hat{\mu}_k^{(2)}\right)^T \hat{\Sigma}^{-1} \left(\hat{\mu}_k^{(1)} - \hat{\mu}_k^{(2)}\right)$.

If the two means come from the same distribution, T^2 has F-distribution with degrees of freedom $(n, N_1 + N_2 - 1 - n)$.

If we use the notation $p_k(d)$ as the p-value returned by the Hotelling T^2-test comparing class k samples before and after time moment d, the probability of change at d can be estimated as

$$P(\text{change}|d) = 1 - \prod_{i=1}^{c} p_k(d). \tag{A.41}$$

To arrive at a probability distribution over $t_1, \ldots, t_i$, we can normalize by dividing the conditional probability (3.13) by

$$P(\text{change}, t_k) = \frac{P(\text{change}|t_k)}{\sum_{d=1}^{i} P(\text{change}|t_d)}. \tag{A.42}$$

In order to calculate the distribution, we need to check all possible split points between t_1 and t_n, as done in other studies performing retrospective change detection [1, 21, 105].

Appendix B

Pseudo Codes for the Peer Algorithms

In this Appendix we present pseudo codes and the settings used for the peer algorithms, which we implemented and used in experimental evaluation through the thesis. For consistency, the algorithms were named using the first three letters of the surname of the first author.

Gama et al [74] (GAM) algorithm is presented in Figure B.1.

Bifet et al [21] (BIF) algorithm is presented in Figure B.2.

Klinkenberg and Rentz [111] (KLI) algorithm is presented in Figure B.3.

Tsymbal et al [199] (TSY) algorithm is presented in Figure B.4.

In Figure B.5 we provide a pseudocode for the growing window (ALL), which is used as a benchmark.

DRIFT DETECTION (GAM)

input: labeled dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$; warning threshold T_w (default $T_w = 2$); detection threshold T_d (default $T_d = 3$); warm-up window size w_0 (default $w_0 = 30$).

1. Initialize the minimum classification error $p_{\min} = \infty$ and the corresponding standard deviation $s_{\min} = \infty$. Set the warning zone flag, f_w, to false.

2. *For* $j = 1$ *to* $t - 1$ (all the observations)

   If $j < w_0$,

   *t*hen $w_{j+1} = w_j + 1$, (warm up, only grow the window),

   *e*lse

   i. Train a classifier on the current window of size w_j
   ii. Classify observation $\mathbf{x}_{j+1}$.
   iii. Update the error rate over the current window. Let $\hat{p}$ be the updated error rate and $\hat{s} = \sqrt{\hat{p}(1-\hat{p})/w_j}$ be the updated standard deviation.
   iv. *If* $(\hat{p} + \hat{s}) < (p_{\min} + s_{\min})$, *then* update the minimum error by $p_{\min} = \hat{p}$ and $s_{\min} = \hat{s}$.
   v. *If* $(\hat{p} + \hat{s}) > (p_{\min} + T_w \times s_{\min})$ *and* f_w = false, *then* switch the warning zone flag f_w = true and set up the warning time $t_w = j$.
   vi. *If* $(\hat{p} + \hat{s}) < (p_{\min} + T_w \times s_{\min})$ *and* f_w = true, *then* cancel the warning zone f_w = false, $t_w = \infty$.
   vii. *If* $(\hat{p} + \hat{s}) > (p_{\min} + T_d \times s_{\min})$, *then* change has been detected. Take as the new training and detection window all the observations since t_w (size $w_{j+1} = j - t_w + 1$), set $p_{\min} = \infty$, $s_{\min} = \infty$, f_w = false, and $t_w = \infty$.
   *E*lse update the window by adding $\mathbf{x}_{j+1}$ to it (size $w_{j+1} = w_j + 1$).

3. Set $N^{GAM} = w_t$.

output: training window size N^{GAM}.

Figure B.1: Warning window algorithm for streaming data (GAM)

WINDOW SPLIT ALGORITHM FOR INSTANCE DATA (BIF)

input: a sequence of unlabeled univariate observations $\{x_1, \ldots, x_t\}$ and a confidence value $D \in (0, 1)$.

1. Set $i = 1$, $j = 2$, $w_t = t$.
2. Partition the data into $\mathbf{x}^{(0)} = (x_i, \ldots, x_{j-1})$ and $\mathbf{x}^{(1)} = (x_j, \ldots, x_t)$,
3. calculate the window cut threshold

$$\epsilon_{cut} = \sqrt{\frac{1}{2m} \log \frac{4}{D}}\,, \quad \text{where} \quad m = \left(\frac{1}{j-1} + \frac{1}{t-j+1}\right)^{-1},$$

4. calculate the means of the two partitions, $\bar{\mathbf{x}}^{(0)}$ and $\bar{\mathbf{x}}^{(1)}$,
5. *if* $|\bar{\mathbf{x}}^{(0)} - \bar{\mathbf{x}}^{(1)}| \geq \epsilon_{cut}$, *then* set $i = j$, $j = j + 1$, $w_t = t - j + 1$ and *repeat* from step 2,
   else if $j < t$, *then* set $j = j + 1$ and *repeat* from step 2.
6. Set $N^{BIF} = w_t$.

output: window size N^{BIF}

Figure B.2: Adaptive windowing algorithm for 1-d data (BIF).

WINDOW SELECTION ALGORITHM (KLI)

INPUT
Data: historical instances $\mathbf{X^H} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ with labels $\mathbf{y^H}$, target instance $\mathbf{X}_{t+1}$ without a label.
Parameters: batch size m. Base learner type: $\mathcal{L}$.
ALGORITHM

1. *For* $j = 1$ *to* t (all the batches),
   (a) *for* $k = 1$ *to* m (all the observations in the last batch),
      i. build a classifier on the data in $\{\mathbf{X}_j, \ldots, \mathbf{X}_t\}$, using cross-validation,
      ii. test the classifier on the excluded observation,
      if correctly classified $e_k = 0$, *else* $e_k = 1$,
   (b) calculate the error $E_j = \frac{1}{m} \sum_{k=1}^{m} e_k$ for past window of size $w_j = j \times m$.
2. Find minimum error $j^* = \arg\min_{j=1}^{t} E_j$.

OUTPUT
The index j^* to form a training set $\mathbf{X_t^T} = (\mathbf{X}_{j*}, \ldots, \mathbf{X}_t)$.

Figure B.3: Window Selection Algorithm (KLI).

DYNAMIC ENSEMBLE ALGORITHM (TSY)

INPUT
Data: historical instances $\mathbf{X^H} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ with labels $\mathbf{y^H}$, target instance $\mathbf{X}_{t+1}$ without a label.
Parameters: training set size N, batch size m, maximal ensemble size M, neighborhood size k.
Base learner type: $\mathcal{L}$.
ALGORITHM

1. Build classifier $\mathcal{L}_t$ using $\mathbf{X}_{t-N+1}, \ldots, \mathbf{X}_t$ labeled observations.
2. *If* ensemble size $< M$ *then* add $\mathcal{L}_t$ to the ensemble, *else* replace the ensemble member which showed the largest error on the latest batch.
3. Find k nearest neighbors to $\mathbf{X}_{t+1}$: $\{\mathbf{X}_{z1}, \ldots, \mathbf{X}_{zk}\}$ using Heterogeneous Euclidean overlap measure d^H, which is the Euclidean distance with normalized features.
4. *For* $j = 1$ *to* M, calculate weights (for each ensemble member $\mathcal{L}_1, \ldots, \mathcal{L}_M$)
   $w(\mathcal{L}_j) = \frac{\sum_{s=1}^{k} (\frac{1}{d^H(\mathbf{X}_{t+1}, \mathbf{X}_{zs})} r_j(\mathbf{X}_{zs}))}{\sum_{s=1}^{k} \frac{1}{d^H(\mathbf{X}_{t+1}, \mathbf{X}_{zs})}}$,
   where *if* $\mathcal{L}_j$ is correct in predicting the label of $\mathbf{X}_{zs}$
   then $r_j(\mathbf{X}_{zs}) = 1$, *else* $r_j(\mathbf{X}_{zs}) = -1$.
5. Select $\mathcal{L}^* = \mathcal{L}_j*$, where $j^* = \arg\max_{j=1}^{M} w(\mathcal{L}_j)$.

OUTPUT
Classifier $\mathcal{L}^*$ for decision making.

Figure B.4: Dynamic ensemble algorithm (TSY).

ALL HISTORY TRAINING SET ALGORITHM (ALL)

INPUT
Data: historical instances $\mathbf{X^H} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$ with labels $\mathbf{y^H}$, target instance $\mathbf{X}_{t+1}$ without a label. Base learner type: $\mathcal{L}$.
ALGORITHM
Train the classifier $\mathcal{L}_t$ using a training set $\mathbf{X_t^T} = (\mathbf{X}_1, \ldots, \mathbf{X}_t)$. **OUTPUT**
Trained classifier $\mathcal{L}_t$ to be applied to the testing instance $\mathbf{X}_{t+1}$.

Figure B.5: All history training set algorithm (ALL).

Appendix C

Datasets

In this Appendix we present details of all the datasets used in the experiments in the thesis. The characteristics are summarized in Table C.1.

Gaus1, Stag*, Hyper1 and SEA* are commonly used in concept drift research [152, 193]. We also used real data sets from the UCI repository [8] adding to them simulated drift.

Gaus1 - Gaussian data . 200 observations were generated as the streaming data, 100 before the concept change, and 100 after. The number of features was chosen to be $n = 4$. The means in source S_1 were $\mu_1^{(1)} = [-1, 0, 0, 0]^T$ and $\mu_2^{(1)} = [1, 0, 0, 0]^T$. The concept drift was set at $\Delta_1 = [0.5, 0.2, 0, 0, 0]^T$ and $\Delta_2 = [0.7, 0, 0, 0, 0]^T$. The prior probabilities in both sources were 0.5/0.5. The common covariance matrix, Σ, was constructed from an identity matrix of size 4 by setting $\sigma_{1,2} = \sigma_{2,1} = 0.3$.

Stag* - STAGGER data [211] Each data point is described by 3 features, each with three possible categories: size $\in$ {small, medium, large}, colour $\in$ {red, green, blue} and shape $\in$ {square, circular, triangular}. The numerical representation of a data point consists of 9 bits, 3 for each feature. For example, a large, red, square object is encoded as the vector $[0, 0, 1, 1, 0, 0, 1, 0, 0]^T$ and treated as a point in $\Re^9$. Three classification tasks were to be learned in a course of 120

Table C.1: Characteristics of the used datasets.

Name	Dimensionality	Size	Class balance	Type of data	Source of drift	Type of drift	Used in Section
Gaus1	4	200	0.5:0.5	artif.	sim.	sudden	3.4.2
Stagger	9	120	changing	artif.	sim.	2×sudden	3.4.2,3.6
Hyper1	2	250	0.5:0.5	artif.	sim.	4×sudden	3.4.2
SEA*	3	800	changing	artif.	sim.	3×sudden	3.4.2
Vote1	16	435	0.61:0.39	real	sim.	sudden	3.4.2
Iono1	34	351	0.61:0.39	real	sim.	sudden	3.4.2
Gaus2	7	400	0.5:0.5	artif.	sim.	sudden	3.6
Hyper2	2	250	0.5:0.5	artif.	sim.	4×sudden	3.6
WRaustralian	14	690	0.56:0.44	real	sim.	sudden	3.6
WRbreast	30	596	0.64:0.36	real	sim.	sudden	3.6
WRcylinder	36	540	0.58:0.42	real	sim.	sudden	3.6
WRgerman	24	1000	0.70:0.30	real	sim.	sudden	3.6
WRhepatitis	19	155	0.79:0.21	real	sim.	sudden	3.6
WRionosphere	34	351	0.64:0.36	real	sim.	sudden	3.6
WRStatlog heart	13	270	0.56:0.44	real	sim.	sudden	3.6
WRSPECT heart	22	267	0.79:0.21	real	sim.	sudden	3.6
WRsonar	60	208	0.53:0.47	real	sim.	sudden	3.6
WRvote	16	435	0.61:0.39	real	sim.	sudden	3.6
Luxembourgh	31	1901	0.51:0.49	real	real	gradual	4.6
Ozone	72	2534	0.94:0.06	real	real	gradual	4.6
Electricity	6	2956	0.57:0.43	real	real	gradual	4.6
German	23	1000	0.70:0.30	real	sim.	gradual	4.6
Vote2	16	435	0.61:0.39	real	sim.	gradual	4.6
Iono2	43	435	0.61:0.39	real	sim.	gradual	4.6
SLIGRO 538 products	20	119	regression	real	real	seasonality	5.5
A (boiler)	1	50977	regression	real	real	24×sudden	6.3
B (boiler)	1	25197	regression	real	real	9×sudden	6.3
C (boiler)	1	25197	regression	real	real	6×sudden	6.3

points. From point 1 to point 40, the classes to be distinguished are [size = small AND colour = red] vs all other values; from 41 to 80, [colour = green OR shape = circular] vs all other values; and from 81 to 120, [size = small OR size = large] vs all other values.

Hyper1- Moving-hyperplane data [63, 93, 112] The initial data and the 4 subsequent concept changes are shown in Figure3.9. The separation line started at 0° and was rotated to 45°, 90°, 135°, 180°. To form streaming data, 50 i.i.d points were drawn from each source before the next rotation.

SEA* data [193] Each data point is described by three features, $\mathbf{x} = [x_1, x_2, x_3]^T$, where $\mathbf{x}$ are uniformly randomly generated from $[0, 10]^3$. Only the first two features are relevant. An instance belongs to class 1 if $x_1 + x_2 \leq \Theta$ and to class 2 otherwise, where Θ is a threshold value, different for each concept. There are four concepts $\Theta = 8; 9; 7; 8.5$. We generate 200 instances for each concept. No label noise was added so the two classes are perfectly separable by a hyperplane in the feature space $[0, 10]^3$, parallel to the x_3 axis.

Vote1 - Vote data [8] This data set represents the 1984 United States Congressional Voting Records (435 data points of which 267 democrats and 168 republicans with 16 binary features).

Iono1 - Ionosphere data [8] This data set represents a two-class problem where a radar returns 'good' and 'bad' signals. The data consists of 351 instances ($136 + 215$), each having 34 features.

To form different i.i.d data streams with concept drift, the real data was first randomly permuted. Then feature pairs were swapped at every 50th instance (two features for the Vote data and four features for the Ionosphere data).

Gaus2 - Gaussian data. We generated two Gaussian classes in $\Re^7$ with identity covariance matrices and $\mu_1^{(1)} = (1, 0, \ldots, 0)^T$, $\mu_2^{(1)} = (-1, 0, \ldots, 0)^T$. A sudden change was simulated by shifting the means by $\Delta_1 = (1, 0.2, 0, \ldots, 0)^T$ and $\Delta_2 = (0.8, -0.4, 0 \ldots, 0)^T$, respectively. The change was introduced after classifying streaming observation 200 and before receiving observation 201.

STAGGER data. (citeSchlimmer86). Each data point is described by 3 features, each with three possible categories: size $\in$ {small, medium, large}, color $\in$

{red, green, blue} and shape $\in$ {square, circular, triangular}. The numerical representation of a data point consists of 9 bits, 3 for each feature. For example, a large, red, square object is encoded as the vector $[0, 0, 1, 1, 0, 0, 1, 0, 0]^T$ and treated as a point in $\Re^9$. Three classification tasks were to be learned in a course of 120 points. From point 1 to point 40, the classes to be distinguished are [size = small AND color = red] vs all other values; from 41 to 80, [color = green OR shape = circular] vs all other values; and from 81 to 120, [size = small OR size = large] vs all other values.

Hyper2 - Moving-hyperplane data. The data sequence is uniformly sampled from the unit square. The class labels are assigned according to a line through the center of the square. The line rotates giving rise to change in the class descriptions (hence 'moving hyperplane'). Starting with a vertical discrimination line, we simulate 4 changes by positioning the discriminating line at 30°, 60°, 90° and 120°. To form a data stream, 50 i.i.d points were drawn from each source before the next rotation.

Luxembourg data (Luxe). We constructed the dataset using European Social Survey [1] [98] 2002-2007. The task is to classify a subject with respect to the internet usage 'high' or 'low'. We use 20 features (31 after transformation of the categorical variables), which were selected as general demographic representation: Country, TV watching, total time on average weekday, Newspaper reading, total time on average weekday, Personal use of internet/e-mail/www, How interested in politics, Trust in the European Parliament, Trust in the United Nations, Signed petition last 12 months, Member of political party, How happy are you, How often socially meet with friends, relatives or colleagues, Subjective general health, How often pray apart from at religious services, Discrimination of respondent's group: gender, How long ago first came to live in country, Number of people living regularly as member of household, Year of birth, Second person in household:Relationship to respondent, Highest level of education, Main activity, last 7 days. All respondents. Post coded, Responsible for supervising other employees, date and time of the interview. The set is balanced

[1] Norwegian Social Science Data Services (NSD) acts as the data archive and distributor of the ESS data.

$977 + 924$.

Ozone level detection [8] (Ozone) represents local ozone peak prediction, that is based on eight hours measurement. Data size 72×2534. The set is highly imbalanced $160 + 2374$, with only 160 ozone peaks.

Electricity market data (Elec), first described by Harries [84]. We use the time period with no missing values comprised of 6×2956 instances collected along a period of 3 months from May 11 to July 11, 1997. Labels 'up' or 'down' indicate the change of the price. The set is moderately balanced $1673 + 1283$.

German credit approval (German2) [8] classifies customers as having good or bad credit risks. Following [115], a gradual concept change was introduced artificially as a hidden context. We sort the data using one of the features (feature 'age' was chosen) and then eliminate this feature from the dataset. Data size 23×1000. The set is imbalanced $700 + 300$.

Vote data [8] (Vote2) represents 1984 United States Congressional Voting Records. The instances represent 435 congressmen (267 democrats, 168 republicans). There are 16 features (votings). The data is categorical, we coded 'no' as -1, 'yes' as 1, missing value as 0.

Ionosphere data [8] (Iono2) represents Johns Hopkins University Ionosphere data. Binary classification task of the radar returns into 'good' and 'bad' signals. Data size 43×351. The set is moderately balanced $136 + 215$.

SLIGRO. Our experimental field consists of 538 product sales quantities over two years period (from July 2006 to October 2008). The sales are aggregated on weekly basis, thus each series is of 119 weeks length. Each series represent the sales of one product aggregated over all outlets.

BOILER. We use three mass signal data sets referred as A,B and C. The data was obtained from a pilot boiler operated in VTT Technical Research Center of Finland. A summary of the data sets is provided in Table 6.1 and they are plotted in Figure 6.6. Different composition of fuel was used in the three data sets. The total length of A is different from B and C.

Appendix D

Complexities

In this Appendix we present the estimates and discussion of computation complexities of the algorithms WR* and FISH presented in Chapters 3, 4 along with the peer algorithms used for experimental comparison KLI, TSY, GAM, BIF, ALL (see Appendix B for details).

First we discuss background and methodology we use for the evaluation and then present the complexity.

D.1 Background and Methodology

Recall incremental learning scenario presented in Chapter 2 Section 2.1. A sequence of instances $(\mathbf{X}_1, \ldots, \mathbf{X}_t)$ with known labels $(\mathbf{y}_1, \ldots, \mathbf{y}_t)$ is observed (*historical data*) and instance $\mathbf{X}_{t+1}$ (*target*) for which we need to output the label $\mathbf{y}_{t+1}$. For that at each time step we use a base learner $\mathcal{L}_t$ trained on all or a subset of the historical data.

In this evaluation we assume all the algorithms use the same embedded base learner, for instance the Nearest Mean Classifier NMC). To estimate the complexity we count the number of times a historical data instance needs to be passed (accessed) to make a classification decision for the target instance $\mathbf{X}_{t+1}$. The complexities are expressed as a function of t, which is the number of historical

instances present. We do not consider dimensionality of the data in complexity evaluation, because the dimensionality is fixed for all the data and all the peer algorithms.

We calculate three complexity estimates for each algorithm:

1. The worst case complexity for decision at time $t+1$.
2. Expected complexity for decision at time $t+1$. It is different from the worst case only for the algorithms operating in batch mode. It means that the model is updated not at every time step.
3. Average complexity over decisions from time 2 to time $t+1$.

In addition to complexities we also list the hyperparameters for each algorithm used as inputs, which need to be prefixed in advance.

D.2 Complexities of the Algorithms

Table D.1: Algorithm complexities.

Method	Worst case	Expected	Average over time	Hyper-parameters
ALL	t	t	$\frac{t}{2}$	$-$
KLI	$\frac{t^2}{2}$	$\frac{t^2}{2b}$	$\frac{t^2}{6b}$	b
TSY	tk	tk	$\frac{tk}{2}$	b,M,k,N
FISH1	tN	tN	$\frac{tN}{2}$	A,N
FISH2	$\frac{t^2k}{2}$	$\frac{t^2k}{2}$	$\frac{t^2k}{6}$	A, k
FISH3	$g\frac{t^2k}{2}$	$g\frac{t^2k}{2}$	$g(\frac{t^2k}{6}$	k
GAM	t	t	mean(N_j^{GAM}), $N^{GAM} \leq t$	T_d,T_w,w_0
BIF	t^2	t^2	$(\text{mean}(N_j^{BIF}))^2$	D
WR*	t^2	t^2	$(\text{mean}(N_j^{WR*}))^2$	$-$

The results (approximations) are presented in Table D.1. Granularity g is a step of the time and space proportion [1]. N_t^{WR*}, N_t^{BIF}, N_t^{GAM} are the window

[1] the number of options tried out, we use 10: option 1 $\alpha_1 = 0, \alpha_2 = 1$, option 2 $\alpha_1 = 0.1, \alpha_2 = 0.9, \ldots$, option 10 $\alpha_1 = 1, \alpha_2 = 0$

sizes output by the respective algorithms WR*, BIF and GAM at time j, where $j = 2 \ldots, t+1$. Parameters: b - batch size; M - ensemble size; k - testing neighborhood size; N - training set size; A - time/space weight; g - granularity of the time and space proportion; t - time since the start of the sequence; D - confidence value; T_d and T_w - thresholds; w_0 warm up window.

Bibliography

[1] R. Adams and D. MacKay. Bayesian online changepoint detection. Technical report, University of Cambridge Technical Report, 2007.

[2] C. Aggarwal. Towards systematic design of distance functions for data mining applications. In *KDD '03: Proc. of the 9th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 9–18. ACM, 2003.

[3] D. Aha. Lazy learning. In *Lazy learning*, pages 7–10. Kluwer Academic Publishers, 1997.

[4] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[5] R. Alcock and Y. Manolopoulos. Time-series similarity queries employing a feature-based approach. In *Proc. of the 7th Hellenic conf. on Informatics*, 1999.

[6] C. Anagnostopoulos, N. Adams, and D. Hand. Deciding what to observe next: adaptive variable selection for regression in multivariate data streams. In *SAC '08: Proc. of the 2008 ACM symposium on Applied computing*, pages 961–965. ACM, 2008.

[7] D. Anguita. Smart adaptive systems: State of the art and future directions of research. In *Proc. of the 1st European symp. on Intelligent Technologies, Hybrid Systems and Smart Adaptive Systems, EUNITE 2001*, 2001.

[8] A. Asuncion and D. Newman. Uci machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[9] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[10] S. Bach and M. Maloof. Paired learners for concept drift. In *Proc. of the 8th IEEE int. conf. on Data Mining*, pages 23–32. IEEE Press, 2008.

[11] M. Baena-Garcia, J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavalda, and

R. Morales-Bueno. Early drift detection method. In *Proc. of the 4th ECML PKDD int. workshop on Knowledge Discovery From Data Streams (IWKDDS '06)*, pages 77–86, 2006.

[12] J. Bakker, M. Pechenizkiy, I. Zliobaite, A. Ivannikov, and T. Karkkainen. Handling outliers and concept drift in online mass flow prediction in cfb boilers. In *Proc. of the 3rd int. workshop on Knowledge Discovery from Sensor Data (SensorKDD-09)*, pages 13–22. ACM, 2009. ISBN 978-1-60558-668-7. best paper award.

[13] J. Bakker, M. Pechenizkiy, I. Zliobaite, A. Ivannikov, and T. Karkkainen. Online mass flow prediction in cfb boilers with explicit detection of sudden concept drift. *under review*, 2009.

[14] M. Basseville and I. Nikiforov. *Detection of abrupt changes: theory and application*. Prentice-Hall, Inc., 1993.

[15] H. Becker and M. Arias. Real-time ranking with concept drift using expert advice. In *KDD '07: Proc. of the 13th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 86–94. ACM, 2007.

[16] R. Bell, Y. Koren, and C. Volinsky. The bellkor 2008 solution to the netflix prize. online, 2008. URL `http://www.research.att.com/~volinsky/netflix/`.

[17] S. Ben-David and R. Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, 73(3):273–287, 2008.

[18] J. Beringer and E. Hullermeier. Efficient instance-based learning on data streams. *Intelligent Data Analysis*, 11(6):627–650, 2007.

[19] S. Bickel, M. Bruckner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *ICML '07: Proc. of the 24th int. conf. on Machine learning*, pages 81–88. ACM, 2007.

[20] A. Bifet. *Adaptive Learning and Mining for Data Streams and Frequent Patterns*. PhD thesis, Universitat Politecnica de Catalunya, 2009.

[21] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proc. of SIAM int. conf. on Data Mining (SDM'07)*, pages 443–448. SIAM, 2007.

[22] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda. New ensemble methods for evolving data streams. In *KDD '09: Proc. of the 15th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages

139–148. ACM, 2009.

[23] D. Billsus and M. Pazzani. A hybrid user model for news story classification. In *UM '99: Proc. of the 7th int. conf. on User modeling*, pages 99–108. Springer-Verlag, 1999.

[24] M. Black and R. Hickey. Classification of customer call data in the presence of concept drift and noise. In *Soft-Ware 2002: Proc. of the 1st int. conf. on Computing in an Imperfect World*, pages 74–87. Springer-Verlag, 2002.

[25] M. Black and R. Hickey. Detecting and adapting to concept drift in bioinformatics. In *Proc. of Knowledge Exploration in Life Science Informatics, International Symposium, KELSI 2004*, volume 3303 of *LNCS*, pages 161–168. Springer, 2004.

[26] M. Black and R. J. Hickey. Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis*, 3(6):453–474, 1999.

[27] D. Blei and J. Lafferty. Dynamic topic models. In *ICML '06: Proc. of the 23rd int. conf. on Machine learning*, pages 113–120. ACM, 2006.

[28] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 20*, pages 129–136. MIT Press, 2008.

[29] R. Bolton and D. Hand. Statistical fraud detection: A review. *Statistical Science*, 17(3):235–255, 2002.

[30] M. Bottcher, M. Spott, and R. Kruse. Predicting future decision trees from evolving data. In *ICDM '08: Proc. of the 8th IEEE int. conf. on Data Mining*, pages 33–42. IEEE Computer Society, 2008.

[31] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, 1990.

[32] P. De Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *HYPERTEXT '03: Proc. of the 14th ACM conf. on Hypertext and hypermedia*, pages 81–84. ACM, 2003.

[33] S. Brown and M. Steyvers. Detecting and predicting changes. *Cognitive Psychology*, 58(1):49–67, 2009.

[34] Card Watch. Card fraud overview. online, accessed July 15, 2009. URL `http://www.cardwatch.org.uk`.

[35] G. Carpenter and S. Grossberg. Adaptive resonance theory (art). In *The handbook of brain theory and neural networks*, pages 79–82. MIT Press, 1998.

[36] G. Castillo. *Adaptive Learning Algorithms for Bayesian Network Classifiers*. PhD thesis, University of Aveiro, 2006.

[37] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection - a survey. *ACM Computing Surveys*, 41(3):article no. 15, 2009.

[38] D. Charles, A. Kerr, M. McNeill, M. McAlister, M. Black, J. Kücklich, A. Moore, and K. Stringer. Player-centred game design: Player modelling and adaptive digital games. In *Digital Games Research Conference 2005, Selected Papers Publication*, pages 285–298, 2005.

[39] C. Chatfield and D. L. Prothero. Box-jenkins seasonal forecasting: Problems in a case-study. *Journal of the Royal Statistical Society. Series A (General)*, 136(3):295–336, 1973.

[40] S. Chen, H. Wang, S. Zhou, and P. Yu. Stop chasing trends: Discovering high order models in evolving data. In *Proc. of the 2008 IEEE the 24th int. conf. on Data Engineering ICDE '08*, pages 923–932. IEEE Computer Society, 2008.

[41] H. Cho, M. Fadali, and K. Lee. Online probability density estimation of nonstationary random signal using dynamic bayesian networks. *Int. Journal of Control, Automation and Systems*, 6(1):109–118, 2008.

[42] F. Chu and C. Zaniolo. Fast and light boosting for adaptive mining of data streams. In *Proc. of the 5th Pacific-Asic conf. on Knowledge Discovery and Data Mining (PAKDD)*, pages 282–292. Springer Verlag, 2004.

[43] J. Cooley and J. Tukey. An algorithm for the machine computation of the complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[44] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9:1757–1774, 2008.

[45] F. Crespo and R. Weber. A methodology for dynamic data mining based on fuzzy clustering. *Fuzzy Sets and Systems*, 150:267–284, 2005.

[46] A. da Silva, Y. Lechevallier, F. Rossi, and F. de Carvalho. Construction and analysis of evolving data summaries: An application on web usage data. In *Proc. of the 7th int. conf. on Intelligent Systems Design and Applications ISDA '07*, pages 377–380. IEEE Computer Society, 2007.

[47] DARPA. Urban challenge. online, accessed July 15, 2009. URL `http://www.darpa.mil/grandchallenge/index.asp`.

[48] R. de Mantaras and E. Enric. Case-based reasoning: an overview. *AI Communications*, 10(1):21–29, 1997.

[49] S. Delany and P. Cunningham. An analysis of case-based editing in a spam filtering system. In *Proc. of the 7th European conf. on Case-Based Reasoning (ECCBR 2004)*, volume 3155 of *LNAI*, pages 128–141. Springer, 2004.

[50] S. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18(4–5):187–195, 2005.

[51] S. Delany, P. Cunningham, and A. Tsymbal. A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering. In *Proc. of the 19th int. conf. on Artificial Intelligence (FLAIRS 2006)*, pages 340–345. AAAI Press, 2006.

[52] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[53] T. Dietterich, G. Widmer, and M. Kubat. Special issue on context sensitivity and concept drift. *Machine Learning*, 32(2), 1998.

[54] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM '05: Proc. of the 14th ACM int. conf. on Information and knowledge management*, pages 485–492. ACM, 2005.

[55] G. Dong, J. Han, L. Lakshmanan, J. Pei, H. Wang, and P. Yu. Online mining of changes from data streams: Research problems and preliminary results. In *Proc. of the SIGMOD2003 workshop on Management and Processing of Data Streams*, 2003.

[56] S. Donoho. Early detection of insider trading in option markets. In *KDD '04: Proc. of the 10th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 420–429. ACM, 2004.

[57] A. Dries and U. Ruckert. Adaptive concept drift detection. *Statistical Analysis and Data Mining*, 5-6(Special issue on the Best of SDM'09):311–327, 2009.

[58] R. Duda, P. Hart, and D. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[59] J. Ekanayake, J. Tappolet, H. Gall, and A. Bernstein. Tracking concept drift of software projects using defect prediction quality. In *Proc. of the 6th IEEE working conf. on Mining Software Repositories*, pages 51–60. IEEE Computer Society, 2009.

[60] W. Fan. Systematic data selection to mine concept-drifting data streams. In *KDD '04: Proc. of the 10th ACM SIGKDD int. conf. on Knowledge discovery*

and data mining, pages 128–137. ACM, 2004.

[61] T. Fawcett. "in vivo" spam filtering: a challenge problem for kdd. *ACM SIGKDD Explorations Newsletter*, 5(2):140–148, 2003.

[62] F. Fdez-Riverola, E. Iglesias, F. Diaz, J. Mendez, and J. Corchado. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications: An International Journal*, 33(1):36–48, 2007.

[63] F. Ferrer-Troyano, J. Aguilar-Ruiz, and J. Riquelme. Incremental rule learning based on example nearness from numerical data streams. In *SAC '05: Proc. of the 2005 ACM symposium on Applied computing*, pages 568–572. ACM, 2005.

[64] D. Fisher and J. Schlimmer. Models of incremental concept learning: A coupled research proposal. Technical report CS-88-05, Vanderbilt University, 1988. URL `http://www.vuse.vanderbilt.edu/~dfisher/tech-reports/tr-88-05/proposal.html`.

[65] O. Flasch, A. Kaspari, K. Morik, and M. Wurst. Aspect-based tagging for collaborative media organization. In *Revised Selected and Invited Papers From Web to Social Web: Discovering and Deploying User and Content Profiles: Workshop on Web Mining, WebMine 2006.*, volume 4737 of *LNAI*, pages 122–141. Springer-Verlag, 2007.

[66] G. Forman. Incremental machine learning to reduce biochemistry lab costs in the search for drug discovery. In *Proc. of the 2nd workshop on Data Mining in Bioinformatics*, pages 33–36, 2002.

[67] G. Forman. Tackling concept drift by temporal inductive transfer. In *Proc. of the 29th annual int. ACM SIGIR conf. on Research and development in information retrieval (SIGIR '06)*, pages 252–259. ACM, 2006.

[68] A. Freitas and J. Timmis. Revisiting the foundations of artificial immune systems for data mining. *IEEE Transactions on Evolutionary Computation*, 11(4):521–540, 2007.

[69] Y. Freund and R. Schapire. A short introduction to boosting. In *Proc. of the 16th int. joint conf. on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999.

[70] J. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[71] K. Fukunaga and R. R. Hayes. Estimation of classifier performance. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 11(10):1087–1101, 1989.

[72] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *ACM SIGMOD Record*, 34(2):18–26, 2005.

[73] P. Gago, A. Silva, and M. Santos. Adaptive decision support for intensive care. In *Proc. of the 13th Portuguese conf. on Artificial Intelligence*, pages 415–425, 2007.

[74] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances In Artificial Intelligence, Proc. of the 17th Brazilian symposium on Artificial Intelligence (SBIA 2004)*, volume 3171 of *LNAI*, pages 286–295. Springer, 2004.

[75] J. Gama, R. Sebastiao, and P. Rodrigues. Issues in evaluation of stream learning algorithms. In *KDD '09: Proc. of the 15th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 329–338. ACM, 2009.

[76] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining data streams under block evolution. *ACM SIGKDD Explorations Newsletter*, 3(2):1–10, 2002.

[77] J. Gao, B. Ding, W. Fan, J. Han, and P. Yu. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12 (6):37–49, 2008.

[78] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. User profiles for personalized information access. In *The Adaptive Web*, pages 54–89. Springer Berlin / Heidelberg, 2007.

[79] R. Giacomini and B. Rossi. Detecting and predicting forecast breakdowns. Working Paper 638, European Central Bank, 2006.

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

[81] S. Grossberg. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23(3):121–134, 1976.

[82] J. Han and J. Gao. Research challenges for data mining in science and engineering. In *Next Generation of Data Mining*. Chapman & Hall, 2009.

[83] D. Hand. Classifier technology and the illusion of progress. *Statistical Science*, 21(1):1–14, 2006.

[84] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales, 1999.

[85] M. Harries and K. Horn. Detecting concept drift in financial time series prediction using symbolic machine learning. In *In Proc. of the 8th Australian*

joint conf. on artificial intelligence, pages 91–98, 1995.

[86] M. Harries, C. Sammut, and K. Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.

[87] M. Hasan and E. Nantajeewarawat. Towards intelligent and adaptive digital library services. In *Proc. of the 11th int. conf. on Asian Digital Libraries (ICADL 08)*, pages 104–113. Springer-Verlag, 2008.

[88] S. Hashemi, Y. Yang, M. Pourkashani, and M. Kangavari. To better handle concept change and noise: A cellular automata approach to data stream classification. In *Proc. of Australian conf. on Artificial Intelligence*, volume 4830 of *LNCS*, pages 669–674. Springer, 2007.

[89] C. Hilas. Designing an expert system for fraud detection in private telecommunications networks. *Expert Systems with Applications: An International Journal*, 36(9):11559–11569, 2009.

[90] J. Hoffbeck and D. Landgrebe. Covariance matrix estimation and classification with limited training data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):763–767, 1996.

[91] R. Horta, B. de Lima, and C. Borges. Data pre-processing of bankruptcy prediction models using data mining techniques. Online, 2009. URL `http://blog.campe.com.br/wp-content/uploads/2009/03/witpress_conf-2.pdf`.

[92] L. Huilin, Z. Guangboa, B. Rushana, C. Yongjina, and D. Gidaspowb. A coal combustion model for circulating fluidized bed boilers. *Fuel*, 79: 165–172, 2000.

[93] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *KDD '01: Proc. of the 7th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.

[94] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *Int. Journal of Forecasting*, 22(4):679–688, 2006.

[95] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[96] C. Jermaine. Data mining for multiple antibiotic resistance. online, 2008. URL `http://www.cise.ufl.edu/~cjermain/DM`.

[97] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Proc. of the 45th annual meeting of the Association of Computational Linguistics*, pages 264–271. Association for Computational Linguistics, 2007.

[98] R. Jowell and the Central Co-ordinating Team. European social survey 2002/2003; 2004/2005; 2006/2007. Technical Reports, London: Centre for Comparative Social Surveys, City University, 2003, 2005, 2007.

[99] F. Kamiran and T. Calders. Classification without discrimination. In *IEEE International Conference on Computer, Control & Communication (IEEE-IC4)*. IEEE press, 2009.

[100] M. Karnick, M. Ahiskali, M. Muhlbaier, and R. Polikar. Learning concept drift in nonstationary environments using an ensemble of classifiers based approach. In *Proc. of IEEE int. joint conf. on Neural Networks (IJCNN 2008)*, pages 3455–3462, 2008.

[101] I. Katakis, G. Tsoumakas, and I. Vlahavas. Dynamic feature space and incremental feature selection for the classification of textual data streams. In *Proc. of ECML/PKDD-2006 int. workshop on Knowledge Discovery from Data Streams*, pages 102–116, 2006.

[102] I. Katakis, G. Tsoumakas, and I. P. Vlahavas. An ensemble of classifiers for coping with recurring contexts in data streams. In *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 763–764. IOS Press, 2008.

[103] I. Katakis, G. Tsoumakas, and I. Vlahavas. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, 2009.

[104] M. Kelly, D. Hand, and N. Adams. The impact of changing populations on classifier performance. In *KDD '99: Proc. of the 5th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 367–371. ACM, 1999.

[105] D. Kifer, Sh. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proc. of the 30th int. conf. on Very Large Data Bases (VLDB 2004)*, pages 180–191, 2004.

[106] J. Kim, P. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross. Immune system approaches to intrusion detection — a review. *Natural Computing: an international journal*, 6(4):413–466, 2007.

[107] J. Kleinberg. Bursty and hierarchical structure in streams. In *KDD '02: Proc. of the 8th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 91–101. ACM, 2002.

[108] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[109] R. Klinkenberg. Meta-learning, model selection and example selection in machine learning domains with concept drift. In *Proc. of Annual Workshop of the Special Interest Group on Machine Learning, Knowledge Discovery, and Data Mining (FGML-2005) of the German Computer Science Society (GI) Learning - Knowledge Discovery - Adaptivity (LWA-2005)*, pages 64–171, 2005.

[110] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. of the 17th int. conf. on Machine Learning (ICML '00)*, pages 487–494. Morgan Kaufmann Publishers Inc., 2000.

[111] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning drifting concepts. In *Proc. of AAAI-98/ICML-98 workshop Learning for Text Categorization*, pages 33–40, 1998.

[112] J. Kolter and M. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8: 2755–2790, 2007.

[113] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proc. of the 15th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[114] A. Kostenko and R. Hyndman. A note on the categorization of demand patterns. *Journal of the Operational Research Society*, 57:1256–1257, 2006.

[115] I. Koychev. Gradual forgetting for adaptation to concept drift. In *Proc. of ECAI 2000 workshop Current Issues in Spatio-Temporal Reasoning*, pages 101–106, 2000.

[116] I. Koychev and R. Lothian. Tracking drifting concepts by time window optimisation. In *Research and Development in Intelligent Systems XXII, Proc. of AI-2005, the 25th SGAI int. conf. on Innovative Techniques and Applications of Artificial Intelligence*, pages 46–59, 2005.

[117] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *ICML '07: Proc. of the 24th int. conf. on Machine learning*, pages 449–456. ACM, 2007.

[118] H. Kriegel, K. Borgwardt, P. Kroger, A. Pryakhin, M. Schubert, and A. Zimek. Future trends in data mining. *Data Mining and Knowledge Discovery*, 15(1):87–97, 2007.

[119] K. Ku-Mahamud, N. Zakaria, N. Katuk, and M. Shbier. Flood pattern detection using sliding window technique. In *Proc. of the 3rd Asia int. conf. on Modelling & Simulation*, pages 45–50, 2009.

[120] M. Kubat, J. Gama, and P. Utgoff. Special issue on incremental learning systems capable of. dealing with concept drift. *Intelligent Data Analysis*, 8 (3), 2004.

[121] M. Kukar. Drifting concepts as hidden factors in clinical studies. In *Proc. of AIME 2003, the 9th conf. on Artificial Intelligence in Medicine in Europe*, pages 355–364, 2003.

[122] P. Kumar and V. Ravi. Bankruptcy prediction in banks and firms via statistical and intelligent techniques - a review. *European Journal of Operational Research*, 180(1):1–28, 2007.

[123] L. Kuncheva. Classifier ensembles for changing environments. In *Proc. of the 5th int. workshop on Multiple Classifier Systems (MCS 2004)*, volume 3077 of *LNCS*, pages 1–15. Springer, 2004.

[124] L. Kuncheva and I. Zliobaite. Linear discriminant classifier (ldc) for streaming data with concept drift. In *Structural, Syntactic, and Statistical Pattern Recognition, Proc. of joint IAPR int. workshop, SSPR & SPR 2008*, volume 5342 of *LNCS*, page 4. Springer-Verlag, 2008.

[125] L. Kuncheva and I. Zliobaite. On the window size for classification in changing environments. *Intelligent Data Analysis*, 13(6):861–872, 2009.

[126] A. Kusiak and A. Burns. Mining temporal data: A coal-fired boiler case study. In *Proc. of Knowledge-Based Intelligent Information and Engineering Systems, the 9th int. conf. (KES 2005), Part III*, LNAI, pages 953–958, 2005.

[127] T. Lane and C. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2 (3):295–331, 1999.

[128] M. Last. Online classification of nonstationary data streams. *Intelligent Data Analysis*, 6(2):129–147, 2002.

[129] N. Lathia, S. Hailes, and L. Capra. knn cf: a temporal social network. In *RecSys '08: Proc. of the 2008 ACM conf. on Recommender systems*, pages 227–234. ACM, 2008.

[130] A. Lattner, A. Miene, U. Visser, and O. Herzog. Sequential pattern mining for situation and behavior prediction in simulated robotic soccer. In *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *LNCS*, 2006.

[131] Y. Law and C. Zaniolo. An adaptive nearest neighbor classification algorithm for data streams. In *Proc. of PKDD*, volume 3721 of *LNCS*, pages 108–120. Springer, 2005.

[132] S. Laxman and P. Sastry. A survey of temporal data mining. *SADHANA, Academy Proceedings in Engineering Sciences*, 31(2):173–198, 2006.

[133] M. Lazarescu and S. Venkatesh. Using selective memory to track concept effectively. In *Proc. of the int. conf. on Intelligent Systems and Control*, pages 14–20, 2003.

[134] M. Lazarescu, S. Venkatesh, and H. Bui. Using multiple windows to track concept drift. *Intelligent Data Analysis*, 8(1):29–59, 2004.

[135] G. Lebanon and Y. Zhao. Local likelihood modeling of temporal text streams. In *ICML '08: Proc. of the 25th int. conf. on Machine learning*, pages 552–559. ACM, 2008.

[136] L. Liao, D. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007.

[137] D. Lin. An information-theoretic definition of similarity. In *Proc. of the 15th int. conf. on Machine Learning (ICML '98)*, pages 296–304. Morgan Kaufmann Publishers, 1998.

[138] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2): 107–144, 2007.

[139] D. Lu, P. Mausel, E. Brondizio, and E. Moran. Change detection techniques. *Int. Journal of Remote Sensing*, 25(12):2365–2401, 2004.

[140] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments. In *Proc. of the IEEE/RSJ int. conf. on Intelligent Robots and Systems (IROS07)*, pages 721–728, 2007.

[141] S. Mandl, B. Ludwig, S. Schmidt, and H. Stoyan. Recurring hidden contexts in online concept learning. In *Proc. of the workshop on Planning, Learning and Monitoring with Uncertainty in Dynamic Worlds*, pages 25–29, 2006.

[142] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems (NIPS 2008)*, pages 1041–1048. MIT Press, 2008.

[143] M. Markou and S. Singh. Novelty detection: a review—part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, 2003.

[144] M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. A multi-partition multi-chunk ensemble technique to classify concept-drifting data streams. In *Proc. of Pacific-Asia conf. on Knowledge Discovery and Data*

Mining (PAKDD'09), pages 363–375, 2009.

[145] M. May, B. Berendt, A. Cornuejols, J. Gama, F. Giannotti, A. Hotho, D. Malerba, E. Menasalvas, K. Morik, R. Pedersen, L. Saitta, Y. Saygin, A. Schuster, and K. Vanhoof. Research challenges in ubiquitous knowledge discovery. In *Next Generation of Data Mining*, Data Mining and Knowledge Discovery, pages 131–150. Boca Raton, FL: Chapman & Hall/CRC Press, 2008.

[146] O. Mazhelis and S. Puuronen. Comparing classifier combining techniques for mobile-masquerader detection. In *Proc. of the 2nd int. conf. on Availability, Reliability and Security (ARES '07)*, pages 465–472. IEEE Computer Society, 2007.

[147] K. Merrick and M. Maher. Motivated learning from interesting events: Adaptive, multitask learning agents for complex environments. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 17 (1):7–27, 2009.

[148] L. Minku, A. White, and X. Yao. The impact of diversity on on-line ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 99(1), 2009. online.

[149] J. Moreira. *Travel time prediction for the planning of mass transit companies: a machine learning approach*. PhD thesis, Faculty of Engineering of University of Porto, 2008.

[150] R. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. SpringerVerlag, 2004.

[151] F. Mourao, L. Rocha, R. Araujo, T. Couto, M. Goncalves, and W. Meira. Understanding temporal aspects in document classification. In *WSDM '08: Proc. of the int. conf. on Web search and web data mining*, pages 159–170. ACM, 2008.

[152] A. Narasimhamurthy and L. Kuncheva. A framework for generating data to simulate changing environments. In *Proc. of the 25th IASTED int. Multi-Conference, Artificial Intelligence and Applications (AIAP'07)*, pages 384–389. ACTA Press, 2007.

[153] K. Nishida. *Learning and Detecting Concept Drift*. PhD thesis, Hokkaido University, Japan, 2008.

[154] K. Nishida and K. Yamauchi. Detecting concept drift using statistical testing. In *Proc. of Discovery Science, the 10th int. conf., DS 2007*, volume

4755 of *LNCS*, pages 264–269. Springer, 2007.

[155] K. Nishida, K. Yamauchi, and T. Omori. Ace: Adaptive classifiers-ensemble system for concept-drifting environments. In *Proc. of the 6th int. workshop on Multiple Classifier Systems (MCS 2005)*, volume 3541 of *LNCS*, pages 176–185. Springer, 2005.

[156] M. Nunez, R. Fidalgo, and R. Morales. Learning in environments with unknown dynamics: Towards more robust concept learners. *Journal of Machine Learning Research*, 8:2595–2628, 2007.

[157] J. Oommen and L. Rueda. Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments. *Pattern Recognition*, 39(3):328–341, 2006.

[158] C. K. Park and P. Bas. A model for prediction of transient response to the change of fuel feed rate to a circulating fluidized bed boiler furnace. *Chemical Engineering Science*, 52:3499–3509, 1997.

[159] A. Patcha and J. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(12): 3448–3470, 2007.

[160] J. Patist. Optimal window change detection. In *Proc. of the 7th IEEE int. conf. on Data Mining Workshops (ICDMW '07)*, pages 557–562. IEEE Computer Society, 2007.

[161] A. Pawling, N. Chawla, and G. Madey. Anomaly detection in a mobile communication network. *Computational & Mathematical Organization Theory*, 13(4):407–422, 2007.

[162] M. Pechenizkiy, A. Tourunen, T. Karkkainen, A. Ivannikov, and H. Nevalainen. Towards better understanding of circulating fluidized bed boilers: a data mining approach. In *Proc. of ECML/PKDD workshop on Practical Data Mining*, pages 80–83, 2006.

[163] N. Pelekis, B. Theodoulidis, I. Kopanakis, and Y. Theodoridis. Literature review of spatio-temporal database models. *The Knowledge Engineering Review*, 19(3):235–274, 2004.

[164] N. Poh, R. Wong, J. Kittler, and F. Roli. Challenges and research directions for adaptive biometric recognition systems. In *Proc. of the 3rd int. conf. on Advances in Biometrics (ICB 2009)*, volume 5558 of *LNCS*, pages 753–764. Springer, 2009.

[165] M. Pourkashani and M. Kangavari. A cellular automata approach to detecting concept drift and dealing with noise. In *Proc. of the 2008 IEEE/ACS int. conf. on Computer Systems and Applications (AICCSA '08)*, pages 142–148. IEEE Computer Society, 2008.

[166] M. Procopio, J. Mulligan, and G. Grudic. Learning terrain segmentation with classifier ensembles for autonomous robot navigation in unstructured environments. *Journal of Field Robotics*, 26(2):145–175, 2009.

[167] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[168] S. Ramamurthy and R. Bhatnagar. Tracking recurrent concept drift in streaming data using ensemble classifiers. In *Proc. of the 6th int. conf. on Machine Learning and Applications (ICMLA '07)*, pages 404–409. IEEE Computer Society, 2007.

[169] P. Rashidi and D. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 39(5):949–959, 2009.

[170] S. Raudys. *Statistical and neural classifiers: an integrated approach to design*. Springer-Verlag, 2001.

[171] S. Raudys and A. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, 1991.

[172] S. Raudys and A. Mitasiunas. Multi-agent system approach to react to sudden environmental changes. In *Proc. of the 5th int. conf. on Machine Learning and Data Mining in Pattern Recognition (MLDM 2007)*, volume 4571 of *LNCS*, pages 810–823. Springer, 2007.

[173] S. Raudys and I. Zliobaite. Prediction of commodity prices in rapidly changing environments. In *Pattern Recognition and Data Mining, Proc. of the 3rd int. conf. on Advances in Pattern Recognition (ICAPR 2005)*, 2005.

[174] S. Raudys and I. Zliobaite. The multi-agent system for prediction of financial time series. In *Proc. of the 8th int. conf. on Artificial Intelligence and Soft Computing (ICAISC 2006)*, volume 4029 of *LNAI*, pages 653–662, 2006.

[175] T. Reinartz. A unifying view on instance selection. *Data Mining and Knowledge Discovery*, 6(2):191–210, 2002.

[176] H. Richter and S. Yang. Learning behavior in abstract memory schemes for dynamic optimization problems. *Soft Computing*, 13(12):1163–1173,

2009.

[177] J. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767, 2002.

[178] J. Rodriguez and L. Kuncheva. Combining online classification approaches for changing environments. In *Proc. of the 2008 joint IAPR int. workshop on Structural, Syntactic, and Statistical Pattern Recognition (SSPR & SPR '08)*, volume 5342 of *LNCS*, pages 520–529. Springer, 2008.

[179] P. Rohlfshagen, P. Lehre, and X. Yao. Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change. In *Proc. of the 11th annual conf. on Genetic and evolutionary computation (GECCO '09)*, pages 1713–1720. ACM, 2009.

[180] M. Rosenstein, Z. Marx, L. Kaelbling, and T. Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, 2005.

[181] A. Rozsypal and M. Kubat. Association mining in time-varying domains. *Intelligent Data Analysis*, 9(3):273–288, 2005.

[182] J. Saastamoinen. Modelling of dynamics of combustion of biomass in fluidized beds. *Thermal Science*, 8:107 – 126, 2004.

[183] M. Salganicoff. Density-adaptive learning and forgetting. In *Proc. of the 10th int. conf. on Machine Learning (ICML)*, pages 276–283, 1993.

[184] J. Scanlan, J. Hartnett, and R. Williams. Dynamicweb: Adapting to concept drift and object drift in cobweb. In *Proc. of the 21st Australasian joint conf. on Artificial Intelligence (AI '08)*, pages 454–460. Springer-Verlag, 2008.

[185] J. Schlimmer and R. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, 1986.

[186] M. Scholz and R. Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.

[187] B. Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2009.

[188] M. Skurichina and R. Duin. Bagging and the random subspace method for redundant feature spaces. In *Proc. of the 2nd int. workshop on Multiple Classifier Systems*, volume 2096 of *LNCS*, pages 1–10. Springer, 2001.

[189] M. Skurichina, S. Raudys, and R. Duin. k-nearest neighbors directed noise injection in multilayer perceptron training. *IEEE Transactions on Neural Networks*, 11(2):504–511, 2000.

[190] X. Song, C. Jermaine, S. Ranka, and J. Gums. A bayesian mixture model with linear regression mixing proportions. In *KDD '08: Proc. of the 14th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 659–667. ACM, 2008.

[191] E. Spinosa. *Novelty Detection in Data Streams*. PhD thesis, University of Sao Paulo, 2008.

[192] K. Stanley. Learning concept drift with a committee of decision trees. Technical Report UT-AI-TR-03-302, Computer Sciences Department, University of Texas, 2003.

[193] N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *KDD '01: Proc. of the 7th ACM SIGKDD int. conf. on Knowledge Discovery and Data Mining*, pages 377–382. ACM, 2001.

[194] B. Su, Y. Shen, and W. Xu. Modeling concept drift from the perspective of classifiers. In *Prof. of the conference on Cybernetics and Intelligent Systems, 2008 IEEE*, pages 1055–1060, 2008.

[195] T. Sung, N. Chang, and G. Lee. Dynamics of modeling in data mining: interpretive approach to bankruptcy prediction. *Journal of Management Information Systems*, 16(1):63–85, 1999.

[196] N. Syed, H. Liu, and K. Sung. Handling concept drifts in incremental learning with support vector machines. In *KDD '99: Proc. of the 5th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 317–321. ACM, 1999.

[197] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 23(9): 661–692, 2006.

[198] C. Tsai, C. Lee, and W. Yang. Mining decision rules on data streams in the presence of concept drifts. *Expert Systems with Applications: An International Journal*, 36(2):1164–1178, 2009.

[199] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9 (1):56–68, 2008.

[200] K. Ueno, X. Xi, E. Keogh, and D. Lee. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In *Proc. of the 6th int. conf. on Data Mining (ICDM '06)*, pages 623–632. IEEE Computer Society, 2006.

[201] H. Valizadegan and P. Tan. A prototype-driven framework for change detection in data stream classification. In *Proc. of IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)*, pages 88 – 95, 2007.

[202] J. van der Vorst, A. Beulens, W. de Wit, and P. van Beek. Supply chain management in food chains: improving performance by reducing uncertainty. *Int. Transactions in Operational Research*, 5(6):487–499, 1998.

[203] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18:77–95, 2002.

[204] C. Wang, D. Blei, and D. Heckerman. Continuous time dynamic topic models. In *Proc. of the 24th conf. in Uncertainty in Artificial Intelligence (UAI 2008)*, pages 579–586. UAI Press, 2008.

[205] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03: Proc. of the 9th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 226–235. ACM, 2003.

[206] H. Wang, J. Yin, J. Pei, P. Yu, and J. Yu. Suppressing model overfitting in mining concept-drifting data streams. In *KDD '06: Proc. of the 12th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 736–741. ACM, 2006.

[207] G. Webb, M. Pazzani, and D. Billsus. Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11(1-2):19–29, 2001.

[208] B. Wenerstrom and C. Giraud-Carrier. Temporal data mining in dynamic feature spaces. In *Proc. of the 6th int. conf. on Data Mining (ICDM '06)*, pages 1141–1145. IEEE Computer Society, 2006.

[209] G. Widmer. Tracking context changes through meta-learning. *Machine Learning*, 27(3):259–286, 1997.

[210] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. In *Proc. of the European conf. on Machine Learning (ECML '93)*, pages 227–243. Springer-Verlag, 1993.

[211] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[212] D. Widyantoro. *Concept drift learning and its application to adaptive information filtering*. PhD thesis, Texas A&M University, 2003.

[213] D. Widyantoro and J. Yen. Relevant data expansion for learning concept drift from sparsely labeled data. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):401–412, 2005.

[214] J. Wu, D. Ding, X. Hua, and B. Zhang. Tracking concept drifting with an online-optimized incremental learning framework. In *MIR '05: Proc. of the 7th ACM SIGMM int. workshop on Multimedia information retrieval*, pages 33–40. ACM, 2005.

[215] T. Yamaguchi. Constructing domain ontologies based on concept drift analysis. In *Proc. of the workshop on Ontologies and Problem-Solving Methods (IJCAI-99)*, 1999.

[216] R. Yampolskiy and V. Govindaraju. Direct and indirect human computer interaction based biometrics. *Journal of computers*, 2(10):76–88, 2007.

[217] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Transactions on Evolutionary Computation*, 12(5):542–561, 2008.

[218] Y. Yang, X. Wu, and X. Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data Mining and Knowledge Discovery*, 13(3):261–289, 2006.

[219] Y. Yang, X. Wu, and X. Zhu. Conceptual equivalence for contrast mining in classification learning. *Data and Knowledge Engineering*, 67(3):413–429, 2008.

[220] P. Zhang, X. Zhu, and Y. Shi. Categorizing and mining concept drifting data streams. In *KDD '08: Proc. of the 14th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 812–820. ACM, 2008.

[221] J. Zhou, L. Cheng, and W. Bischof. Prediction and change detection in sequential data for interactive applications. In *Proc. of the 23rd national conf. on Artificial intelligence (AAAI)*, volume 2, pages 805–810. AAAI, 2008.

[222] I. Zliobaite. Introduction of new expert and old expert retirement under concept drift. In S. Singh and M. Singh, editors, *Progress in Pattern Recognition*, Advances in Pattern Recognition, chapter 7. Springer, 2007.

[223] I. Zliobaite. Ensemble learning for concept drift handling - the role of new expert. In *Poster Proceedings of the 5th Int. Conf. on Machine Learning and Data Mining in Pattern Recognition MLDM 2007*, pages 251–260. IBaI

publishing, 2007.

[224] I. Zliobaite. Expected classification error of the euclidean linear classifier under sudden concept drift. In *FSKD '08: Proc. of the 2008 5th Int. Conf. on Fuzzy Systems and Knowledge Discovery*, pages 29–33. IEEE Computer Society, 2008.

[225] I. Zliobaite. On use of historical information under sudden and gradual concept drift. vilnius university, faculty of mathematics and informatics. Technical Report 2009-02, Vilnius University, 2009. URL `http://sites.google.com/site/zliobaite/Zliobaite_historInfo.pdf`.

[226] I. Zliobaite. Combining time and space similarity for small size learning under concept drift. In *Proc. of ISMIS 2009 - 18th International Symposium on Methodologies for Intelligent Systems*, volume 5722 of *LNCS*, pages 412–421. Springer-Verlag, 2009.

[227] I. Zliobaite. Instance selection method (fish) for classifier training under concept drift. Technical Report 2009-01, Vilnius University, Faculty of Mathematics and Informatics, 2009. URL `http://sites.google.com/site/zliobaite/Zliobaite_WKDD_full.pdf`.

[228] I. Zliobaite. Learning under concept drift: an overview. Technical report, Vilnius University, Faculty of MAthematics and Computer Science, 2009. URL `http://sites.google.com/site/zliobaite/Zliobaite_CDoverview.pdf`.

[229] I. Zliobaite. On recognition of seasonal predictability in sligro product sales. technical report TR-Jul2009, Eindoven University of Technology, 2009. URL `http://www.win.tue.nl/~mpechen/projects/sligro/TR-Jul2009.pdf`.

[230] I. Zliobaite. Combining similarity in time and space for training set formation under concept drift. *under review*, 2010.

[231] I. Zliobaite and T. Krilavicius. Clan: Clustering for credit risk assessment. An entry to pakdd 2009 data mining competition, Vilnius University and Vytautas Magnus University, 2009.

[232] I. Zliobaite and L. Kuncheva. Theoretical window size for classification in the presence of sudden concept drift. *under review*, 2010.

[233] I. Zliobaite and L. Kuncheva. Determining the training window for small sample size classification with concept drift. In *Proc. of 2009 IEEE International Conference on Data Mining Workshops, the 1st Int. Workshop on Transfer*

Mining (TM-09), pages 447–452. IEEE Computer Society, 2009.

[234] I. Zliobaite, J. Bakker, and M.Pechenizkiy. Towards context aware food sales prediction. In *Proc. of the 21st Benelux Conf. on Artificial Intelligence BNAIC-09*, pages 449–450, 2009.

[235] I. Zliobaite, J. Bakker, and M. Pechenizkiy. Omfp: An approach for online mass flow prediction in cfb boilers. In *Discovery Science, proc. of the 12th int. conf. (DS-09)*, number 5808 in LNAI, pages 272–286. Springer Berlin Heidelberg, 2009.

[236] I. Zliobaite, J. Bakker, and M. Pechenizkiy. Towards context aware food sales prediction. In *Proc. of 2009 IEEE International Conference on Data Mining Workshops, Int. Workshop on Domain Driven Data Mining (DDDM09)*, pages 94–99. IEEE Computer Society, 2009.

Publications by the Author

Periodic

1. Kuncheva, L.I. and Žliobaitė, I. (2009). On the Window Size for Classification in Changing Environments. Intelligent Data Analysis 13(6), p. 861-872. ISSN:1088-467X [ISI]

2. Žliobaitė, I. (2009). Combining Time and Space Similarity for Small Size Learning under Concept Drift. Proc. of ISMIS 2009, the 18th int. symposium on Methodologies for Intelligent Systems, book: Foundations of Intelligent Systems, Zhong, N.; Ras, Z.W.; Tsumoto, S.; Suzuki, E. (Eds.), (LNCS 5722), p. 412-421. ISSN: 0302-9743 [ISI proceedings]

3. Žliobaitė, I.,Bakker, J., Pechenizkiy, M. (2009). OMFP: An Approach for Online Mass Flow Prediction in CFB Boilers. Proc. of Discovery Science, the 12th int. conf. DS 2009 (LNAI 5808), p. 272-286 . ISSN: 0302-9743 [ISI proceedings]

4. Žliobaitė, I., Bakker, J., Pechenizkiy, M. (2009). Towards Context Aware Food Sales Prediction. Extended abstract. Proc. of the 21st Benelux conf. on Artificial Intelligence BNAIC-09, p. 449-450. ISSN:1568-7805

5. Žliobaitė, I. (2007). Introduction of New Expert and Old Expert Retirement under Concept Drift. Book: Progress in Pattern Recognition, series: Advances in Pattern Recognition. S. Singh, M. Singh (Eds.) 2007, XIII, p.64-74. ISSN: 1617-7916 [ISI proceedings]

6. Žliobaitė, I. (2007). Ensemble Learning for Concept Drift Handling - the Role of New Expert. Poster proc. of the 5th int. conf. on Machine Learning

and Data Mining in Pattern Recognition MLDM 2007, book: Machine Learning and Data Mining in Pattern Recognition, Petra Perner (Ed.). IBaI publishing, p. 251-260. ISSN: 1864-9734

7. Raudys, Š., Žliobaitė, I. (2006). The Multi-Agent System for Prediction of Financial Time Series. Proc. of the 8th int. conf. on Artificial Intelligence and Soft Computing ICAISC 2006 (LNAI 4029), p. 653-662. ISSN: 0302-9743 [LNAI was ISI at the time of publication, now ISI proceedings]

8. Raudys, Š., Žliobaitė, I. (2005). Prediction of Commodity Prices in Rapidly Changing Environments. Pattern Recognition and Data Mining, Proc. of the 3rd int. conf. on Advances in Pattern Recognition, ICAPR 2005 (LNCS 3686), p. 154-163. ISSN: 0302-9743 [LNCS was ISI at the time of publication, now ISI proceedings]

Reviewed conference publications

9. Bakker, J., Pechenizkiy, M, Žliobaitė, I., Ivannikov, A. and Karkkainen, T. (2009). Handling Outliers and Concept Drift in Online Mass Flow Prediction in CFB Boilers. Proc. of the 3rd int. workshop on Knowledge Discovery from Sensor Data (SensorKDD-09), p. 13-22, ACM. ISBN:978-1-60558-668-7 [The Best Paper Award]

10. Žliobaitė, I., Bakker, J., Pechenizkiy, M. (2009). Towards Context Aware Food Sales Prediction. Proc. of 2009 IEEE int. conf. on Data Mining Workshops, int. workshop on Domain Driven Data Mining (DDDM09), p. 94-99. ISBN: 978-8-7695-3895-2 [IEEE/IEE]

11. Žliobaitė, I., Kuncheva, L. (2009). Determining the Training Window for Small Sample Size Classification with Concept Drift. Proc. of 2009 IEEE int. conf. on Data Mining Workshops, the 1st int. workshop on Transfer Mining (TM-09), p. 447-452. ISBN: 978-8-7695-3895-2 [IEEE/IEE]

12. Žliobaitė, I. (2008). Expected Classification Error of the Euclidean Linear Classifier under Sudden Concept Drift. Proc. of the 5th int. conf. on Fuzzy Systems and Knowledge Discovery (FSKD 2008). IEEE Computer Society: vol 2, p. 29-33. ISBN: 978-0-7695-3305-6 [ISI proceedings]

13. Kuncheva, L., Žliobaitė, I. (2008). Linear Discriminant Classifier (LDC) for Streaming Data with Concept Drift. Ext. abstract. Proc. of SSPR/SPR 2008, joint IAPR int. workshop (LNCS 5342), p: 4. ISSN: 0302-9743 [SpringerLINK]

Journal submissions under review

- Žliobaitė, I. and Kuncheva L. (2009). Theoretical Window Size for Classification in the Presence of Sudden Concept Drift. Submitted to Information Processing Letters, Elsevier.
- Bakker, J., Pechenizkiy, M, Žliobaitė, I., Ivannikov, A. and Karkkainen, T. (2009). Online Mass Flow Prediction in CFB Boilers with Explicit Detection of Sudden Concept Drift. Submitted to ACM SIGKDD Explorations.
- Žliobaitė, I. (2010). Combining Similarity in Time and Space for Training Set Formation under Concept Drift. Submitted to Intelligent Data Analysis.

Unreviewed manuscripts [2]

- Žliobaitė, I. and Krilavičius, T. (2009). CLAN: Clustering for Credit Risk Assessment. An entry to PAKDD 2009 Data Mining Competition.
- Žliobaitė, I. (2009). On Use of Historical Information under Sudden and Gradual Concept Drift. Vilnius University, Technical Report 2009-02.
- Žliobaitė, I. (2009). Instance selection method (FISH) for Classifier Training under Concept Drift. Vilnius University, Technical Report 2009-01.
- Žliobaitė, I. (2009). On Recognition of Seasonal Predictability in SLIGRO product sales. TUE, Technical Report.
- Žliobaitė, I. (2009). Learning under Concept Drift: an Overview. Vilnius University, Technical Report (36p.).

[2] Available online at http://sites.google.com/site/zliobaite/publications

Curriculum Vitae

Indrė Žliobaitė graduated from Vilnius Lyceum of Sciences (gymnasium) in 2000. She received BSc degree in Economics and Business with specialization in Economics and Finance from Stockholm School of Economics in Riga (Latvia) in 2003. In 2006 she graduated from Vilnius University (Lithuania) receiving MSc degree in Informatics. From 2006 to 2010 she was enrolled into a PhD study program (Informatics) at Vilnius University.

I. Žliobaitė worked as a visiting researcher at Bangor University (UK) for three months in 2008. In 2009 she had a three months internship at Helsinki Institute for Information Technologies (HIIT). I. Žliobaitė worked as a visiting researcher at Eindhoven University of Technology for four months in 2009. She gave invited talks on adaptive learning strategies under concept drift in all the three institutions.

I. Žliobaitė is a reviewer for the following journals: Pattern Recognition, Pattern Recognition Letters, European Journal of Operational Research, Journal of Computational and Graphical Statistics, Journal of Pattern Recognition Research.

Vocabulary - Žodynėlis

base learner - bazinis klasifikatorius
baseline - bazinis metodas
change point - pokyčio taškas
concept drift - koncepcijos pokytis
context aware - kontekstinis
data mining - duomenų gavyba
data source - duomenų šaltinis
gradual drift - palaipsnis pokytis
instance - vektorius
instance based learning - mokymas pagal vektorius
label - klasė
moving average - slenkantis vidurkis
peer methods - lyginamieji metodai
recurring concepts - pasikartojantis pokytis (pasikartojančios koncepcijos)
sequential learning - mokymas paeiliui
source - šaltinis
sudden drift - staigus pokytis
supervised learning - mokymas su mokytoju
training window - mokymo langas
unsupervised learning - mokymasis

Summary in Lithuanian (Santrauka)

Šiandieninėje, dinamiškai besikeičiančioje aplinkoje reikalingi adaptyvūs duomenų gavybos metodai. Nepageidaujamų laiškų klasifikatoriai, rekomendavimo bei rinkodaros, įsilaužimų į kompiuterinius tinklus aptikimo, verslo rodiklių prognozavimo bei sprendimų priėmimo sistemos turi nuolat persimokyti reaguoti į besikeičiančius duomenis. Stacionarioje aplinkoje kuo daugiau mokymo duomenų - tuo tikslesnis modelis. Besikeičiančioje aplinkoje seni duomenys blogina tikslumą. Tokiu atveju, vietoje visų turimų istorinių duomenų panaudojimo, gali būti tikslingai išrenkama tik tam tikra jų dalis, pvz. naudojamas mokymo langas (tik naujausi duomenys).

Tiriamojo darbo objektas yra adaptyvūs mokymo metodai, kurie remiasi kryptingu mokymo imties formavimu. Patobulintos žinomos mokymo strategijos esant staigiems, palaipsniams ir pasikartojantiems pokyčiams. Sukurti ir eksperimentiškai aprobuoti keturi adaptyvaus mokymo imties formavimo algoritmai, kurie leidžia pagerinti klasifikavimo bei prognozavimo tikslumą besikeičiančiose aplinkose, esant atitinkamai kiekvienam iš trijų pokyčių tipų. Naudojant generuotus bei realius duomenis, eksperimentiškai parodytas klasifikavimo bei prognozavimo tikslumo pagerėjimas, lyginant su visų istorinių duomenų naudojimu mokymui, bei žinomais šioje srityje naudojamais adaptyviais mokymo algoritmais. Sukurta metodika pritaikyta pramoninio katilo atvejui, jungiančiam kelis aplinkos pokyčių tipus.

Index

P

R

S

T

U

W