Multiple adaptive mechanisms for predictive models on streaming data

RASHID BAKIROV

A thesis submitted in partial fulfilment of the requirements of
Bournemouth University for the degree of

Doctor of Philosophy

February, 2017

Abstract

Making predictions on non-stationary streaming data remains a challenge in many application areas. Changes in data may cause a decrease in predictive accuracy, which in a streaming setting require a prompt response. In recent years many adaptive predictive models have been proposed for dealing with these issues. Most of these methods use more than one adaptive mechanism, deploying all of them at the same time at regular intervals or in some other fixed manner. However, this manner is often determined in an ad-hoc way, as the effects of adaptive mechanisms are largely unexplored. This thesis therefore investigates different aspects of adaptation with multiple adaptive mechanisms with the aim to increase knowledge in the area, and propose heuristic approaches for more accurate adaptive predictive models. This is done by systematising and formalising the "adaptive mechanism" notion, proposing a categorisation of adaptive mechanisms and a metric to measure their usefulness, comparing the results after deployment of different orders of adaptive mechanisms during the run of the predictive method, and suggesting techniques on how to select the most appropriate adaptive mechanisms.

The literature review suggests that during the prediction process, adaptive mechanisms are selected to be deployed in a certain order which is usually fixed beforehand at the design time of the algorithm. For this reason, it was investigated whether changing the selection method for the adaptive mechanisms significantly affects predictive accuracy and whether there are certain deployment orders which provide better results than others. Commonly used adaptive mechanism selection methods are then examined and new methods are proposed.

A novel regression ensemble method which uses several common adaptive mechanisms has been developed to be used as a vehicle for the experimentation. The predictive accuracy and behaviour of adaptive mechanisms while predicting on different real world datasets from the process industry were analysed. Empirical results suggest that different selection of adaptive mechanisms result in significantly different performance. It has been found that while some adaptive mechanisms adapt the predictive model better than others, there is none which is the best at all times. Finally, flexible orders of adaptive mechanisms generated using the proposed selection techniques often result in significantly more accurate models than fixed orders commonly used in literature.

Acknowledgements

This thesis would not have been possible without those around me while I was writing it. Firstly I would like to thank to my supervisors, Bogdan Gabrys and Damien Fay for their continuous guidance, sharing of their knowledge and shaping me as a researcher. I also thank my former supervisor Indre Žliobaitė for her input at the early stages of my PhD.

Many academics and peers have assisted the completion of the thesis in various ways during this time, for which I am very grateful. In particular, Petr Kadlec has provided me with his Matlab code. Marcin Budka, Manuel Martín Salvador and Neil Vaughaun have given helpful feedback my thesis. Alejandro Tabas discussed some of the math with me. Akanda Ashraf, Bastian Fraune and Mohsen Amiribesheli shared their opinions about various plots and provided my wrists with exercise during daily table football games. Jan Walter Schroeder gave me general advices on writing the thesis.

I was blessed to have made many truly great friends during my PhD time, who made me feel at home in Bournemouth and have immensely contributed to my development as a person. For this, I deeply thank every single one of them - you know who you are. My special thanks goes to Diana for her constant support and encouragement. I would also like to thank Bomojam crew for the magical hours of music that we made together and lunchers for making our meals at the university something to look forward to. Also huge thanks to Kim, who has greatly contributed to my preparation for the viva.

Many people, including my former colleagues at University of Siegen, ABB Research Germany and PwC Germany have helped and encouraged me on this path. Among others, I thank Klaus Hartmann, Chris Stich and Markus Anderle for their supervision and many recommendation letters, and Seyed Eghbal Ghobadi for getting me interested in Machine Learning.

Staff at Bournemouth University has been most helpful during my PhD. In particular, I thank Naomi Bailey for the administrative assistance, Natalie Andrade and Malcolm Green for the help with finance related matters, and Shaun Bendall for the assistance with computing cluster.

I am grateful to my final viva examiners Emili Balaguer-Ballester and Vasile Palade, as well as transfer viva examiners Raian Ali and Hammadi Nait-Charif for their constructive and helpful comments on my theses. I thank anonymous reviewers, whose comments on my publications have improved my work and members of academia for the all their research that I was inspired by. I would also like to thank the creators of free software that have helped me greatly during writing of the thesis. These include LaTeX, TexStudio, briss, Excel2Latex, Putty and WinScp. My thanks also goes to the members of stackexchange.com whose questions and answers has solved many of Matlab and LaTeX riddles for me. I

Finally, I am ever thankful to my parents and my sister, who were always there for me during this time and in fact for whole my life.

Contents

List of Figures

List of Tables

Notation

Symbol	Description
y	Target value
$\psi(\cdot)$	Function of the real process which generates the data
$\boldsymbol{x}$	Input instance
ξ	Noise
$\hat{y}$	Predicted output
$f(\cdot)$	Prediction function/predictive model
Θ_f	Set of parameters of f
$\boldsymbol{\mathcal{V}}$	Whole dataset, $\boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{V}}_1 \cup \cdots \cup \boldsymbol{\mathcal{V}}_K$
$\boldsymbol{\mathcal{V}}_k$	Data batch at time k, $\boldsymbol{\mathcal{V}}_k = \{\boldsymbol{X}_k, \boldsymbol{y}_k\}$
$\boldsymbol{X}_k$	Input data in batch $\boldsymbol{\mathcal{V}}_k$
$\boldsymbol{y}_k$	Target data in batch $\boldsymbol{\mathcal{V}}_k$
N	Total number of instances
N_*	Number of instances in a batch (assuming batches of a same size)
N_k	Number of instances in k-th batch
$\mathcal{P}(\cdot)$	Probability
$E[\cdot]$	Expected value
λ	Forgetting (decay) factor
τ	Time
l	Moving window size
$s(\cdot)$	Expert / predictor
S	Set of experts, $S = \{s_1, \cdots, s_I\}$
w	Weight of an expert
W	Set of weights, $W = \{w_1, \cdots, w_I\}$
ν	Weights update function
β	Weights update factor
ϵ	Error value, $\epsilon = \hat{y} - y$
$\boldsymbol{\epsilon}$	Errors on the whole dataset
$\boldsymbol{\epsilon}_k$	Errors on batch $\boldsymbol{\mathcal{V}}_k$
c	Class label
C	Set of class labels, $C = \{c_1, \cdots, c_J\}$
Z	Product of probabilities of all inputs for Naive Bayes classifier
ζ	Threshold to add a new expert for (Kolter & Maloof, 2005)
β	Threshold to remove an expert for Dynamic Weighted Majority algorithm
Ω	Period (Dynamic Weighted Majority)
ω_c	sum of weighted predictions for a class c (Dynamic Weighted Majority)
ϕ	Distribution
α	Significance level

Background

Symbol	Description
$\boldsymbol{X}$	Input data
$\boldsymbol{Y}$	Output data
$\tilde{\boldsymbol{Y}}$	Estimates of $\boldsymbol{Y}$
N	Number of data instances
M	Number of input variables
K	Number of output variables
L	Number of latent variables
$\boldsymbol{T}$	Score matrix
$\boldsymbol{t}$	Latent vector
$\boldsymbol{D}$	Corresponding loading matrix
$\boldsymbol{d}$	Column vector of $\boldsymbol{D}$
$\boldsymbol{U}$	Score matrix
$\boldsymbol{u}$	Latent vector
$\boldsymbol{Q}$	Corresponding loading matrix
$\boldsymbol{q}$	Column vector of $\boldsymbol{Q}$
$\boldsymbol{V}$	Input data residuals
$\boldsymbol{F}$	Output data residuals
$\boldsymbol{R}$	Regression residuals
$\boldsymbol{B}$	Regression weights matrix
$\boldsymbol{b}$	Column vector of $\boldsymbol{B}$
$(\cdot)^\top$	Transpose of a matrix, e.g. $\boldsymbol{P}^\top$

Recursive Partial Least Squares (RPLS)

Symbol	Description
m	Feature number
$\mathcal{D}_{i,m}$	Descriptor of m-th feature of i-th expert
$\boldsymbol{\mathcal{D}}$	Descriptors matrix
$\boldsymbol{\mathcal{V}}^{tr}$	Training data
$v(\boldsymbol{x}_n)$	Weight for n-th instance
$\Phi(\mu_n^m, \boldsymbol{\Sigma})$	Two-dimensional Gaussian kernel function
$\mu = (x_n^m, y_n)$	Mean value of Gaussian kernel function
$\boldsymbol{\Sigma}$	Variance matrix of kernel function with σ at the diagonal positions
σ	Kernel width
w_i	Weight of i-th expert's prediction
p	p-value of t-test between two experts' prediction errors
$\boldsymbol{P}$	Matrix of pairwise p-values between prediction errors of all experts
$\boldsymbol{\mathcal{D}}^{\mathbf{0}}$	Old descriptors matrix before their recalculation
$\boldsymbol{\mathcal{D}}^{\mathbf{1}}$	New descriptors matrix after their recalculation
δ_0	Weight of an old descriptor
δ_1	Weight of a new descriptor

Simple Adaptive Batch Learning Ensemble (SABLE)

Symbol	Description
f^-	*A priori* prediction function (before the adaptation)
f^+	*A posteriori* prediction function (after the adaptation)
g	Adaptive Mechanism (AM)
G	Set of AMs, $G = \{g_1, \cdots, g_H\}$
Θ_g	Set of parameters of g
g_{h_k}	AM deployed on batch k
f_k^{+h}	Predictive model adapted using g_{h_k}, $f_k^{+h} = f_{k+1}^{-h}$
r	Number of steps for exhaustive deployment/retrospective correction
$\boldsymbol{\hat{Y}_k}$	Predictions of $\boldsymbol{Y}_k$ by different models in the AM exhaustive deployment tree
$\mathcal{S}$	Cross-validation training subset
$\not{\mathcal{S}}$	Cross-validation test subset
$\langle\ \rangle$	Error measure
$\langle\ \rangle^\times$	Cross-validated error measure
h_k^{opt}	Index of optimal AM for batch k
$\hat{h}_k$	Index of AM for batch k predicted by meta-classifier
υ	Meta-classifier function
$\boldsymbol{\chi}$	Meta-features vector
$\mathcal{C}$	Meta-classifier classification cost matrix

Adaptation

Declaration

The work contained in this thesis is the result of my own investigations and has not been accepted nor concurrently submitted in candidature for any other award.

Chapter 1

Introduction

1.1 Background and motivation

An essential element of human understanding of the complex systems and phenomena is creation of models. “Models are graphical, mathematical (symbolic), physical, or verbal representation or simplified version of a concept, phenomenon, relationship, structure, system, or an aspect of the real world.”[1] The degree of how well the model reflects its object is called *accuracy*. Models are usually constructed using observations of the object of modelling which may entail information about its inputs, structure, and outputs. Modelling serves to explain the objects and provide insights about their behaviour. Models have been very useful in areas such as physics, quantitative finance, marketing, industrial processes, social sciences, weather forecasting and others. With the rise of computing power and the ubiquitous recording of data, the usage of models, and particularly computational models, has been rapidly increasing.

Often, models describe entities which are inherently subject to changes. For instance, in industrial process modelling, these changes may be caused by degradation of the equipment, in climate modelling by the climate change phenomenon, in financial modelling by sudden crises, in enterprise modelling by merger of the organisations. *After* the changes, the accuracy of the model which was constructed *before* the change and was based on the outdated assumptions, will often deteriorate. For these kind of changing environments, the need for model’s adaptation is paramount. Depending on the type of the model there are different ways to adapt it.

This thesis specifically focuses on predictive modelling in machine learning. Machine learning is the science of learning patterns from observed data, to make predictions on new, previously unseen data. These patterns are often formulated as models, which are built by applying algorithms to the historical data. Models can be then used to calculate the desired predictions.

In some cases, these predictive models concern static datasets. However, quite often the predictive models are being applied to the new data which keeps getting generated by the underlying processes. This type of data is called streaming data (Wang et al., 2003). In fact, with the current

[1] http://www.businessdictionary.com/definition/model.html

advances in data storage, database and data transmission technologies, streaming data becomes increasingly relevant. This fact may pose additional challenges, one of them being the changes in the data generating process.

If the model remains unchanged for the whole duration of machine learning process, it is called a static machine learning model. However in situations when such models are used to describe dynamic processes, their accuracy might degrade. Specifically, in machine learning, this effect may be caused by changes in data distribution (Žliobaitė, 2011), changes in features' relevance (Fern & Givan, 2000), novel classes and features (Gabrys & Bargiela, 1999; Masud et al., 2013). For instance, in manufacturing settings, where the process is being observed by various sensors, gradual wearing of a sensor, its sudden failure, removal, replacement or addition of a new type of sensors can cause all of the above mentioned changes. Other examples of real world changes affecting relevant predictive models are special events, terrorist attacks or competitions which influence plane tickets demand (Riedel & Gabrys, 2007; Lemke et al., 2009, 2013), network intrusions which adapt to bypass the installed firewalls (Lee et al., 2000), seasonal changes influencing many areas such as electricity or gas consumption (Kolter & Maloof, 2007), change in lighting which can seriously affect image recognition in videos (Thrun et al., 2006). It has been shown that a changed environment, which is no longer being reflected by the model, contributes to the deterioration of model's accuracy over time, (Schlimmer & Granger, 1986a; Gabrys & Bargiela, 1999; Street & Kim, 2001; Gabrys, 2004; Klinkenberg, 2004; Kolter & Maloof, 2007; Sahel et al., 2007; Martín Salvador et al., 2016c).

In many cases it is possible to alleviate accuracy deterioration. Several options are available for this purpose. One of them is regular retraining, or in other words, creation of new models. This could be time consuming and in some cases impossible, for example when the required historical data is not available any more. Alternatively, one might employ various proposed online learning and adaptation approaches. Their purpose is keeping the existing model and making it *self-adapt* to the possible changes in environment.

Adaptive approaches range from high-level general adaptation concepts (Holland, 1992; Grisogono, 2006) to more practical adaptive machine learning algorithms (Littlestone et al., 1991; Carpenter et al., 1992; Jacobs, 1995; Kolter & Maloof, 2007; Bouchachia et al., 2007; Tsakonas & Gabrys, 2013; Žliobaitė & Gabrys, 2014). They show better results while predicting on synthetic and real-world dynamic data than traditional non-adaptive models. It has been proposed that adaptive machine learning be applied to many different areas, such as chemical processing (Kadlec et al., 2011; Grbić et al., 2013), network intrusion detection (Lee et al., 2000; Haag et al., 2007), video image and concept recognition (Thrun et al., 2006; Yang et al., 2007) and others. Adaptive models can originate in different research fields: statistical and probabilistic approaches, machine learning, computational intelligence and different combinations thereof.

Typically adaptation operates by reducing the weight applied to the parts of the historical data that are less similar to the current data, which may be implemented in a variety of ways. In addition, recent adaptive methods use more than one Adaptation Mechanisms (AMs). Employing multiple AMs is more versatile than using a single one and has been found to lead to superior prediction performance (Kadlec & Gabrys, 2011; Jin et al., 2015b). In practice, most research has used AMs deployed in a fixed manner with the choice of deployment order set at model design time. The common choice is to deploy all of the AMs at the same time, however this does not always deliver the best results with some AMs potentially cancelling the effect of others or changing the model more than required (Bakirov et al., 2017).

The possibility of having multiple adaptive mechanisms working together in one system provides additional ways of handling the adaptation. However this also makes the control over it more complex. As will be seen from Chapter 4, the choice of adaptive mechanism needs to be carefully considered to achieve optimal results. Otherwise there is a risk of degrading predictive accuracy. To the best of author's knowledge, no works explicitly research multiple adaptive mechanisms related issues for streaming data prediction.

The main topic of this PhD project is therefore the investigation of predictive systems with multiple adaptive mechanisms in a streaming non-stationary data setting, with the aim to ultimately assist in improving adaptation capabilities of such systems, which would in turn result in higher accuracy rates.

1.2 Aims and objectives of the PhD project

While realizing the complexity of the described issues and challenges, this thesis tackles certain aspects of the usage of multiple adaptive mechanisms. The aims of the project are listed in the following sections.

1.2.1 Analysis and categorization of methods with multiple adaptive methods

Despite the large volume of research dealing with adaptive predictive modeling, there seems to be a lack of work focusing explicitly on adaptation; it seems to be considered to have a secondary role in regards to the whole predictive algorithm. Therefore, one of the goals of the thesis is to facilitate research into adaptive predictive modeling, while focusing on adaptation mechanisms. The following objectives are specified for this purpose:

- Conducting a large survey of research dealing with this topic,
- Identifying the most commonly used adaptive mechanisms, and
- Categorizing them in a meaningful way, where categories include adaptation mechanisms with similar characteristics.

Completing these objectives will create a framework for research into adaptive mechanisms, conducted further in this thesis and elsewhere.

1.2.2 Investigation of the necessity of adaptive mechanism selection

It is quite feasible that the selection of AMs to deploy is a big factor in the predictive performance of the model. However, explicit research to support this proposition has not hitherto been conducted. Therefore, one goal of the thesis is devising a study which tests this proposal. This study includes the following objectives:

- Development of an algorithm for experimentation purposes which uses multiple AMs from identified categories to deal with changes.
- Development of an experiment which will investigate the importance of AM selection.
- Performing the experiments on datasets with different adaptation needs and analysis of results.

1.2.3 Research into strategies of adaptive mechanisms' deployment

It has been found that AM selection plays a crucial role in the predictive performance. Therefore a research into the ways and order of AM deployment will be conducted. The aim of this research is suggesting strategies for AM deployment (adaptive strategies) and identifying how these strategies affect the predictive performance. More concretely, it is aimed to achieve the following objectives:

- Implementing popular adaptive strategies using the predictive algorithm from the previous section.
- Suggesting adaptive strategies which can provide better results than the common ones.
- Analysing the results, identifying the circumstances where different adaptive strategies work well and the reasons behind it.

1.3 Original contributions

' The main scientific contributions of this thesis are:

- **Investigation of the importance of AM order selection (adaptive strategies).** Using an empirical experiment on the real process industry data provided by project partner Evonik Industries AG, it was established that the selection of AM order is a significant factor for the predictive accuracy of the algorithm.
- **Identification and performance analysis of different fixed and flexible AM strategies, suggestion of cross-validation based AM strategies and retrospective model correction technique.** The performance of common fixed adaptive strategies and suggested flexible strategies on real datasets were compared and it has been found that flexible strategies suggested in this thesis outperform other strategies most of the time.
- **Development of Simple Adaptive Batch Learning Ensemble (SABLE) online learning framework.** An adaptive framework in plug-and-play fashion, where a number of AMs are deployable in various manners has been developed for the experimentation purposes.

- **Categorisation and formalisation of adaptive mechanisms.** A comprehensive survey of adaptive predictive algorithms has been conducted. The results of the survey were a base for the categorisation of commonly used adaptive mechanisms. A formal notation of adaptation process has been also introduced.
- **Research into the criteria and training data of added global experts in streaming classification problems.** The effects of different sizes of training data for newly added experts, different conditions for their addition have been empirically analysed.

1.4 List of resulting publications

The research leading to these contributions has resulted in the following publications:

- Bakirov, R; Gabrys, B. Investigation of Expert Addition Criteria for Dynamically Changing Online Ensemble Classifiers with Multiple Adaptive Mechanisms In: Artificial Intelligence Applications and Innovations, 646–656, 30 September–2 October 2013, Paphos, Cyprus.
- Bakirov, R.; Gabrys, B. and Fay, D. On Sequences of Different Adaptive Mechanisms in Non-Stationary Regression Problems. In: 2015 International Joint Conference on Neural Networks 12 July–17 July 2015, Killarney, Ireland
- Bakirov, R.; Gabrys, B. and Fay, D. Augmenting Adaptation with Retrospective Model Correction for Non-Stationary Regression Problems. In: 2016 International Joint Conference on Neural Networks 24–29 July 2016 Vancouver, Canada.
- Bakirov, R.; Gabrys, B. and Fay, D., Multiple adaptive mechanisms for data-driven soft sensors Computers and Chemical Engineering, vol. 96, pp. 42–54, 2017.

1.5 Organisation of the thesis

Following this introductory chapter, the thesis is organised as follows. Chapter 2 presents background information about predictive modeling on streaming data and gives details of existing adaptive mechanisms. AMs' hierarchical categorization is proposed, and several types of base learners used in predictive methods with multiple adaptive mechanisms are introduced. The details of these predictive methods are given as well. Chapter 3 gives short information about the process industry, where the considered data sets for experiments are originating from, further introduces these datasets and analyses their characteristics in regards to exhibited changes. Chapter 4 looks into whether deployment of different AM sequences affects the accuracy of a predictive model. Here the predictive model with multiple AMs, which is used as a vehicle for experimentation, is introduced in detail. Chapter 5 analyses the effects of different AMs, identifies and conducts experiments with many traditional adaptive strategies as well as with the proposed novel ones. The results are compared and it is analysed which strategies provide better predictive accuracy and what are possible reasons for this. Chapter 6 gives an overview of the findings, provides concluding remarks and possible future research directions. An investigation into expert addition criteria for streaming classification ensembles is conducted in Appendix A.

Chapter 2

Learning and adaptation on streaming data

2.1 Introduction

With the rise of streaming data analytics, it is becoming increasingly more important to be able to incorporate new data in predictive models as well as to be able to react to changes in the data. A variety of adaptive learning methods on streaming data (e.g. (Schlimmer & Granger, 1986a; Kolter & Maloof, 2007; Kadlec & Gabrys, 2011; Elwell & Polikar, 2011; Lemke et al., 2013; Žliobaitė & Gabrys, 2014; Gomes Soares & Araújo, 2015b)) have been developed for these purposes. In recent times this type of methods often features multiple adaptive mechanisms, (e.g. (Castillo & Gama, 2006; Kolter & Maloof, 2007; Kadlec & Gabrys, 2011; Minku & Yao, 2012; Žliobaitė et al., 2012; Gomes Soares & Araújo, 2015b; Bakirov et al., 2017)) increasing the number of ways they can deal with these issues, and potentially providing better predictive performance.

This chapter introduces the key concepts in adaptive learning, which is the area of Machine Learning dealing with non-stationary data. The chapter starts by presenting general techniques of learning and adaptation on streaming data in Section 2.2. It lists the types of learning/adaptation in this setting, as well as the possible reasons why the adaptation might be needed (Section 2.3.1). Then in Section 2.4 it proceeds to explore adaptation in machine learning in a greater detail. The different types of adaptation mechanisms, starting with the ones which adapt the coverage of available training data and continuing with the adaptation of models' parameters and structure, are identified and categorised. In Section 2.5 a more detailed analysis of ensemble methods is given, since these are the methods which are the most relevant to the research produced in this thesis. Finally, the methods which employ multiple adaptive mechanisms are discussed in Section 2.6.

2.2 Learning on streaming data

In many applications, existing predictive models are being constantly applied to new data and after a certain time some or all true target values become available. This is the case for instance with credit scoring where the goodness of credit is revealed only after some time (Dal Pozzolo et al., 2015), process industry where lab measurements can be much slower than the process (Kadlec & Gabrys, 2011), stock market predictions (Hazan & Seshadhri, 2009) and other areas. The natural question is, how is it possible to use this new information to improve the accuracy of the models. To give a mathematical formulation, it is assumed that the data is generated by an unknown dynamic process which can be formulated as:

$$y = \psi(\boldsymbol{x}) + \xi, \tag{2.1}$$

where ψ is an unknown function, ξ a noise term, $\boldsymbol{x} \in \mathbb{R}^M$ is an input data instance, and y is the observed output. Then the predictive model as a function:

$$\hat{y} = f_0(\boldsymbol{x}, \Theta_f | \boldsymbol{\mathcal{V}}_0), \tag{2.2}$$

is considered, where $\hat{y}$ is the prediction, f_0 is an approximation (i.e. the model) of $\psi(\boldsymbol{x})$, and Θ_f is the associated parameter set. It is assumed that the f_0 was built using some training set $\boldsymbol{\mathcal{V}}_0 = \{\boldsymbol{X}_0, \boldsymbol{y}_0\}$. At some point, new data from the process, $\boldsymbol{\mathcal{V}}_1 = \{\boldsymbol{X}_1, \boldsymbol{y}_1\}$ is obtained.[1] The goal of learning in this case is to use new data, $\boldsymbol{\mathcal{V}}_1$, to improve the predictive model, f_0. As having a larger training dataset typically increases predictive accuracy, learning new data is useful especially in the cases when the initial dataset might have been insufficient to achieve the desired performance.

Methods to learn new data operate in several ways. Most straightforwardly, they combine both historical and new data to generate a new dataset, which will be used to train a new model, thereby updating the old model with new information (Figure 2.1(a)). Essentially a new predictive model

$$f_1(\boldsymbol{x}, \Theta_f | \boldsymbol{\mathcal{V}}_0 \cup \boldsymbol{\mathcal{V}}_1) \tag{2.3}$$

is created. This approach is possible for all types of learning algorithms (e.g. decision trees (Quinlan, 1993) (Breiman et al., 1984), model trees (Quinlan, 1992), SVM (Vapnik, 1995)) as long as the required data is available. However this is often not possible or not the ideal solution; historical data might not be available[2] any more or building a model from scratch might be too time or resource consuming. Methods called online learners, such as Recursive Least Squares (Plackett, 1950) or naive Bayes (Hastie et al., 2009) which instead of explicitly combining the data to form a new dataset, achieve the same effect without requiring access to old data are preferred for

[1] For brevity purposes, Equation 2.2 will be shortened as $\hat{y} = f_0(\boldsymbol{x})$ in this thesis.

[2] Even when all historical data is available, using it may also not be a good/possible choice when it is not relevant to or compatible with new data. This requires adaptation, discussed in Section 2.3

such cases. These algorithms store only the model characteristics (e.g. counts, linear coefficients, etc.) and update them when new data arrives (Figure 2.1(b)). This can be formulated as

$$f_1(\boldsymbol{x}, \Theta_f | f_0, \boldsymbol{\mathcal{V}}_1). \tag{2.4}$$

Online variations for many popular off-line machine learning algorithms specified above have been also developed (Cauwenberghs & Poggio, 2001; Hulten et al., 2001; Kivinen et al., 2004; Potts & Sammut, 2005).

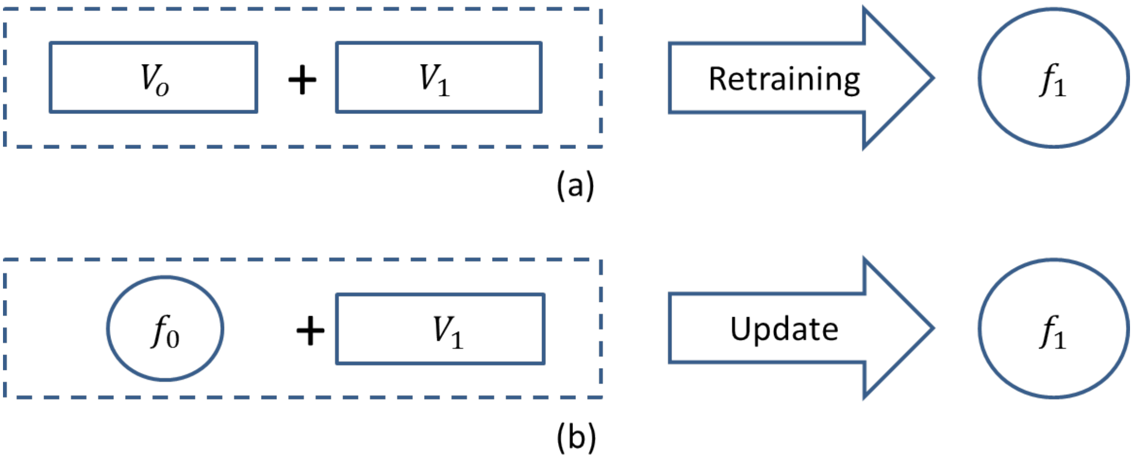


Figure 2.1: *Learning new data: a) using historical data b) online learning.*

Methods for learning on streaming data fall into two types; incremental and batch learning. Incremental learning is learning from a single new data instance.[3] While being able to learn one-by-one is certainly a powerful asset, in many practical situations the data becomes available in small batches over a period of time, as noted in (Polikar et al., 2001). For these cases, incremental learning is often redundant and potentially inefficient. The second group of models, batch learners, operate on whole batches of data simultaneously, and are better suited to this scenario. Naturally batch learners may also operate in the scenarios when data instances are arriving one-by-one by waiting until the required amount of new data is available, which however slows down the reaction to the changes. Batch learning allows the creation of new experts based on a considerable amount of data increasing the stability of predictions (Elwell & Polikar, 2011) and the measurement of metrics on a batch to potentially use them as meta-features (Alippi et al., 2012). Methods such as Naive Bayes and online versions of popular algorithms, for instance Recursive Least Squares Estimator (Plackett, 1950), Recursive Partial Least Squares (Joe Qin, 1998) and others are capable of both incremental and batch learning.

Streaming data learning algorithms can further be differentiated by their ability to use old data for learning, number of passes performed on the data, the ability to learn previously unseen classes

[3] In various works this is defined differently, the most common definitions are used here.

and the ability to retain prior learned knowledge. This chapter reviews works from different research directions, where the only requirement is the learning of information from new data.

2.3 Adaptation for predictive models on streaming data

In a static machine learning setting the parameters of the model are trained using a historical data set and then fixed with no modifications as the process continues regardless of the observations. This approach assumes that there are no significant changes in the process throughout the duration of the model's usage, or that those changes would not significantly degrade the performance of the model. However, this is not always the case. As noted in Chapter 1, machine learning is often applied to dynamic non-stationary environments where changes of various types are common. These changes often result in reduced predictive accuracy of the models or sometimes and more dramatically could even render the current model technically inapplicable to the new environment (this is discussed in more detail in next section).

Dealing with changes requires using the current data either for building a new model (and discarding the existing one) or adapting the existing one. Adapting a model involves reducing the effect of old and irrelevant data, and is often the preferred choice, especially when the current data may not be large enough to build a meaningful model or when the old knowledge needs to be preserved. It is possible to see that the concept of adaptation is closely related to the concept of learning new data, because it is precisely the new data that prompts adaptation. Learning new data can be to some extent considered to adapt a model, in a sense that with the increase in the amount of new data, the effect of old data is gradually reduced. However, simply learning new data after a change improves model's accuracy much slower than its explicit adaptation (Kuncheva, 2004a; Kolter & Maloof, 2007). It should also be noted that depending on the implementation, it is not always possible to simply learn data after certain types of changes in the data, e.g. after the change in input features.

2.3.1 Reasons for adaptation

The reason for the adaptation of predictive models is that, given the available data, a new model which performs significantly better than the existing one can be trained. This happens if:

- *The model is inapplicable to the new data.* This situation often occurs after the changes in the definition of data, for instance when number of features or their domains change. For example when a platform where the videos are rated using "thumbs up/down" ranking is switched to 5 stars ranking, the models trained to predict videos' ranking using the old system, would not be suitable for the new data, resulting in useless predictions. Depending on the algorithm, these type of changes might render the outcome meaningless, or prevent the model from making predictions whatsoever. It should be noted that certain changes of this kind may be addressed using pre- or post-processing techniques, for example treating removed features as missing data, however these type of solutions may not be adequate for the long term.

- *Alterations to the model would better explain the observations.* Other types of changes can lead to situations where the model delivers meaningful predictions, however its accuracy could be further improved. This is often manifested by the deterioration of the predictive accuracy over time. This can be more formally depicted as follows:

$$E[f(\boldsymbol{x}_{\tau_0})] < E[f(\boldsymbol{x}_{\tau_1})] \tag{2.5}$$

  where $E[f(\boldsymbol{x})]$ is the expected error of the predictive function, $\boldsymbol{x}_{\tau_0}, \boldsymbol{x}_{\tau_1}$ are the input data instances at times $\tau_0, \tau_1,\ \tau_1 > \tau_0$. In other words, the increase of the expected error over time means that the deterioration of the accuracy takes place. The types of changes in the real world causing this can be varied; these could be wear of sensors for process industry models, market perturbations in financial modelling, change of light settings in image recognition etc. In general, this is often a result of a change in input-output conditional probability of the data generation process, which is also known as concept drift.

The work in this thesis will be mainly focused on the second type of changes, however some of the discussed techniques are applicable to the first type as well.

2.3.1.1 Concept drift

One of the most researched and well defined reasons for the adaptive models is the problem of concept drift, as introduced in (Schlimmer & Granger, 1986b) or (Widmer & Kubat, 1996). Using the probabilistic formulation from (Gao et al., 2007), joint input-output probability is given by $\mathcal{P}(\boldsymbol{x}, y) = \mathcal{P}(y|\boldsymbol{x}) \cdot \mathcal{P}(\boldsymbol{x})$. Concept drift occurs when either the independent probability distribution $\mathcal{P}(\boldsymbol{x})$ or conditional probability distribution $\mathcal{P}(y|\boldsymbol{x})$ changes with time[4]. The former is often called virtual and the latter real concept drift. Many authors, e.g. (Tsymbal et al., 2008) or (Žliobaitė, 2009) note that for practical purposes, these two types can be considered the same in that they both require changes in models. In (Elwell & Polikar, 2011) it is stated that, virtual drift requires supplemental learning or refining of the model and real drift requires replacement learning or forgetting. In the terms of change speed and behaviour the following categories of drift are identified in the literature (Gama et al., 2014):

- *Sudden drift.* This is sudden change of concept at a certain time τ.
- *Gradual drift.* Occurs when one concept gradually replaces another over certain time period $\tau_2 - \tau_1$.
- *Incremental drift.* This is a slow change involving the passage through intermediate concepts between the concepts before the drift has started and the final concept after the drift is finished.

[4] It can be implicitly inferred that the combination $\mathcal{P}(\boldsymbol{x}), \mathcal{P}(y|\boldsymbol{x})$ is called a *concept*.

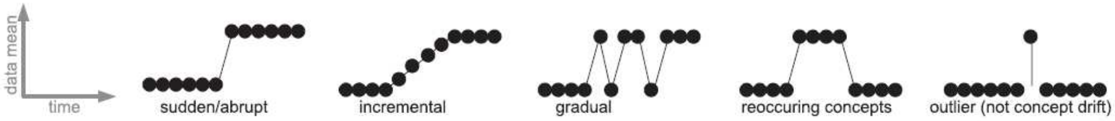


Figure 2.2: *Concept drift types (Gama et al., 2014).*

- *Reoccurring concepts*. Concepts are replaced by the new ones, only to reappear at the later stage. This is typical for cyclical and seasonal processes.

Figure 2.2 visualizes types of drift. Although the notion of concept drift appears more often in classification scenario, there are works which use this concept in regression setting as well (Ikonomovska et al., 2010; Gomes Soares & Araújo, 2015a,b). For a recent survey on concept drift, reader is referred to (Gama et al., 2014).

2.3.2 Adaptation for predictive modelling

As mentioned in the previous section, this thesis concentrates mainly on the accuracy of the predictive models. Therefore a following definition can be used: ***Definition*:** Adaptation is an update of combination of the model's *training data coverage*, *structure*, and *parameters* to optimize predictive accuracy in order to respond to the changes in the data.

These changes may include the ones listed above. Adaptation also learns new data, however the difference between simply learning new data and adaptation based on new data is that the goal of simple learning is *refining* the model, and the goal of adaptation is *changing* it. The methods for both types of learning can be similar. However, due to the temporal nature of adaptation, many adaptive learning methods assume that the data before the change is less relevant for the new model than the data after change. As a result of this assumption, these methods tend to give more weight to the newer data instances than to the older ones while constructing the models.

An important aspect of adaptive models is when the adaptation process is triggered. Some models use change detection routines for example described in Basseville & Nikiforov (1993); Gama et al. (2004); Baena-García et al. (2006) to attempt to detect when the model needs adaptation. Stationarity tests such as tests pioneering Priestley-Subba-Rao test (Priestley & Subba Rao, 1969) or more recent approaches described in (von Sachs & Neumann, 2000; Nason, 2013; Balaguer-Ballester et al., 2014) may be also applied for this purpose. If the change is correctly detected, these approaches avoid needless adaptation, therefore allowing for more robust models. However, as change detection is no trivial task, other approaches prefer to adapt the models in regular time intervals without additional triggers (Stanley, 2002; Elwell & Polikar, 2011; Scholz & Klinkenberg, 2007; Jacobs et al., 2010). This ensures a guaranteed response to the changes, however risks the overreaction to noise on one hand, and delayed adaptation on another hand, subject to the parameter choice. In classification scenario, triggering adaptation after every misclassification is also practised (Kolter & Maloof, 2005, 2007).

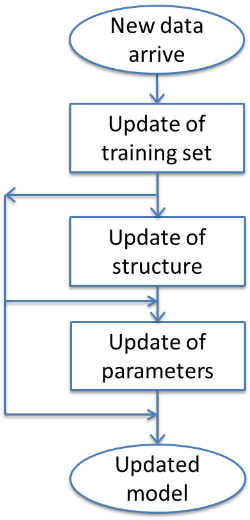


Figure 2.3: *General adaptations scheme.*

There are various adaptive predictive algorithms in the machine learning literature. They use adaptive mechanisms which can be categorized into several types. The next sections present a detailed overview of these adaptive mechanisms.

2.4 Overview of adaptive mechanism types

In this section a hierarchical structure of adaptive mechanisms types is presented. More concretely, on the lowest tier of the hierarchy the object of adaptation is the underlying training set of the model, and on the higher tiers its parameters and structure. This hierarchy is depicted on Figure 2.3. Here, the hierarchy is meant in a sense that the application of an adaptive mechanism of the higher level, requires the application of the adaptive mechanism of lower level. It should be noted that, depending on the adaptation settings, the effects of adaptive mechanisms on each level can be differently strong. Evolutionary adaptation approaches which construct a special niche are also discussed in Section 2.4.4.

2.4.1 Adapting training data coverage

The basic adaptive mechanism is updating the coverage of models' training data set to give more weight to recent data. The most commonly used techniques for this purpose are moving window and decay factors approaches as discussed in Sections 2.4.1.1 and 2.4.1.2. For some methods which do not build an explicit general model for the prediction of new data instances, also known as *lazy* learning methods, such as k-nearest neighbours (Hastie et al., 2009), just an updated dataset will instantly change the output of the model. For the methods which do construct a general prediction model, also called *eager* learning methods (e.g. linear regression, SVM, decision trees), new data prompts the update of the model's parameters or structure, which is discussed in the sections below.

Deciding which subset of data to use as training set is one of the paramount questions that needs to be answered while developing an adaptive predictive algorithm. Generally it is assumed that, in the case of no changes in the process, the more data is included in the training set, the lower the variance of the parameter estimates will be, thus increasing the predictive accuracy. It has been shown that the size of the training data sufficient to achieve the desired accuracy is determined by the VC dimension[5] of the algorithm (Blumer et al., 1989; Anthony & Biggs, 1997; Vidyasagar, 1997). More practically, it is determined by the total number of free parameters and permitted error (Haykin, 2009).

In non-stationary data setting, it becomes important to not only select a training set of sufficient size but also one which is relevant to the current data. The parameters of the adaptive approaches determine the scale of reducing the influence of the old data. Less influence leads to faster adaptation, however, it also increases the leverage of noise or structural changes in the recent past. Conversely, increasing the influence of older data reduces the rate of adaptation, but at the same time creates more robust models. This is called the *stability-plasticity* dilemma (Carpenter & Grossberg, 1987) in adaptive learning. Sometimes data which is considered irrelevant is simply discarded. This is called “forgetting” (Kuh & Petsche, 1990) or “catastrophic forgetting” (Polikar et al., 2001) and can be detrimental if the discarded data becomes relevant in the future. Popular approaches to choose the training data for the model are moving window and decay factors. It should be noted that not all of the approaches require to explicitly store the old data as online learning methods which are often used for adaptive models implicitly update their training sets when the new data becomes available.

2.4.1.1 Moving window approaches

The moving window approaches limit the training data for predictive model at time τ to the most recently seen l instances as $\boldsymbol{\mathcal{V}} = \{(\boldsymbol{x}_{\tau-l}, y_{\tau-l}), \cdots, (\boldsymbol{x}_{\tau}, y_{\tau})\}$ where l is the size of the window. The advantages of this approach are its simplicity and effectiveness in many cases (Widmer & Kubat, 1996). The window size is the only parameter, and is critical in controlling how fast the adaptation is performed. The speed of adaptation is inversely proportional to the size of the window. However insufficient training data might result in more inaccurate models, and overreacting to noise. While l is usually fixed, works such as (Widmer & Kubat, 1996; Klinkenberg, 2004) propose heuristics for an adaptive window size which proves to be more beneficial, particularly when dealing with random changes. (Widmer & Kubat, 1996) propose to increase the window when the input data is stable and decrease it when concept drift occurs. (Klinkenberg, 2004) suggests testing several sizes of windows on the latest data batch and choosing the one that performs the best. In many scenarios, for example if a single model is created from the sliding window, it is beneficial to set window size so that its contents exhibit stationary behaviour. Stationarity tests

[5] (Vapnik & Chervonenkis, 1971)

mentioned in Section 2.3.2, (Priestley & Subba Rao, 1969; von Sachs & Neumann, 2000; Nason, 2013) may be used for this purpose. The moving window approaches are usually based on the time of data instances' arrival, however the similarity of training data to the current data can also be considered for this purpose.

2.4.1.2 Decay

One could view a window approach as weighting vector of ones applied to the most relevant data, and zeros to the rest of data. In a more general case, a continuous decreasing weight could be applied. The simplest approach is to use a single decay factor $\lambda < 1$. The repeated use of this decay factor leads to the exponential reduction of data's weight. Decay can be based not only on time of instances' arrival (Joe Qin, 1998; Klinkenberg & Joachims, 2000), but also on similarity to the current data (Tsymbal et al., 2008), combination thereof (Žliobaitė, 2011), density of the input data region (Salganicoff, 1993b) or consistency with new concepts (Salganicoff, 1993a). Non-exponential decay approaches also exist, for example autoregressive model (Mills, 1991).

2.4.2 Adaptation of predictive models' structure

Structure of predictive model is the set of its components and the way these components are connected to each other. Some model types actually define a family of models with different possible structures. For example, a 2-layer feed-forward neural network may have different number of nodes. Adding or removing a node does not change the nature of the model in this case. Structure of the predictive model could be for example, hierarchical (for example decision trees) or graph-like (Bayesian or neural networks). Here, the structure is not necessarily limited to the topological context – number of rules in rule based systems or number of experts in an ensemble could be considered part of the model's structure. Updates in the models' structure are also used for the adaptation purposes, often changing the model more radically than update of the parameters would be able to accomplish. Relevant examples are listed below:

- **Decision trees.** There is a considerable amount of research dedicated to decision trees with updatable structure. For instance, Very Fast Decision Trees (VFDT) for classification by (Domingos & Hulten, 2000) incrementally grows the tree with the arrival of new observations and its extension Concept-adapting Very Fast Decision Tree (CVFDT) (Hulten et al., 2001) uses a sliding window approach to deal with concept drift. The regression-capable online decision tree introduced in (Basak, 2006) is also capable of altering the tree structure, albeit with a fixed tree depth.
- **Model trees**. Structural updates are also applied for model trees. For instance, (Potts & Sammut, 2005) propose an incremental algorithm with the strategy of splitting the node if it is statistically unlikely that the examples in it are generated by a linear model. A method proposed in (Ikonomovska et al., 2010) can replace a subtree with a leaf or grow an alternative subtree if a change is detected.

- **Neural networks.** Although most of the existing neural network architectures use a fixed structure once trained, some developments indirectly include the ability of changing networks' structure after retraining by using variable node *dropout* probabilities (Ba & Frey, 2013). Another neural-network based approach used for online learning, and capable of adapting its structure, is the ARTMAP family of methods (Carpenter et al., 1991; Vakil-Baghmisheh & Pavešić, 2003). It adds nodes to accommodate for new classes which are encountered during the prediction process.
- **Bayesian networks.** The Bayesian network as proposed in (Pearl, 1988) is an approach which aims to capture causal relationships between features and model them as a direct acyclic graph using Bayes's theorem. Updating Bayesian networks' structure is described in research such as (Friedman & Goldszmidt, 1997; Lam, 1998; Alcobé, 2004). (Castillo & Gama, 2006) propose a multi level adaptation scheme for a Bayesian network, which involves both changing its parameters and structure.
- **Ensemble methods.** Adding or removing experts (e.g. in (Stanley, 2002; Kolter & Maloof, 2007; Hazan & Seshadhri, 2009; Gomes Soares & Araújo, 2015b)) is an important adaptation mechanism for adaptive ensemble methods, and can be considered a structural change. This is discussed in greater detail in the Section 2.5.3.

.

2.4.3 Adaptation of predictive models' parameters

Most of the predictive models' predictions are dependant on a number of parameters (see Equation 2.2), which may be adjusted to adapt the model. Training the model in this case comes down to estimating these parameters and adapting it involves updating the parameters as new data instances are available. Many popular algorithms have online versions to be able to update their parameters faster than retraining them from scratch and to not require the historical data. Following examples can be considered:

- **Linear regression.** For linear regression the set of parameters are the linear coefficients, which can be estimated using different methods, such as Ordinary Least Squares Estimation. There are many types of linear models where the coefficients are allowed to change with time. An example of these is adaptive version of Least Squares Estimation described in (Jang et al., 1997). It recursively updates the parameters, optionally employing the decaying of the weights of old observations.
- **Naive Bayes classifier.** The parameters of Naive Bayes classifier (described in more details in Section 2.5.1.1) are the prior probabilities of classes and conditional probabilities of features given classes. These are calculated during the training of the model and could be incrementally updated during the prediction process by saving sums and counts of features, and number of observations.
- **Neural Networks.** The weights of neural network can be considered its parameters. They

can be updated with each learned instance using for example back-propagation algorithm (Werbos, 1974). As back-propagation does not provide explicit control over the adaptation, neural networks which use it or similar training methods can be prone to catastrophic forgetting (Kasabov, 2001). Several approaches address this problem. For instance, *experience replay* (Lin, 1992) stores a set of old instances in memory and uses them along with the new ones to train the network. This approach has been successfully used recently in Deep Q-Network (DQN) algorithm (Mnih et al., 2015). Another solution of forgetting problem, which has become very popular in recent years is Long Short-term Memory (LSTM) architecture introduced in (Hochreiter & Schmidhuber, 1997) and recently analysed in (Greff et al., 2016). LSTM deals with this issue using special blocks which assist the network to control the forgetting.

- **Ensemble methods.** Expert weights are an important parameter of the ensemble methods. They are often recalculated using the new data or updated in various ways (Littlestone & Warmuth, 1994; Kolter & Maloof, 2007; Elwell & Polikar, 2011; Kadlec & Gabrys, 2011; Bakirov et al., 2017). This is discussed in more detail in the Section 2.5.2.

2.4.4 Evolutionary approaches

A family of methods collectively called evolutionary approaches practice adaptation which is loosely based on the theory of evolution. These approaches involve having a population of solutions, adding new members using certain operations like mutation and crossover, and selecting the “fittest” members from the population based on some fitness criteria. One group of evolutionary methods which are able to operate on streaming data are Learning Classifier Systems (LCS), introduced in (Holland, 1976). LCS work with a pool of rules where new rules are created if no existing rules are relevant for new examples, and also by constant application of genetic operators and selection of the best rules using a fitness function. The approach combines both exploration by selecting a random passing rule to determine the action (or label) and exploitation by selecting an action which has maximum combined fitness among all the passing rules. One example of LCS for classification is sUpervised Classifier System (UCS) (Bernadó-Mansilla & Garrell-Guiu, 2003). Further example are evolutionary mechanisms and algorithms applied in the context of multi-component multi-level predictive systems (Gabrys & Ruta, 2006; Tsakonas & Gabrys, 2012, 2013; Lemke et al., 2013).

Another interesting area of population based techniques are Artificial Immune Systems (AIS) (Castro & Zuben, 1999) which simulate the human immune system. Simplistically described, AIS create and maintain a pool of antibodies for recognising the threats to the organism via genetic operations and try to increase diversity of this pool for better recognition abilities. Antibodies gradually age and are replaced with new ones. Ageing speed is inversely proportional to their recognition rates, which means that antibodies that are less effective are replaced faster and the ones that are more effective are replaced slower. Some characteristics of AIS, such as adaptation, forgetting and ability to deal with new threats make it potentially very useful for non-stationary streaming data. For example (Abi-Haidar & Rocha, 2008) apply AIS methods to spam detection and (Haag et al., 2007) use them for network intrusion detection. A survey of AIS research can be found in (Dasgupta et al., 2011).

2.5 Ensemble methods

A popular approach in machine learning is combining multiple models (which in this scenario are often called "predictors", "learners" or "experts"[6]) for predictions (Ruta & Gabrys, 2000). Ensemble methods were originally developed in 1960-s for static data (Bates & Granger, 1969), but have gained widespread use for streaming and non-stationary data as well. In the static setting the ensembles are widely used for "boosting", which aims to increase the prediction accuracy for parts of the input space where accuracy is low (Freund & Schapire, 1995), or "bagging", which aims to increase the generalising ability of a predictive model (Breiman, 1996). (Dietterich, 2000) presents three reasons why ensembles of experts often perform better than single learners in stationary settings:

- Statistical: many models can perform quite similar on training data but differently on the test data. To eliminate the risk of choosing a poor model, a safer choice would be using them all in the ensemble and average their predictions.
- Computational: many experts reduce the risk of getting stuck in local optima during the learning process.
- Representational: in many situations the concept might be represented better by the weighted sum of multiple learners rather than a single learner.

The drawback of combining multiple learners is that the resulting model will have higher complexity than using a single learner (Kuncheva, 2004b). Moreover, it has been also shown that it can be very beneficial to organise predictions in multiple layers (Ruta & Gabrys, 2002, 2005) (similarly to the recently popularised deep neural network structures) leading to even further increase in the of the complexity of the resulting system together with a great potential performance gain. This has been illustrated in many sucessful experimental studies carried out in author's group (Ruta & Gabrys, 2005, 2007; Riedel & Gabrys, 2007; Lemke et al., 2009; Budka et al., 2010; Ruta et al., 2011; Tsakonas & Gabrys, 2012, 2013; Lemke et al., 2013; Martín Salvador et al., 2016b).

In a streaming data setting, experts are used to improve the generalisation properties of the model and to serve as a representation for historical data. In addition, in a non-stationary setting, some models attempt to align experts with concepts as they arise (Kadlec & Gabrys, 2011; Shao & Tian, 2015; Jin et al., 2015a).[7] Many studies (e.g. (Kolter & Maloof, 2007; Elwell & Polikar, 2011; Kadlec & Gabrys, 2011)) have found that adapting ensemble combination weights, for

[6] These terms are used interchangeably throughout this thesis.

[7] Clear partition between concepts is often not attainable.

instance based on experts' prediction accuracy, is an effective method of dealing with changes in the data. Often the combination weights of ensemble members are updated at a rate inversely proportional to their prediction errors. More concretely, $w_1 = \nu(\epsilon_1, w_0)$, where w_1 is the weight after the update, and $\nu(\cdot)$ is the update function, its inputs being error ϵ_1 and optionally previous weight w_0. For instance, the update function can be realised as $\nu(\epsilon_1, w_0) = w_0 e^{-\epsilon}$ (Herbster & Warmuth, 1998). When the input w_0 is absent, the weight update function takes the form of $w_1 = \nu(\epsilon_1)$ and essentially becomes weights recalculation, used for instance in (Elwell & Polikar, 2011). Experts may be separately updated (Kolter & Maloof, 2007) or adapted using moving windows or forgetting factors (Kadlec & Gabrys, 2011; Bakirov et al., 2015). Creation of new experts from newly arrived observations is another useful strategy to learn new information which has been found to provide good results (Klinkenberg, 2004; Wang et al., 2006; Kolter & Maloof, 2007; Elwell & Polikar, 2011). An alternative to combination is *gating*, where a prediction of a single expert, which is considered the most relevant to the current state of the system, is selected as a final prediction of the ensemble.

Adaptive ensemble methods for learning on streaming data often feature all of the levels of AM hierarchy introduced in Section 2.4. Adapting individual expert may be realised via the training data coverage adaptation and the parameter adaptation levels. Adapting the combination weights may be considered a form of parameter adaptation and adding/removing experts an adaptation of model's structure. In Sections 2.5.1, 2.5.2 and 2.5.3 a short overview of the most prominent adaptation methods and techniques belonging to each of these levels is presented.

2.5.1 Base learners and their adaptation

There are many choices for the type of models that can be used in an ensemble as the base learners, in fact any predictive model, including other ensembles, can be used for this purpose. For instance, machine learning methods such as linear regression (Kadlec & Gabrys, 2011; Gomes Soares & Araújo, 2015b), naive Bayes (Bakirov & Gabrys, 2013), SVM (Scholz & Klinkenberg, 2007), decision trees (Street & Kim, 2001; Stanley, 2002; Bifet et al., 2009), neural networks (Polikar et al., 2001; Lan et al., 2009; Gomes Soares & Araújo, 2015a), k-nearest neighbours (Masud et al., 2013) have been used as base learners. Base learners may be updated by explicitly or implicitly increasing their training data and may be adapted using adaptation mechanisms such as moving windows or decay factors (Sections 2.4.1.1 and 2.4.1.2). Examples of algorithms which employ learnable/updatable base learners are (Stanley, 2003; Kolter & Maloof, 2007; Bifet & Gavaldà, 2007; Kadlec & Gabrys, 2011; Grbić et al., 2013; Gomes Soares & Araújo, 2015a,b; Shao & Tian, 2015; Jin et al., 2015b; Bakirov et al., 2017). However, it is not always the case that experts are adapted, as the adaptation can be performed purely at the combination level and/or via adding/removing experts. This is characteristic only to batch learning algorithms (e.g. (Scholz & Klinkenberg, 2007; Elwell & Polikar, 2011)).

Sections 2.5.1.1 and 2.5.1.2 reviews Naive Bayes classifier and Recursive Partial Least Squares

(RPLS) which were used as base learners for experiments in this thesis.

2.5.1.1 Bayesian learning and Naive Bayes classifier

Bayesian learning is an inherently probabilistic approach for building predictive models in which unknown quantities are given a distribution based on observations. The distributions are updated and predictions are formed from the posterior. The core of Bayesian learning is Bayes' theorem $\mathcal{P}(A|B) = \frac{\mathcal{P}(B|A)P(A)}{\mathcal{P}(B)}$ which relates the of conditional probability of $\mathcal{P}(A|B)$ (event A given B) to the $P(B|A)$ (likelihood of B given A; typically the observation given the model) and independent *a priori* probabilities of two events $P(A)$ and $P(B)$. In the classification setting this equation can be written as:

$$\mathcal{P}(c|\boldsymbol{x}) = \frac{\mathcal{P}(\boldsymbol{x}|c)\mathcal{P}(c)}{\mathcal{P}(\boldsymbol{x})}, \tag{2.6}$$

where c is a class label. The popular Bayesian learning method, Naive Bayes classifier, assumes the conditional independence of features from each other. Then, the classification problem comes down to estimating the probability of each class and determining the class with the highest probability value, as:

$$\mathcal{P}(c|\boldsymbol{x}) = \frac{1}{Z}\mathcal{P}(c)\Pi_{m=1}^{M}\mathcal{P}(x_i|c) \tag{2.7}$$

Here Z is a product of probabilities of all inputs $\mathcal{P}(x_1), \mathcal{P}(x_2), ..., \mathcal{P}(x_M)$ which is constant for all class labels. Returning to the described hierarchy, independent probabilities of classes/features and conditional probabilities $\mathcal{P}(x_m|c),\ m \in 1 \cdots M$ may be considered as the parameters of Naive Bayes classifier. It is possible to incrementally update these values when new instances are observed. Despite the strong assumption of feature independence, the Naive Bayes classifier is widely used in practice, because of its usually satisfactory prediction accuracy, ease of implementation and inherent capability of online learning.

2.5.1.2 Recursive Partial Least Squares

Recursive Partial Least Squares is an extension of Partial Least Squares (Wold, 1966), both being popular in chemical process modelling. PLS projects the scaled and mean centered multidimensional input data $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ and output data $\boldsymbol{Y} \in \mathbb{R}^{N \times K}$, where N is the number of data instances, M is the number of input variables and K is the number of output variables, to separate latent variables,

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{D}^\top + \boldsymbol{V} \tag{2.8}$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}^\top + \boldsymbol{F}. \tag{2.9}$$

Here $\boldsymbol{T} \in \mathbb{R}^{N \times L}$ ($L \leq M$ is the number of latent variables) and $\boldsymbol{U} \in \mathbb{R}^{N \times L}$ are the score matrices, $\boldsymbol{D} \in \mathbb{R}^{M \times L}$ and $\boldsymbol{Q} \in \mathbb{R}^{K \times L}$ are the corresponding loading matrices, and $\boldsymbol{V}$ and $\boldsymbol{F}$ are the input and output data residuals. Then the score matrices $\boldsymbol{T}$ and $\boldsymbol{U}$ consist of so called latent vectors:

$$\boldsymbol{T} = [\boldsymbol{t}_1, ..., \boldsymbol{t}_L], \text{ where } \boldsymbol{t}_l \in \mathbb{R}^{N\times 1} \text{ for } 1 < l < L \tag{2.10}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_L], \text{ where } \boldsymbol{u}_l \in \mathbb{R}^{N\times 1} \text{ for } 1 < l < L. \tag{2.11}$$

where the column vectors $\boldsymbol{d} \in \mathbb{R}^{M\times 1}$ and $\boldsymbol{q} \in \mathbb{R}^{M\times 1}$ of the loading matrices $\boldsymbol{D}$ and $\boldsymbol{Q}$ represent the contributions of the input and output variables to the mutually orthonormal latent vectors $\boldsymbol{t}$ and $\boldsymbol{u}$, respectively. Equations 2.8 and 2.9 constitute the PLS outer model. Afterwards a regression model, which is also called the PLS inner model, between the latent scores is constructed:

$$\boldsymbol{U} = \boldsymbol{T}\boldsymbol{B} + \boldsymbol{R}, \tag{2.12}$$

where $\boldsymbol{B} \in \mathbb{R}^{L\times L}$ is a diagonal matrix of regression weights which minimizes the regression residuals $\boldsymbol{R}$. Then the estimates $\tilde{\boldsymbol{Y}}$ of $\boldsymbol{Y}$ are:

$$\tilde{\boldsymbol{Y}} = \boldsymbol{T}\boldsymbol{B}\boldsymbol{Q}^\top, \tag{2.13}$$

There are different methods to calculate the required vectors $\boldsymbol{t}$, $\boldsymbol{d}$, $\boldsymbol{u}$, $\boldsymbol{q}$ and $\boldsymbol{b}$. One of the most popular ones, Nonlinear Iterative Partial Least Squares (NIPALS) (Geladi & Kowalski, 1986), updates latent vectors in an iterative way. After each iteration, the explained covariance is removed from the data:

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i - \boldsymbol{t}_i\boldsymbol{d}_i^\top \tag{2.14}$$

$$\boldsymbol{Y}_{i+1} = \boldsymbol{Y}_i - \boldsymbol{u}_i\boldsymbol{q}_i^\top. \tag{2.15}$$

The subsequent $(i+1)$-th vectors are calculated by the resulting new input and output data $\boldsymbol{X}_{i+1}$ and $\boldsymbol{Y}_{i+1}$. Recursive PLS, which uses NIPALS, updates the matrices $\boldsymbol{D}$, $\boldsymbol{T}$, $\boldsymbol{Q}$, $\boldsymbol{U}$ and $\boldsymbol{B}$ when the new data becomes available, on either sample-by-sample (incremental) or batch basis. The batch adaptation used in this thesis works by applying PLS on the new batch and constructs new input and output matrices as follows:

$$\boldsymbol{X}_{new} = \begin{bmatrix} \lambda\boldsymbol{D}_0^\top \\ \boldsymbol{D}_1^\top \end{bmatrix} \tag{2.16}$$

$$Y_{new} = \begin{bmatrix} \lambda\boldsymbol{B}_0\boldsymbol{Q}_0^\top \\ \boldsymbol{B}_1\boldsymbol{Q}_1^\top \end{bmatrix}, \tag{2.17}$$

where the matrices $\boldsymbol{D}_0$, $\boldsymbol{B}_0$ and $\boldsymbol{Q}_0$ describe the old model and $\boldsymbol{D}_1$, $\boldsymbol{B}_1$ and $\boldsymbol{Q}_1$ the new one created from the most recent batch. $0 \leq \lambda \leq 1$ is the forgetting factor which determines how much influence the historic data will have, with $\lambda = 0$ meaning zero influence and $\lambda = 1$ meaning that the historical data has the same influence as the new batch. After constructing the new input and output data matrices, PLS is applied on them to get the updated matrices. The condition for this update is that the number of latent variables must be equal to the rank of $\boldsymbol{X}$. This condition can be practically met by finding a number of latent variables for which the error on the training data is less than a threshold value defined close to 0.

2.5.2 Combinational adaptation methods

A prominent adaptation method for ensemble models is changing the combination weights of experts. Techniques belonging to this category are described in this section.

Consider a set of I experts $S = \{s_i, ..., s_I\}$ which produce predictions $\hat{\boldsymbol{y}} = \{\hat{y}_1, ..., \hat{y}_I\}$ where $\hat{y}_i = s_i(\boldsymbol{x})$ with input $\boldsymbol{x}$. These may be labels or numeric values depending on the problem. In case of classification, there exists a set of all possible labels $C = \{c_1, ..., c_J\}$. Then for all $i = 1 \cdots I$ and $j = 1 \cdots J$ the matrix A with following elements can be calculated:

$$a_{i,j} = \begin{cases} 1 & \text{if } s_i(\boldsymbol{x}) = c_j \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

Assuming weights vector $\boldsymbol{w} = \{w_1, ..., w_I\}$ for respective predictors in S, the sum of the weights of predictors which voted for label c_j is $z_j = \sum\limits_{i=1}^{I} w_i a_{i,j}$. The final prediction is:[8]

$$\hat{y} = \underset{c_j}{\operatorname{argmax}}(z_j). \tag{2.19}$$

For the most common case of averaging the experts' predictions for regression, the final prediction is simply

$$\hat{y} = \frac{\sum\limits_{i=1}^{I} w_i \hat{y}_i}{\sum\limits_{i=1}^{I} w_i} \tag{2.20}$$

Un-weighted combination is a special case of above equations where the weights are assumed to be equal; $w_1 = w_2 = \cdots = w_I$.

There are two types of experts' combination; fusion (also called competition) and selection (also called gating or cooperation) (Kuncheva, 2004b). The former involves combining the experts using *global*[9] weights which often depend on their *predictive performance* (in this thesis this will be called *global weighting*), the latter selects a single expert often based on *input data*. Often these approaches are mixed, typically considering input data and sometimes outputs for the calculation of weights. The mixed approach will be called *local weighting* in this thesis.

Global weighting is used for adaptive ensembles more commonly than local weighting (Kuncheva, 2004b). Examples of methods that employ global weighting include (Littlestone &

[8] This definition is adapted from (Kuncheva, 2004b).

[9] Here, the term "global" means that the weights are not influenced by input or output data.

Warmuth, 1994; Stanley, 2002; Kolter & Maloof, 2005, 2007) for classification and (Kaneko & Funatsu, 2014, 2015; Gomes Soares & Araújo, 2015b,a) for regression. For incremental learning setting, the weight change is usually implemented via multiplication by $0 < \beta < 1$, $w_{\tau+1} = \beta w_{\tau}$, where w_{τ} is the weight of an expert at time τ and $w_{\tau+1}$ is the updated weight at time $\tau + 1$ (Littlestone & Warmuth, 1994; Kolter & Maloof, 2007). For classification tasks, this is usually performed only when the prediction of the expert is wrong. For regression task, β is usually a function of expert's error (Vovk, 1990; Kolter & Maloof, 2005). These type of algorithms often normalize the weights after reweighing (e.g. (Kolter & Maloof, 2007)). In batch learning scenario, recalculation of weights based on experts' performance is often used (Wang et al., 2003; Elwell & Polikar, 2011; Bakirov et al., 2015). Other aspects such as *lift*, which is a measure of correlation between a specific prediction and a specified true label (Scholz & Klinkenberg, 2007) or expert creation time, i.e. expert's age (Elwell & Polikar, 2011) could be taken into account during the weighting process as well.

Adaptive local weighing is less common in the literature, despite the intuition that together with well trained experts, it is probably the better choice (Kuncheva, 2004b). The difficulty of theoretical analysis may contribute to this fact. One of the first approaches to use adaptive local weighting was (Jacobs et al., 1991). They use a feed-forward gating network which outputs the selection probabilities for each local expert, based on the input data. Local ensemble learning has been applied to regression in (Kadlec & Gabrys, 2010, 2011; Tsakonas & Gabrys, 2012, 2013; Grbić et al., 2013; Shao & Tian, 2015; Jin et al., 2015b; Al-Jubouri & Gabrys, 2016).

Expert selection, used in (Jacobs et al., 1991) can be considered a special case of weighting when all of the weights except for one are 0. A dedicated method for dealing with concept changes in incremental settings using this kind of approach via model trees with dynamic structure is proposed in (Ikonomovska et al., 2010). The use of meta-features to identify to select an expert from the pool is described in (Alippi et al., 2012).

2.5.3 Adaptation via adding or removing predictors

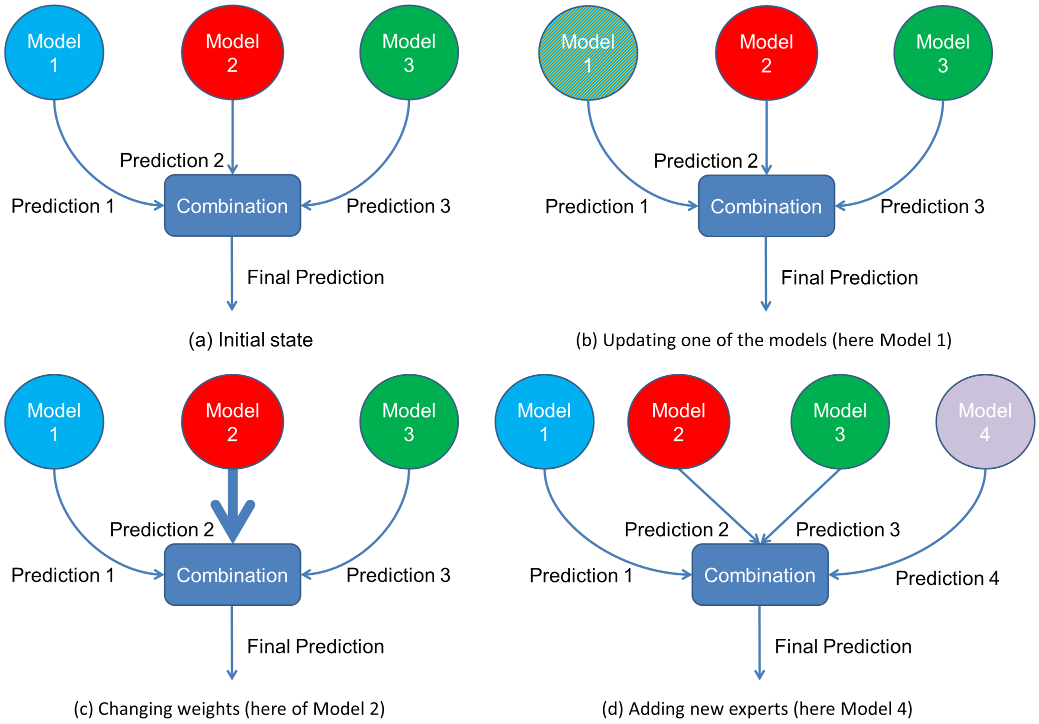
The next hierarchy level of adaptation in ensembles is changing the ensemble structure, which is realised by adding predictors to the ensemble or removing them. Similar to the single learners' adaptation, adding/removing experts can refine the existing model, add new information to the model and forget old data. New experts are usually trained on the latest data and so are likely to be better suited to predict on it than the existing ones. Adding new experts to the mix brings additional flexibility and adaptive power to ensemble methods, and is especially important when other adaptive mechanisms such as adapting the experts or combination do not provide satisfactory results. It can be done either using a defined trigger mechanism or on a regular basis without additional trigger.

In triggered expert creation, methods often attempt to create a new expert each time a specific condition is satisfied. Most of the times this indicates that there is a potential change occurring in the data. The change indicators are often based on error of the model. For instance in classification tasks, in the simplest case an expert may be added after each misclassification (Kolter & Maloof, 2005, 2007). Combined with the reduction of votes of under performing experts, this method works often well. However, as observed in (Kolter & Maloof, 2007) adding a new expert after every misclassification can lead to the unnecessarily high amount of predictors, particularly in noisy domains. To alleviate this problem authors propose adding an expert only if every Ω^{th} data instance is misclassified. Another strategy is adding an expert on the basis of ensemble's performance on a window of last observed data, as proposed in (Bouchachia & Balaguer-Ballester, 2014). To explore other expert creation strategies for this scenario, several alternatives were implemented and discussed (Bakirov & Gabrys, 2013) (refer also to Appendix A). For the regression case (Kolter & Maloof, 2005) propose adding a new expert if $|\hat{y}_t - y_t| > \zeta$, where $\hat{y}$ is predicted value, y real value and ζ predefined threshold. (Gomes Soares & Araújo, 2015b) add an expert every time the ratio of prediction error of the ensemble on the current data instance to the true value is higher than a certain threshold. Other methods use more complicated change detection mechanisms to trigger creation of new experts. These could be also based on error measures such as in (Gama et al., 2004; Baena-García et al., 2006) or based on input data as in (Bifet et al., 2009; Raza et al., 2015). (Alippi et al., 2012) use two change detection mechanisms to trigger the adaptation, and adds an expert if no experts in the ensemble have been trained on the new concept.

Perhaps the easiest method to add new predictors is to create them at fixed intervals irrespective of the model's performance and without using any other trigger. This method is widely used in batch learning scenario, where data arrives in batches and each time a new predictor is trained from the latest batch (Scholz & Klinkenberg, 2007; Elwell & Polikar, 2011; Gomes Soares & Araújo, 2015a). Adding an expert at fixed intervals is practised in incremental learning as well. For instance (Stanley, 2002) and (Hazan & Seshadhri, 2009) add a new predictor every instance and (Jacobs et al., 2010) propose adding new predictor every Ω data instances. (Kaneko & Funatsu, 2014) add a new expert when the model trained on new data is not a duplicate of an existing predictor in the ensemble. (Kaneko et al., 2014) use the most relevant instances to the current data for so called *lazy* or *just-in-time* learning, in other words for predicting on the fly using the similar data instances.

Removal of experts is often performed when their performance is unsatisfactory (Kolter & Maloof, 2007; Gomes Soares & Araújo, 2015b), when change is detected (Bifet et al., 2009), when a new expert performs better and replaces the existing one (Street & Kim, 2001), or based on their age (Hazan & Seshadhri, 2009). Usually, because of weights update, poor performing experts end up having low combination weights, so in this case their removal has less impact on the model than the addition of new ones.

Figure 2.4 shows the general scheme of ensemble methods and their adaptation mechanisms described in the sections above. Table 2.1 summarizes these mechanisms.



(a) Initial state

(b) Updating one of the models (here Model 1)

(c) Changing weights (here of Model 2)

(d) Adding new experts (here Model 4)

Figure 2.4: *Ensembles and their adaptation mechanisms.*

Object of adapation	**Type of adaptation**	**Useful for**
Base learners update	Training data Parameters Structure	Learning new data Adaptation with forgetting
Combinational adaptation	Parameters	Switching between experts Reweighing experts
Add/remove experts	Structure	Learning new data Adaptation with forgetting Adaptation without forgetting

Table 2.1: *Summary of ensemble adaptation methods.*

2.5.4 Dynamic Weighted Majority

An example of an ensemble which features all three adaptation types is the Dynamic Weighted Majority (DWM) algorithm (Kolter & Maloof, 2007). DWM adapts to drift by creating a new expert each time an instance is misclassified by the existing ensemble. New expert gets the weight of one. All experts are updated and whenever an expert misclassifies a data instance, its weight is scaled by $0 < \beta < 1$. After each classification, to reduce the dominance of newly added experts, the weights of existing experts are normalized, so that the expert with the largest weight is assigned a weight of 1. To reduce the number of experts, those experts whose weight is less than a defined threshold, η, are deleted. In (Kolter & Maloof, 2005) the same authors present

AddExp.D, a variation of this method where the weight assigned to the new experts is the current weight of the ensemble multiplied by a constant γ. Here the authors bound the error of this type of ensemble on the error of the latest created expert, provided that $\beta + 2\gamma < 1$. The same paper also introduces AddExp.C algorithm for regression based on the same principle. These algorithms show good performance when tested on several synthetic and real datasets. However, a known issue is their tendency to create large number of experts, particularly in noisy conditions. Also the choices regarding the adaptation mechanisms, such as choosing new weights for the experts, the value of β, are largely arbitrary, especially for DWM. DWM pseudocode is shown in Algorithm 1.

Some of the issues regarding the expert adding criteria and their training data have been addressed in Appendix A and in (Bakirov & Gabrys, 2013). Several modifications of the original DWM algorithm, focusing on creating new experts using larger training data sets, testing their performances before deciding whether to add them to the ensemble, as well as employing dynamic expert addition accuracy threshold values were proposed. Empirical tests were conducted on 26 synthetic two-dimensional datasets with various types of changes, as well as two real world datasets from the electricity consumption domain. Different settings of expert training and their weight adjustment were tested, along with comparing them to several popular change detectors and an alternative adaptive algorithm.

Accuracy-wise, there was no single method which performs best in all of the conditions. On average, MATure EXperts (MATEX) approach (allowing the experts to train on additional incoming data after their creation, without considering their outputs for a certain time) performs the best among synthetic datasets and well for real data as well. Performance of the original DWM is found to be inferior to that of its modifications in many cases for synthetic data, however it has shown good results on real data. There are also no single best algorithm settings, although certain setting choices often show considerably inferior accuracy rates.

2.6 Predictive models with multiple adaptive mechanisms

As seen from previous sections, learning on streaming data presents additional challenges to learning from static data. Even more challenging is building models which require adaptation over time. Perhaps the issue which requires the most attention is the stability-plasticity dilemma. Adaptive algorithms must find a balance between adapting to changes in data but at the same time being robust to noise. Ideally, algorithms must adapt to the true changes in the data, and ignore noise. Generally, this can be done at the cost of slowing the adaptation down, and adapting the model once the change has been confirmed. This is different from static learning, where it is desired to learn the concept as fast as possible.

Normally learning on non-stationary streaming data consists of two components: refinement and adaptation. The refinement component does not assume changes in data distribution and seeks to *improve* the learned concept by learning the new examples. In the literature, mainly the

Algorithm 1 The DWM algorithm.

```
Dynamic_Weighted_Majority ($\{\boldsymbol{X}, \boldsymbol{y}\}, K, \beta, \eta, \Omega$)
$\{\boldsymbol{X}, \boldsymbol{y}\} = \{(\boldsymbol{x_1}, y_1), \cdots, (\boldsymbol{x_N}, y_N)\}$: $N$ instances of training data
$\eta$: threshold for deleting experts
$\Omega$: period between expert removal, creation, and weight update
$S = \{s_1, \cdots, s_I\}$: set of $I$ experts
$W = \{w_1, \cdots, w_I\}$: experts' weights
$C = \{c_1, \cdots, c_J\}$ set of possible classes: $\hat{y}, \hat{c} \in C$: global and local predictions
$\boldsymbol{\omega}$: sums of weighted predictions for each class
$I = 1$
$s_I = Create\_New\_Expert()$
$w_I = 1$
for $n = 1, \cdots, N$ do
    $\boldsymbol{\omega} = 0$
    for $i = 1, \cdots, I$ do
        $\hat{c} = s_j(\boldsymbol{x}_n)$
        if $\hat{c} \neq y_n$ $and$ $n$ $mod$ $\Omega = 0$ then
            $w_i = \beta w_i$
        end if
        $\omega_{\hat{c}} = \omega_{\hat{c}} + w_i$
    end for
    $\hat{y}_n = \underset{\hat{c}}{argmax}(\omega_{\hat{c}})$
    if $n$ $mod$ $\Omega = 0$ then
        $W = Normalize\_Weights(W)$
        $S, W = Remove\_Experts(S, W, \eta)$
        if $\hat{y}_n \neq y_n$ then
            $I = I + 1$
            $s_I = Create\_New\_Expert()$
            $w_I = 1$
        end if
    end if
    for $i = 1, \cdots, I$ do
        $s_i = Train(s_i, (\boldsymbol{x_n}, y_n))$
    end for
end for
```

training set or parameter update is used during this phase. The adaptation component seeks to *change* the learned concept using new data. This can be achieved by all adaptation types; training set update, parameter update, changes in model structure. Refinement (e.g. via experts' online learning) may be performed simultaneously with adaptation (e.g. via weights adjustment).

As seen from the listed algorithms, the adaptation itself is a highly non-trivial and complex issue. In particular, using multiple adaptive mechanisms requires careful model design to fully use AMs' potential and avoid possible negative effects, for example slowing adaptation when the adaptive elements are working against each other or make the model too sensitive, when the elements are adapting the model in the same direction without synchronisation. Decisions must be made on when each AM should be deployed. In present thesis this is called adaptive strategy. Most of the explored methods employ fixed strategies, which means that the AMs' deployment choice is fixed at the design time of the method and remains the same throughout the prediction process. The most popular types of adaptive strategies are listed below:

- ***Symmetric***: Adaptation of different elements is triggered[10] at the same time (Figure 2.5(a)). Example: changing weights of existing experts and adding a new expert at the same time in (Kolter & Maloof, 2005, 2007).
- ***Asymmetric***: Adaptation of different elements is triggered at the different times using different triggers (Figure 2.5(b)). Examples: Member of ensemble in ASHT are weighted inversely proportional to their error rate and are removed when their complexity exceeds the threshold in (Bifet et al., 2009), perpetual and triggered adaptation in (Kadlec & Gabrys, 2009a).
- ***Bottom-up delegation***: Adaptation is started at the bottom level of hierarchy of adaptive elements. It is gradually passed to higher levels, until a certain stopping condition is met (Figure 2.5(c)). Example: When new data arrives, (Castillo & Gama, 2006) first adapt parameters of Naive Bayes classifier, if this is not improving the performance, they update Bayesian network's structure.

Here, triggers may be change detection mechanisms for example described in Basseville & Nikiforov (1993); Gama et al. (2004); Baena-García et al. (2006). Stationarity tests such as

The methods with multiple AMs reviewed in this chapter usually deploys these AMs in a fixed order, most of the times all at the same time. Some works offer basic flexibility for some AMs. For instance, (Kadlec & Gabrys, 2010) creates new experts when existing ones are not trained on the relevant data, (Gomes Soares & Araújo, 2015a,b) create new experts when the predictive error on an instance is above a set threshold. Once the change is detected, (Alippi et al., 2012) can either switch the active expert if there is one in the pool which corresponds to the new concept, or otherwise create a new expert.

What is common about the above discussed selection of methods and algorithms is that they all use multiple AMs but in a fixed order and manner. Flexible adaptive strategies are able to deploy any AM or combination of AMs which can provide finer control of the adaptation. However this requires carefully designed adaptation strategies, which take into account the specifics of adaptive mechanisms and of data while choosing AM to deploy.

Only few works consider the possibility of adaptation without any fixed order. One of these, (Kadlec & Gabrys, 2009a) presents a plug and play architecture for preprocessing, adaptation and prediction which foresees the possibility of using different adaptation methods modularly, but does not address the method of AM selection. In (Kaneko et al., 2014) the predictive accuracy is assessed to switch between two predictive models. Simple Adaptive Batch Learning Ensemble (SABLE) algorithm (Bakirov et al., 2015) (see Chapter 4) developed in this thesis presents

[10] Refer to Section 2.3.2 for the description of common triggers.

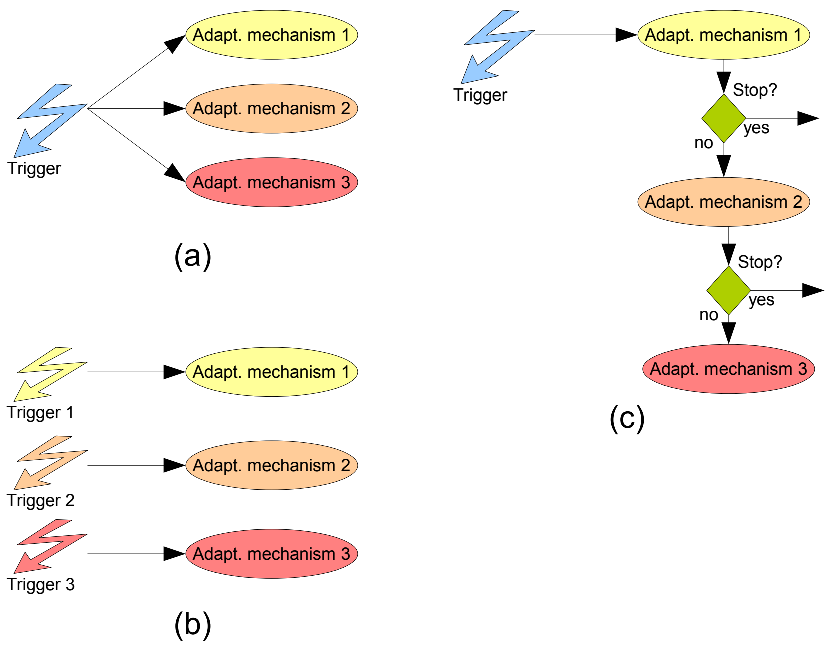


Figure 2.5: *Different synchronisations styles of adaptive elements.*

a framework for using multiple adaptive mechanisms with different possible fixed and flexible adaptive strategies.

Following issues can be caused by fixed adaptation order:

- the dominance of newly added experts which have more weight (Kolter & Maloof, 2005, 2007),
- creating and maintaining a large amount of experts, which leads to increase in computational costs (Kolter & Maloof, 2005, 2007)
- abrupt removal of the models which were trained on a large amount of data (Wang et al., 2003; Minku & Yao, 2012).
- addition of poorly trained experts (Kolter & Maloof, 2007).
- creation of new experts from every batch of new data, which might lead to the dominance of similar experts (Wang et al., 2003; Elwell & Polikar, 2011)
- creation of only a single expert from new batches, which ignores possible changes within the batch (Wang et al., 2003; Elwell & Polikar, 2011; Scholz & Klinkenberg, 2007)
- possibly slowing the adaptation down when using *champion-challenger* schemas, where alternate models are being trained simultaneously with the main predicting one and replace it when change is detected (Nath, 2007; Hulten et al., 2001; Ikonomovska et al., 2010)
- abrupt forgetting (Domingos & Hulten, 2000; Ikonomovska et al., 2010)

- using ineffective sliding window strategies (Widmer, 1993; Klinkenberg & Joachims, 2000; Tsymbal et al., 2008).

Additionally, in Chapter 5 of this thesis, it is shown that fixed adaptation order can lead to lower predictive accuracy than flexible ones.

2.7 Summary

Since this thesis focuses on learning and adaptation for predictive models on non-stationary streaming data, this chapter has given a background in relevant areas. It was discussed that the non-adaptive learning of new data is essentially augmenting existing models with new encountered data. Different methods of learning new data were introduced. It was further continued with the topic of adaptation, which also learns the new data, but additionally reduces the importance of the irrelevant data. The main reason for adaptation is improving accuracy of the model after a change in data. Main generic techniques for models' adaptation are also listed in this chapter.

The chapter continues by providing a more detailed look into different types of adaptive mechanisms. For this purpose, a categorisation of AMs in machine learning is introduced, which includes adaptation of training data, parameters, structure and populations. Modern adaptive algorithms often employ more than one AM. Many ensemble methods belong to this type of algorithms, and are described in more detail in this chapter. They can be adapted by updating their experts, changing the combination weights or adding/removing new experts. Multiple available AMs make the adaptation potentially more flexible. However, by deploying the AMs in fixed manner, which is the common strategy, this advantage seems to be unused. Flexible adaptive strategies may potentially be able to perform better than the fixed ones, when the appropriate adaptation strategies, which can identify the optimal AMs to deploy, are employed. This is analysed in subsequent chapters.

Having summarised the existing research in relevant areas, it is proceeded with the next chapter which introduces the process industry as the case study for experiments.

Chapter 3

Process industry and datasets

3.1 Introduction

This thesis uses data from process industry as a case study for its experimental part. Currently process industry is becoming a data-centric field, which extensively uses predictive modeling in form of data-driven soft sensors (Kadlec et al., 2009). These can be alternatives to ordinary sensors, in the situation where physical measurements are hard to obtain or for backup purposes. The inputs of data-driven soft sensors are known values about the process and their outputs are the necessary measurements. Since the industrial processes are usually highly complex, involving different inputs at various stages, multitudes of physical sensors, mechanical parts susceptible to wear and tear, and being affected by external environmental conditions, the data generated during process' run may suffer from missing variables, co-linearity, outliers and drifts (Kadlec, 2009; Martín Salvador et al., 2016a). Some of these artefacts are usually dealt with during the pre-processing stages of the modeling process. Others, especially changes and drifts require adaptivity from soft-sensors.

Three real world datasets from the process industry (catalyst activation, thermal oxidizer and industrial drier) were used for the experiments in this thesis. Section 3.2 of this chapter presents a short introduction to process industry, informs the reader about soft-sensors and their different types, and focuses on adaptive data-driven soft sensors. In Section 3.4 the datasets used for experimentation and the background processes are presented. The need for adaptation of each dataset is then analysed by building a non-adaptive model using the first batch of the data and applying it to the rest of the data. Analysing the changes in resulting prediction error, conclusions about the changeability of each dataset are presented.

3.2 Introduction to process industry

"Process manufacturing is the production of goods that are typically produced in bulk quantities, as opposed to discrete and countable units. Process manufacturing industries include chemicals, food and beverage, gasoline, paint and pharmaceutical.” [1] Specifically, chemical process plants produce a desired chemical substance from input materials given to the process. The process of conversion of the inputs to the output substance often involves multiple steps. Figure 3.1 shows a diagram of hydrodesulphurization process as an example. The actual chemical reactions take place in the *reactor* of the process plant. To control and monitor these reactions, the reactors are equipped with a multitude of sensors. To split the desired substance from by-products, the output of the reaction is usually passed into *distillation column*. Often, to achieve the desired quality of the output, the distillation has to be done repeatedly multiple times. During this process, the amounts of the target substance and by-products are strictly monitored using sensor readings to ensure the required quality of output. Commonly the sensors in the process plant measure values of temperature, pressure, viscosity, and material flow at different stages of the process.

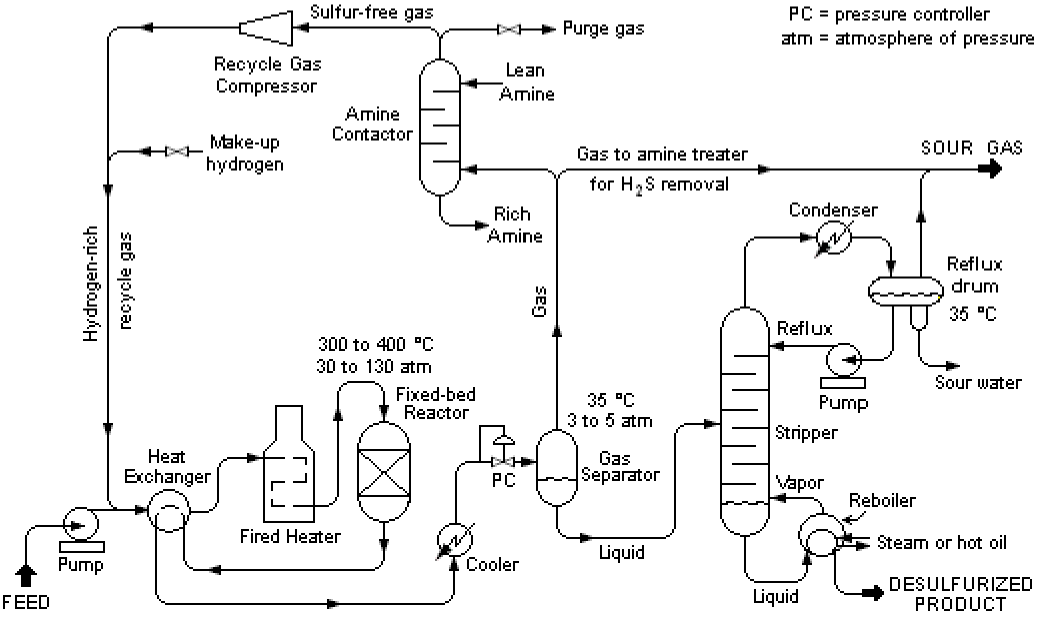


Figure 3.1: *Hydrodesulphurization process diagram (Moore et al., 2011).*

Two different types of processing plants, namely *continuous* and *batch* processing plants can be distinguished. Continuous plants run in an uninterrupted manner, during their whole operation process. The steady-state of the process is still subject to changes caused by changes in required output amounts, changed catalyst activity, physical deterioration of the equipment and other factors. Batch process plants run in a discontinuous fashion, for a certain limited duration. This duration is often dependant on the amount of the required product. Batch processes are often used in microelectronics, pharmaceutical (Cinar et al., 2003), as well as foods or biochemistry industries and for production of special chemicals (Kadlec, 2009).

[1] http://searchmanufacturingerp.techtarget.com/definition/process-manufacturing

3.3 Soft sensors

The careful monitoring of the process state is of paramount importance for the successful run of the plants. This heavily depends on getting accurate measurements from each stage of the process. However taking certain measurements is often excessively costly or too time consuming to keep up with the process. Solution for these cases are often the deployment of *soft sensors*. Soft sensors are models which are used to estimate the necessary values on the basis of physical measurements taken from the running process. These are either (a) model-driven models based on the equations describing physical and chemical characteristics of the process or (b) data-driven models based on using predictive algorithms with real time process data as inputs. Hybrid soft sensors which combine (a) and (b) are also used.

Model-driven models are developed mainly in order to assist with design of the processing plants by incorporating full phenomenological knowledge about the process. The estimates of these models are based on first principles of physics and chemistry, making the reasoning behind them strong and clear. However these type of models have several drawbacks. One of them is requiring extensive knowledge for their development, which may not be available for the process. Furthermore, they often describe the simplified behaviour of the process, without consideration of many external effects on the process. Moreover, these models usually concern only the steady states of the process, excluding changes in the process and transient states. Examples of soft sensors based on model-driven models are (De Wolf et al., 1996; Doyle, 1998; Prasad et al., 2002).

Data-driven models are based on the historical and real-time data which has been collected in the duration of the process run. In addition to the process control and monitoring purposes, the sensor outputs are also saved in large databases called Process Information Management Systems (PIMS). The rise of volume of this data in recent years has popularized the development and use of data-driven soft-sensors (Fortuna et al., 2007). Data-driven soft sensors are essentially predictive models based on machine learning algorithms. They do not require knowledge about the process and are closer to the real process settings. A review of data-driven soft sensors can be found in (Kadlec et al., 2009).

3.3.1 Adaptive data-driven soft sensors

Due to the size and complexity of modern chemical plants, there are many factors which can affect the data which is generated during the process. This causes various changes in data, which encourages the use of data-driven sensors based on adaptive predicted models discussed in the previous chapter. Changes in data may indicate real change in the process. This may be caused by internal factors such as the deterioration of plant's mechanical parts or external factors such as environmental effects, change in the purity of input materials, deactivation of catalyst, etc. These kind of changes may distort the normal run of the process, so they should be detected and addressed promptly. Another type of changes in data, where the process remains unchanged, are caused by the variation in physical sensor readings, which may be caused by their wearing off or malfunctioning. In both cases, the changes in data may negatively affect the accuracy of soft sensors' measurements (Kadlec, 2009). Hence these sensors need to be adapted to reflect the accurate values of the monitored substance.

Adaptive soft-sensors have been an active research area in recent years, starting with the ones based on very well known and commonly used recursive Principal Component Analysis (PCA) and Partial Least Squares (PLS) approaches presented in (Dayal & MacGregor, 1997) and (Li et al., 2000) and more recently proposed adaptive soft sensors, e.g. (Kadlec & Gabrys, 2011; Grbić et al., 2013; Kaneko & Funatsu, 2014, 2015; Jin et al., 2014; Shao et al., 2014; Ni et al., 2014; Jin et al., 2015a,b; Shao & Tian, 2015; Shao et al., 2015b,a).

Soft sensors using local ensembles are described in (Kadlec & Gabrys, 2011, 2009b, 2010; Shao & Tian, 2015; Shao et al., 2015a; Jin et al., 2015b). These methods first identify the disjoint segments of the historical input space where the process produced outputs described by a common model, sometimes also called *receptive fields*. Then they build a model for each receptive field using PLS or Support Vector Regression (Drucker et al., 1996). The models therefore describe different regions of the process. The final prediction is a weighted average of all of the experts. Here, for each new data instance, the weights of experts depend on the location of the observed instance and in some cases the prediction. The AM used in (Kadlec & Gabrys, 2009b) is based on change of models' local weights depending on their error. This model was extended in (Kadlec & Gabrys, 2011) to include adaptation of the base models using the RPLS forgetting. (Kadlec & Gabrys, 2010) further extend the model to include creation of additional experts. (Shao & Tian, 2015; Shao et al., 2015a) use adaptation of base models and adaptive weighting with (Jin et al., 2015b) additionally introducing adaptive offset correction. Another soft sensor based on local ensemble with a moving window and weights change AMs is described in (Grbić et al., 2013). Global ensembles are also used in soft sensor algorithms (Kaneko et al., 2014; Kaneko & Funatsu, 2014, 2015). For example, (Kaneko et al., 2014) use *time difference* ensemble weighting based on the distance between the current input and historical inputs. This method can use either moving window or just-in-time approaches for adaptation.

It can be seen that adaptive soft sensors use various adaptation mechanisms. A review of these mechanisms for soft sensors is given in (Kadlec et al., 2011). The mechanisms target different characteristics of the model; the error, the current location in the input space (or output space), and the temporal distance. Most of the described works above have a common characteristic that whatever the AMs, they are applied at every time step in the same manner. In contrast, the approaches proposed in (Kadlec & Gabrys, 2009a, 2010; Kaneko et al., 2014; Jin et al., 2015a) change the order of the adaptation. In particular, (Kadlec & Gabrys, 2010) creates new experts when existing ones are not built on the relevant data. In (Kaneko et al., 2014) the predictive accuracy is assessed to switch between two predictive models. Again, the predictive accuracy is used to choose between just-in-time model creation and offset update in (Jin et al., 2015a). To analyse the issues related to adaptive mechanisms' deployment, such as the order of AMs and their selection, the experiments on real datasets from the process industry are conducted. The sections below describe the used datasets in detail.

3.4 Process industry datasets

The subsequently introduced datasets are the courtesy of Evonik Industries AG and were used during the INFER EU project[2] (Stahl et al., 2013; Martín Salvador et al., 2014; Žliobaitė & Gabrys, 2014; Lemke et al., 2015)and in previous co-operations between Evonik Industries and Bournemouth University (Kadlec & Gabrys, 2009a; Kadlec, 2009; Kadlec & Gabrys, 2009b, 2010, 2011; Budka et al., 2014). Since the experiments were performed in batch learning scenario, these datasets were split into multiple data batches of equal size, such as $\boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{V}}_1 \cup \cdots \cup \boldsymbol{\mathcal{V}}_K$, where $\boldsymbol{\mathcal{V}} = \{\boldsymbol{X}, \boldsymbol{y}\}$ is the whole dataset, and $\boldsymbol{\mathcal{V}}_1 = \{\boldsymbol{X_1}, \boldsymbol{y_1}\}, \cdots, \boldsymbol{\mathcal{V}}_K = \{\boldsymbol{X_K}, \boldsymbol{y_K}\}$ are the distinct batches of data and K is the number of batches.

3.4.1 Catalyst activation dataset

This simulated data set was used for the NiSIS 2006 competition (Strackeljan, 2006). The description of the process below is taken from the same source. “The reactor to be modelled consists of some 1000 tubes filled with catalyst, used to oxidize a gaseous feed (ethane is taken as example). It is cooled with a coolant supposed to be at constant temperature. The description of the reaction speed is taken from literature and depends strongly non-linearly from temperature. Its exothermal reaction is counteracted by the cooling and leads to a temperature maximum somewhere along the length of the tube. As the catalyst decays, this becomes less pronounced and moves further downstream. The catalyst activity usually decays within some time to zero, a year is taken as example here. The process to be modelled takes input from other, larger processes, so that the feed will vary over the days. The operating personal reacts to this by choosing appropriate operating conditions. The catalyst decay is however much slower than these effects. All measurable influences are considered as input variables for a mathematical multi-input-single-output-model describing relevant process variables (model outputs) representative for chemical industry. The process is equipped with measurements to log all the variations of the feed and the operating conditions. In addition, there are measurements showing some concentrations, flows and a lot of temperatures along the length of a characteristic tube to identify the processes state. All inputs and the output vary dynamically, and there might occur large time-delays.” The reactor and the catalyst are shown in Figure 3.2.

Catalyst dataset includes 14 sensor measurements like flows, concentrations and temperatures from a real process. The target variable is the simulation of catalyst’s activity inside the reactor. The description of the reaction speed is taken from the literature, showing a strong non-linear dependency on temperature. Further complicated processes like cooling and catalyst decay con-

[2] http://www.infer.eu

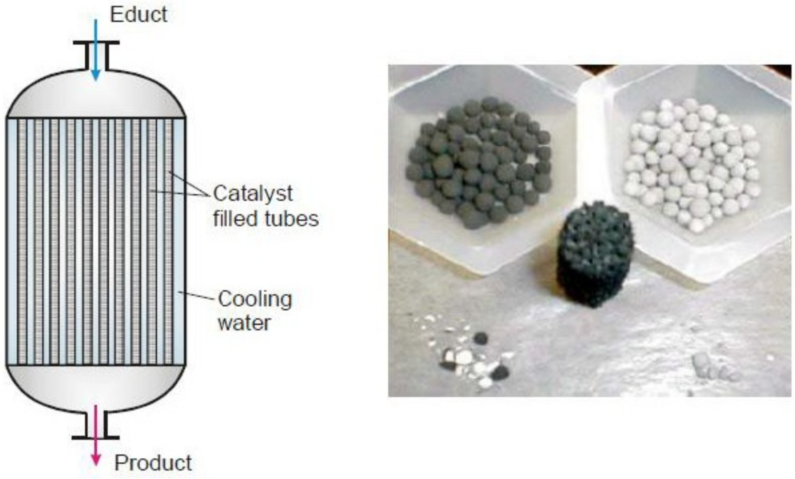


Figure 3.2: *Reactor and catalyst used for the oxidization process (Strackeljan, 2006).*

tribute to changes in the data. The data set covers one year of operation of the plant. Many of the features exhibit high co-linearity and contain high number of outliers.

This dataset includes 5,867 data samples. Two features with mostly missing and 0 values were removed during the preprocessing. In the following chapters this dataset was split into the batches of 50, 100 or 200 instances[3] (denoted Catalyst50, Catalyst100 and Catalyst200 for convenience). Number of batches per each batch size for this and other used datasets can be found in Table 3.1. Features and target value plots of the Catalyst dataset are shown in the Figure 3.3.

Dataset	Number of batches
Catalyst50	116
Catalyst100	58
Catalyst200	29
Oxidizer50	56
Oxidizer100	28
Oxidizer200	14
Drier50	24
Drier100	12
Drier200	6

Table 3.1: *Number of batches per each batch size for the used datasets.*

[3] Here and in subsequent datasets, the choices of batch size are not originating from the process definition, but rather motivated by a need to have sufficient data to bea able to train a predictive model from each batch, as well as having an adequate number of batches to be able to observe the dynamics of the process.

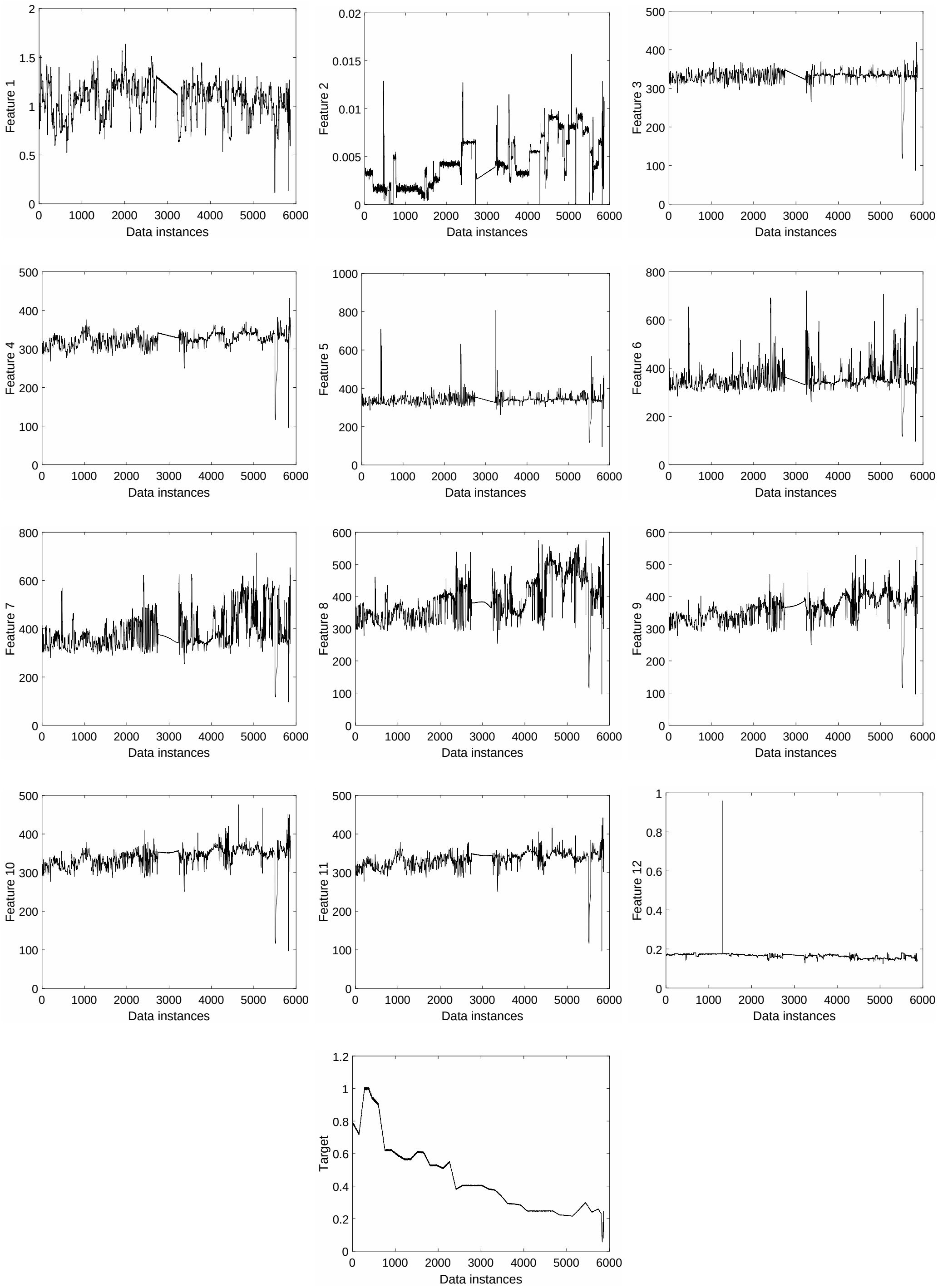
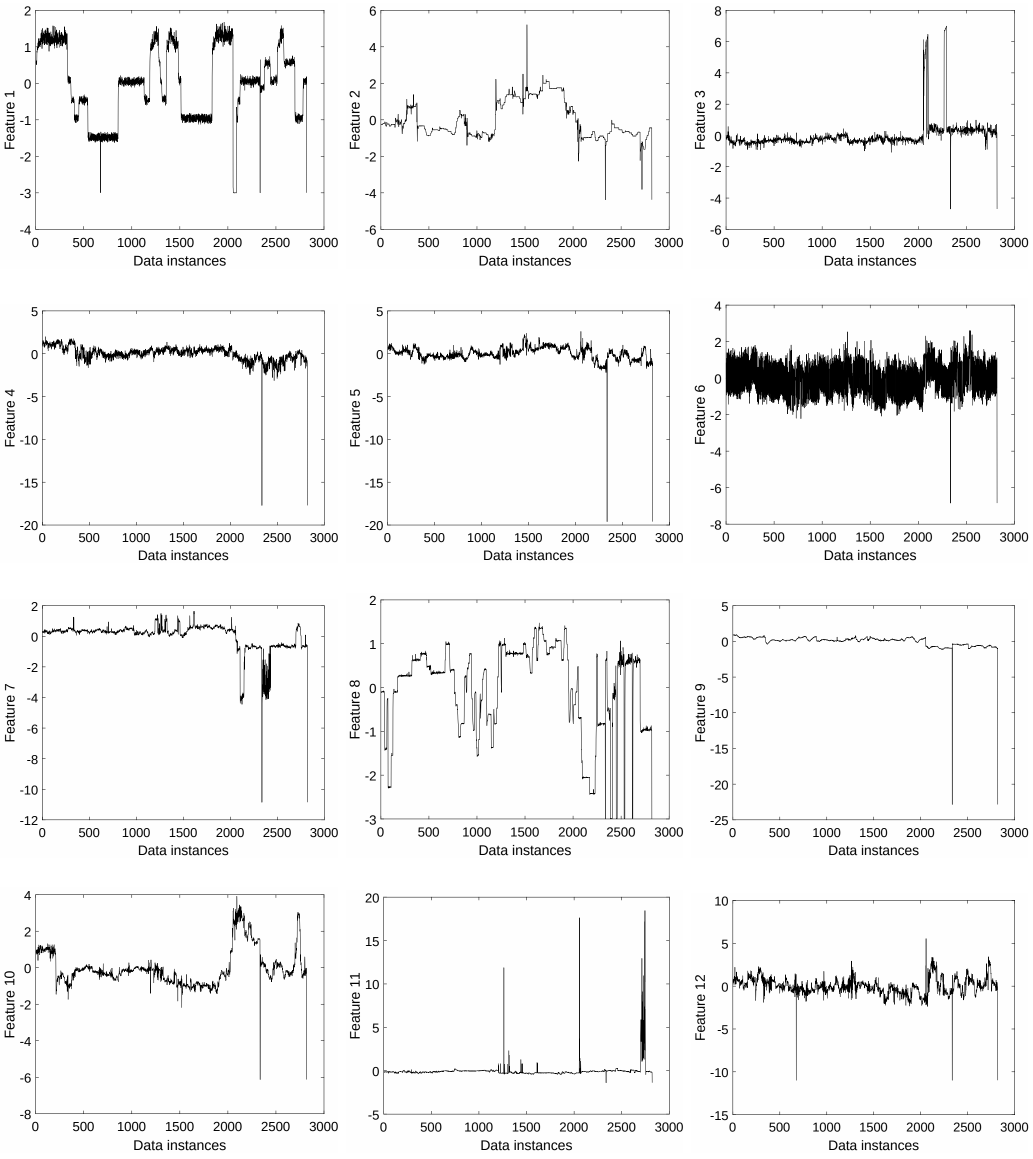


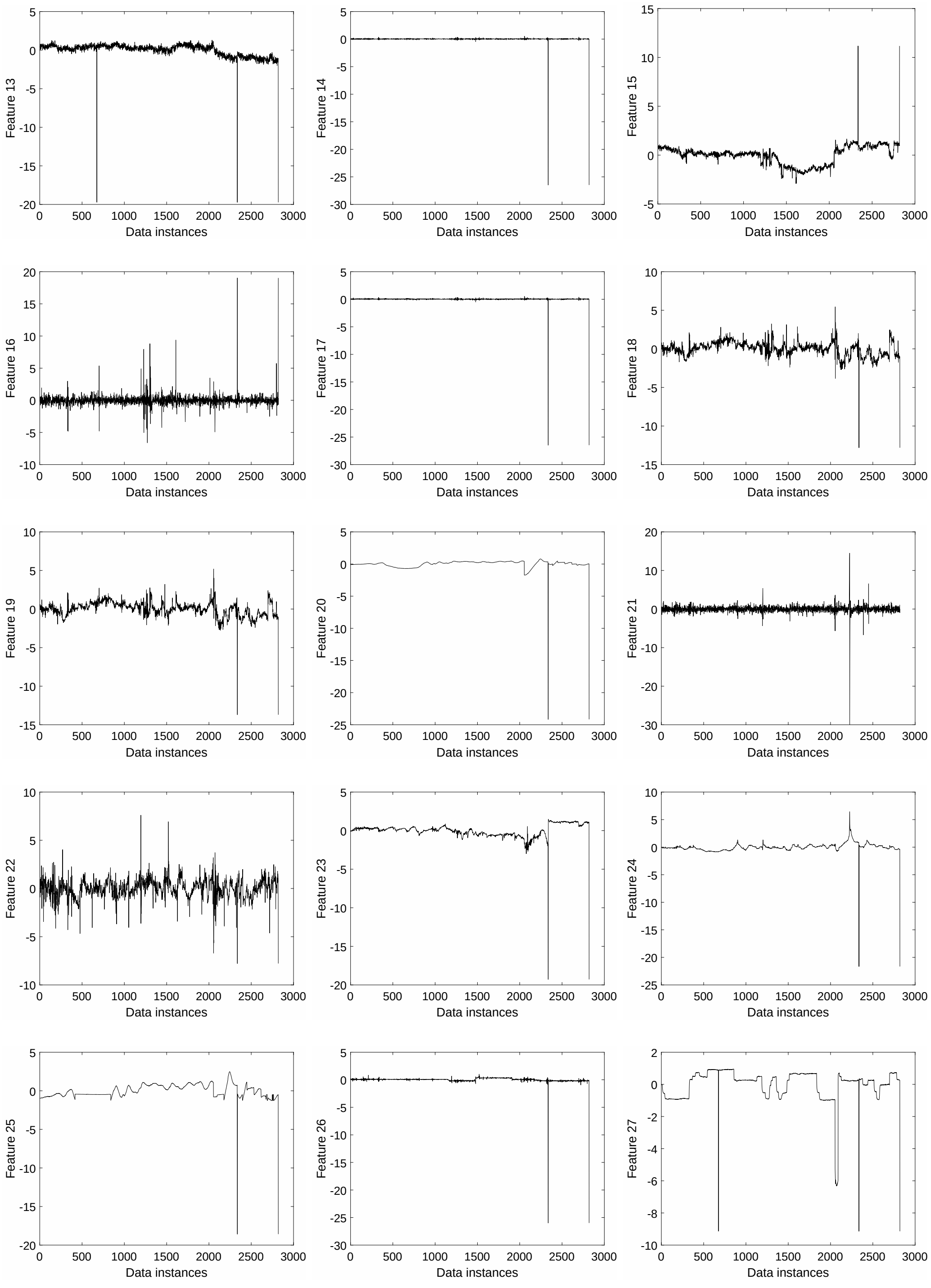
Figure 3.3: *Catalyst dataset features and target value.*

3.4.2 Thermal oxidizer dataset

This dataset deals with the prediction of the concentration of exhaust gas during an industrial process where the task is to predict the concentrations of NO_x in the exhaust gases. The data set consists of 36[4] input features which are physical sensor measurements. These are values like concentrations, flows, pressures and temperatures measured during the operation of the plant. The dataset consists of 2,820 samples with outliers and missing values present in the data. The batch sizes of 50, 100 or 200 instances (Oxidizer50, Oxidizer100, Oxidizer200) were investigated. Number of batches per each batch size for this and other used datasets can be found in Table 3.1. Features and target value plots of the Oxidizer dataset are shown in Figure 3.4.



[4] Several additional features corresponding to time stamps as well as severely affected by missing values were removed by the compilers of the dataset.


Feature 13
5
0
-5
-10
-15
-20
Feature 14
Feature 15
15
10
Feature 16
Feature 17
Feature 18
Feature 19
Feature 20
Feature 21
Feature 22
Feature 23
Feature 24
Feature 25
Feature 26
Feature 27
-25
-30
2
-2
-4
-6
-8
0
500
1000
1500
2000
2500
3000
Data instances

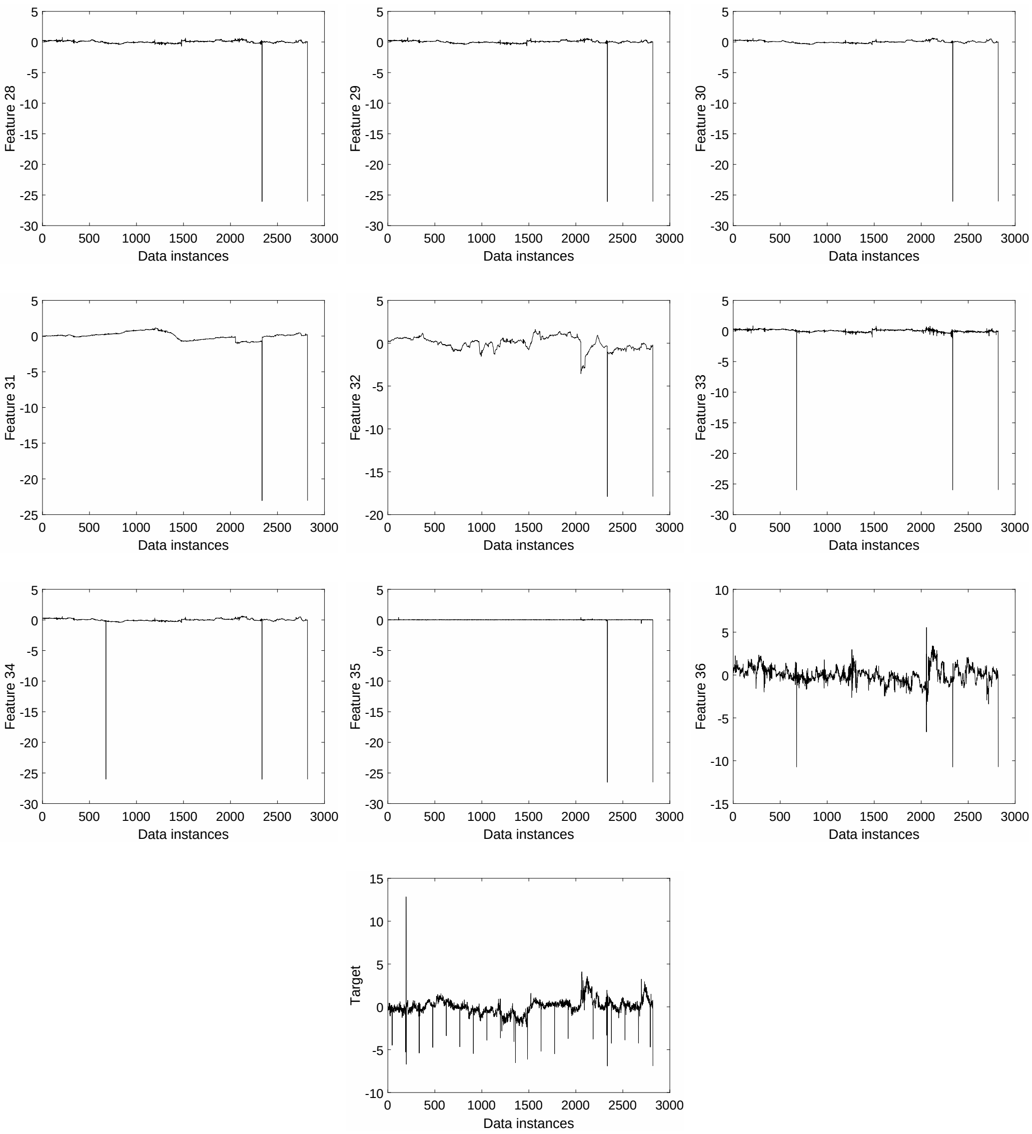
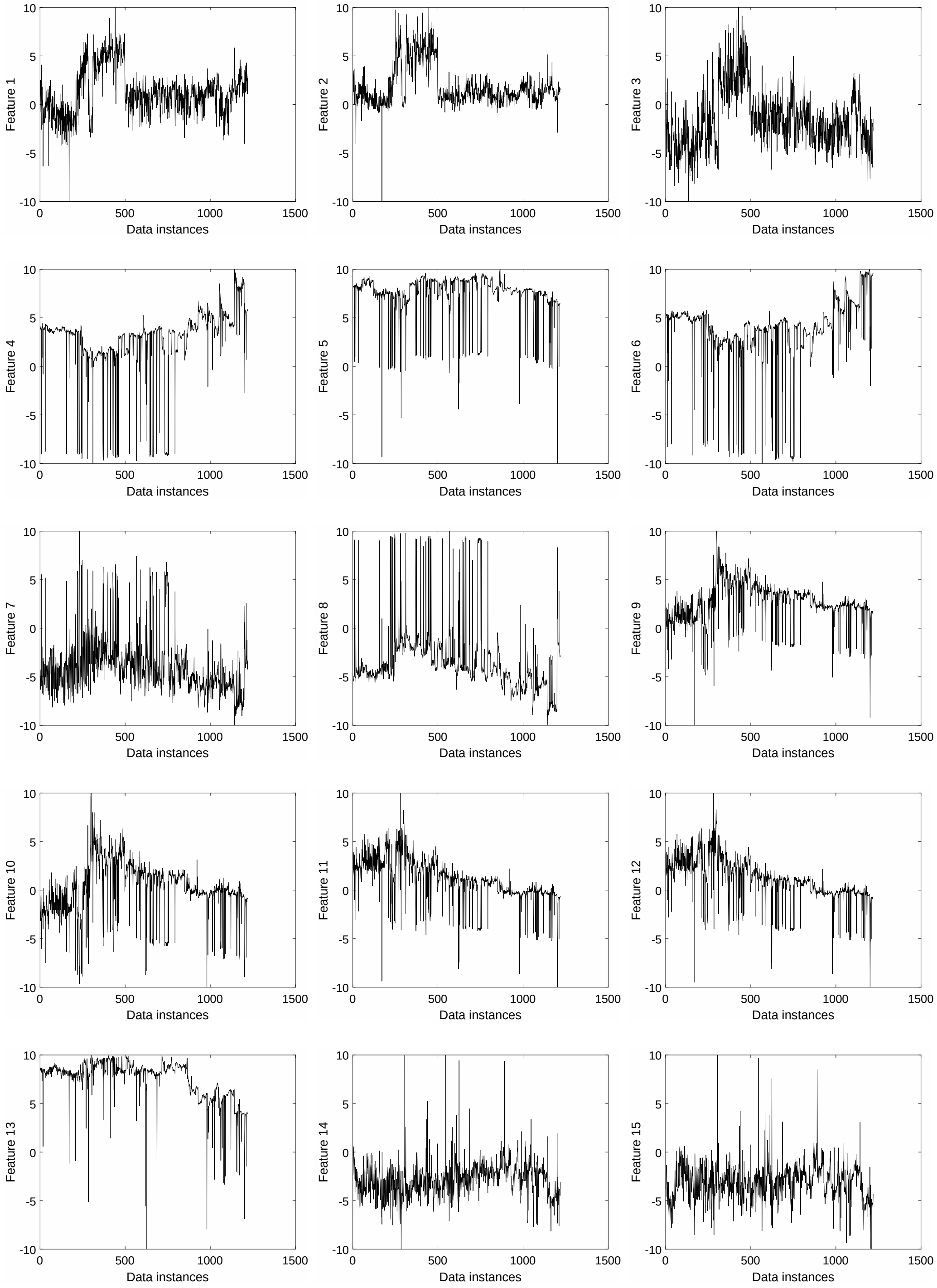


Figure 3.4: *Oxidizer dataset features and target value.*

3.4.3 Industrial drier dataset

The target value of this dataset describes the laboratory measurements of the residual humidity of the process product. The dataset has 19 input features, most of them being temperature, pressure and humidity values measured in the processing plant. The original dataset consists of 1,219 data samples covering almost seven months of the operation of the process. It consists of raw unprocessed data as recorded by the process information and measurement system. Many of the input variables show problems common in industrial data like measurement noise, missing values or data outliers. The only pre-processing step conducted for this data was the removal of 3 input features which mostly consisted of missing data. As with the previous datasets, batch sizes of 50, 100 and 200 instances (Drier50, Drier100 and Drier200) were used for the experiments. Number of batches per each batch size for this and other used datasets can be found in Table 3.1. Features and target value plots of the Drier dataset are shown in the Figure 3.5.



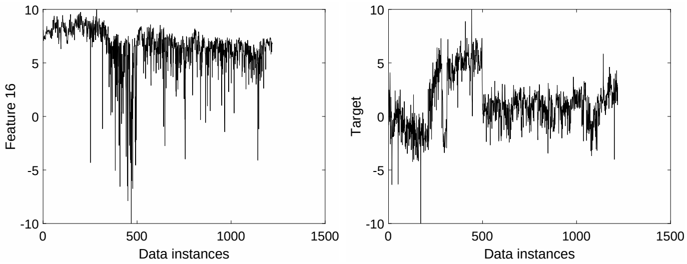


Figure 3.5: *Drier dataset features and target value.*

3.5 Estimating changes in the datasets

This work is focusing on the adaptation of predictive models. As such, it is helpful to establish what is the behaviour of the changes in the datasets described above. As the goal of the thesis is to facilitate the adaptation which improves the predictive performance, the changes in prediction error were chosen to be the indicator of the changes in the data. For this purpose, a predictive model was constructed from the first batch of data, $\boldsymbol{\mathcal{V}}_1$, and was used to predict on the rest of the dataset, $\boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{V}}_2 \cup \cdots \cup \boldsymbol{\mathcal{V}}_K$ without any adaptation. This resulted in the error values $\boldsymbol{\epsilon} = \{\epsilon_{N_*+1}, \cdots, \epsilon_N\}$ where N is the number of instances in the whole dataset, N_* is the number of instances per batch and $\epsilon_n = \hat{y}_n - y_n$ for $n = N_* + 1, \cdots, N$. Here and in the remainder of this chapter, to estimate the changeability of datasets, the batch size of 100 as a has been used for each of them. This size was chosen as a compromise between the sizes 50, 100 and 200 used in subsequent experiments. This choice provides adequate training data for the creation of a model, as well as ample number of batches to observe datasets' behaviour.

(Dasu et al., 2006) suggested quantifying change in data using Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951), which is an information theoretical measure of a distance between two distributions. KL divergence has a number of attractive properties, among them being a generalization of many standard tests of difference, such as t-test or chi-square test (Dasu et al., 2006). As suggested in (Alippi et al., 2016), this thesis uses symmetric KL (sKL) divergence to measure the change magnitude. Given two data distributions ϕ_0 and ϕ_1 sKL is calculated as:

$$\begin{gathered} sKL(\phi_0, \phi_1) = \frac{1}{2}(KL(\phi_0, \phi_1) + KL(\phi_1, \phi_0)) = \\ \frac{1}{2}\Big(\int_{\mathbb{R}^d} log(\frac{\phi_0(x)}{\phi_1(x)})\phi_0(x)dx + \int_{\mathbb{R}^d} log(\frac{\phi_1(x)}{\phi_0(x)})\phi_1(x)dx\Big). \end{gathered} \tag{3.1}$$

Since the experiments will be performed in the batch mode, an intuitive choice is splitting this error vector into the batches of the same size, $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}_2 \cup \cdots \cup \boldsymbol{\epsilon}_K$ and comparing them to each other. The sKL divergences between subsequent error batches, $sKL(\boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}_3), sKL(\boldsymbol{\epsilon}_3, \boldsymbol{\epsilon}_4), \cdots, sKL(\boldsymbol{\epsilon}_{l-1}, \boldsymbol{\epsilon}_l)$ are calculated.

For the considered datasets, the errors and respective sKL divergence values are shown in Fig-

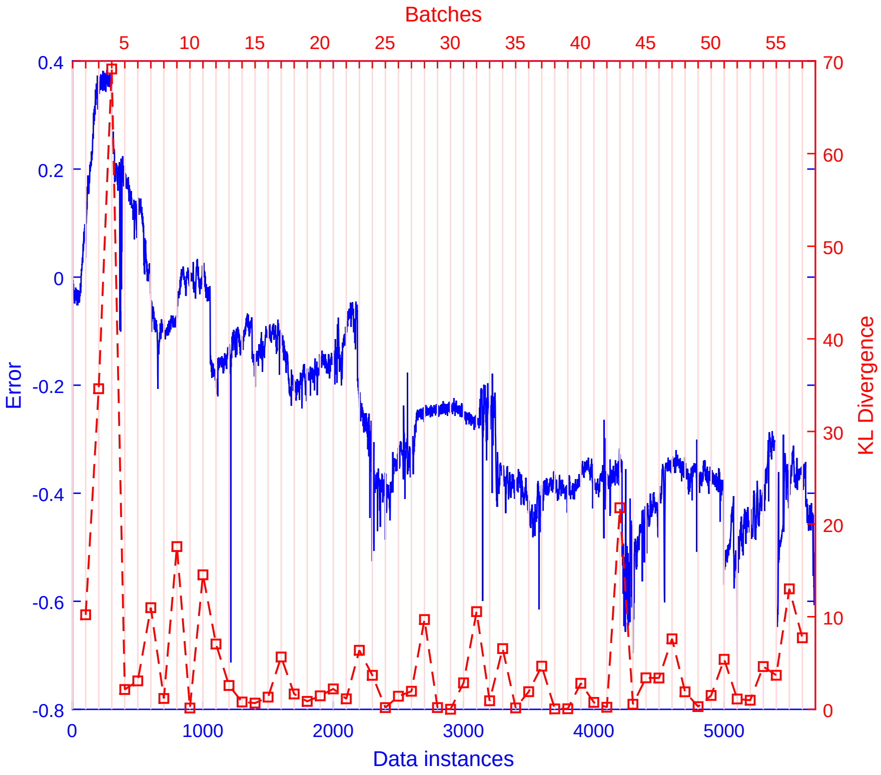


Figure 3.6: *Catalyst100 errors and symmetric Kullback-Leibler divergence values.*

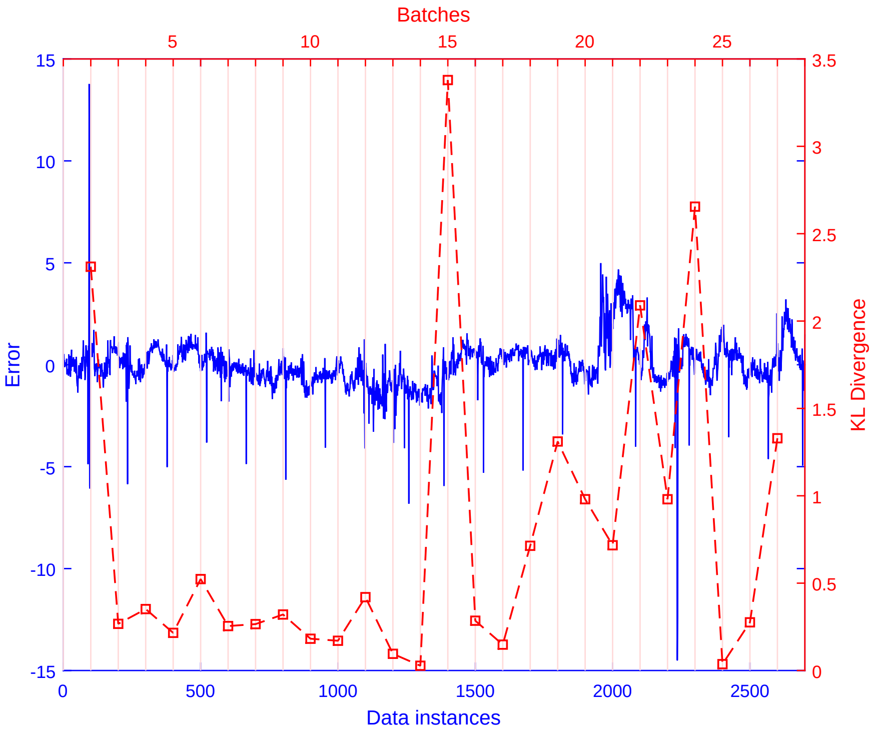


Figure 3.7: *Oxidizer100 errors and symmetric Kullback-Leibler divergence values.*

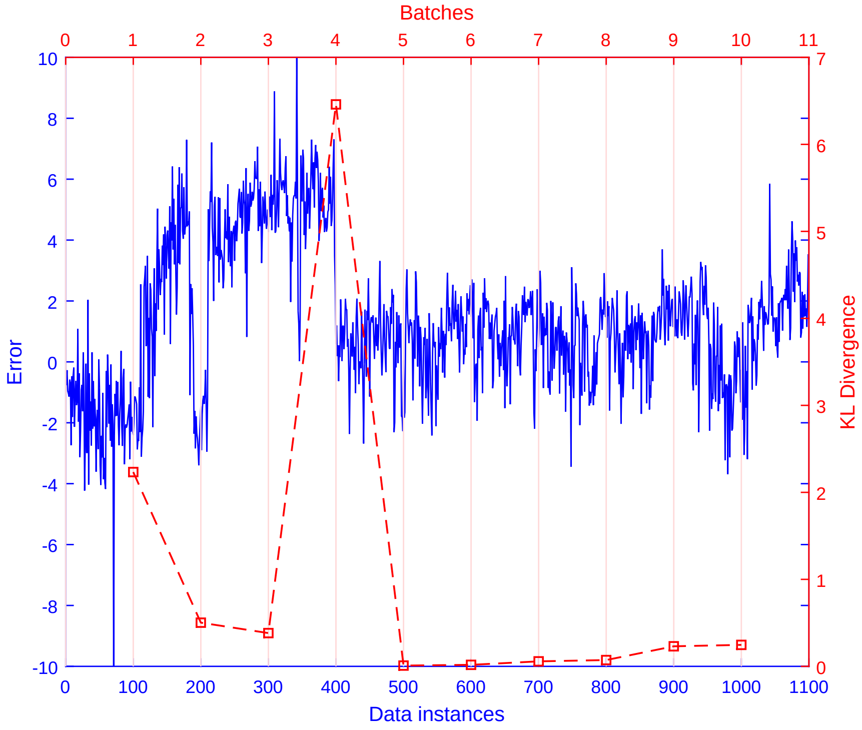


Figure 3.8: *Drier100 errors and symmetric Kullback-Leibler divergence values.*

ures 3.6, 3.7 and 3.8. The histograms of normalised sKL divergence values for all three datasets are presented in Figure 3.9. The results show that the Catalyst dataset is the most changeable out of the three, with rapid changes in the early batches and constantly changing throughout, with only a few similar subsequent batches. The other two datasets are much less changing. The Oxidizer dataset is relatively stable in the first half and becomes more changing in the second half. The Drier dataset is characterised with one sudden change and stable behaviour after it. This confirms the background expert knowledge about the processes (Kadlec, 2009).

3.6 Summary

The process industry was used as a case study for research in this thesis, with the datasets used for the subsequent experiments originating from this area. Therefore, this field and its specifics have been briefly presented in this chapter. Then the three datasets with their characteristics are described. Since the focus of this thesis is put on the adaptation of predictive models, the changes in the datasets were quantified and visualised using symmetric Kullback-Leibler divergence. The results confirm the background knowledge about the datasets' behaviour - Catalyst dataset being the most changing, and Oxidizer and Drier datasets less so. The choice of the datasets have also been dictated by the desire to have a spectrum of processes with different dynamic behaviours in order to both test and illustrate the behaviour of the investigated and proposed adaptive approaches and algorithms.

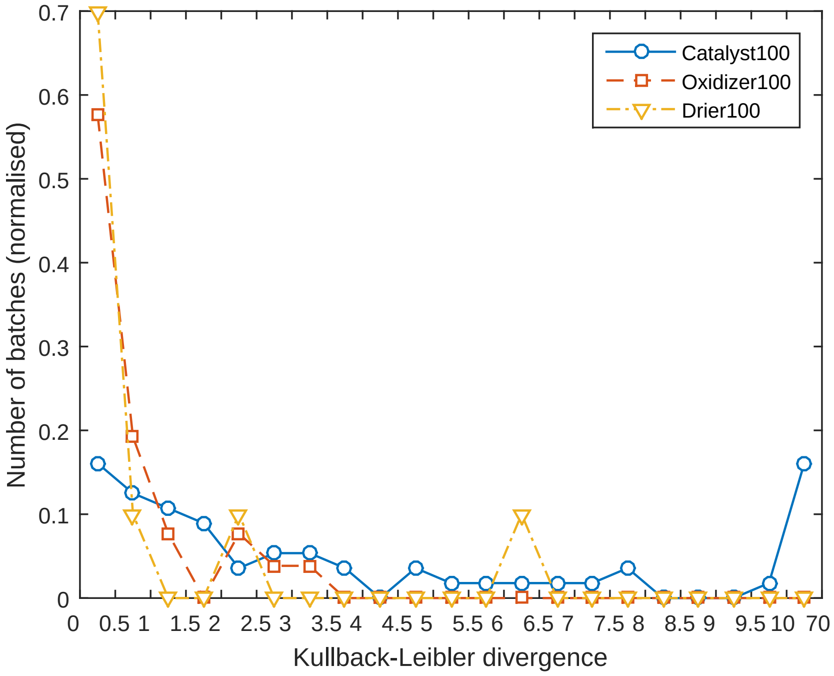


Figure 3.9: *Histograms of symmetric Kullback-Leibler values for all datasets.*

Chapter 4

Effects of the choice of adaptive mechanism

4.1 Introduction

The previous chapters have discussed the adaptive predictive methods and the fact that they often offer multiple adaptive mechanisms to deal with the changes in the data. As noted in the Chapter 2, there are various methods of deploying these multiple AMs; all of the AMs or a subset of them may be deployed at different times during the prediction process. However the effects of choice of deployed AMs on the predictive accuracy of the algorithm have not been explicitly explored before. Moreover, the question whether the AM choice significantly affects the accuracy of the algorithm has not been answered. This chapter sheds light into this issue by an empirical analysis of the behaviour of an algorithm with multiple AMs, while predicting in streaming fashion on the datasets introduced in the previous chapter.

The chapter starts with formalising adaptive mechanisms, batch learning and adaptation setting used in the thesis in Section 4.2. Subsequently, the algorithm Simple Adaptive Batch Learning Ensemble (SABLE) and its AMs which are the main vehicle for the experimentation in this research are introduced in Section 4.3. SABLE is an adaptive regression algorithm which includes such AMs as update, reweighing, adding and removal of the experts.

SABLE deploys a selected set of AMs after the true values for batches of data are received (workflow of this process is presented in Figure 4.1). Thus the sequences of AMs used throughout the process are obtained. It is investigated whether the choice of AMs matter, by comparing a greedy optimal[1] AM sequence, i.e. the AM sequence which is known to minimize the error for each subsequent batch of data, with randomly generated AM sequences for each data set. The results of experiments suggest that the optimal AM sequences indeed result in significantly better performance than the random AM sequences.

[1] For conciseness purposes, unless stated otherwise, this sequence will be also referred to as the "optimal sequence" and the strategy which deploys this sequence as the "optimal strategy".

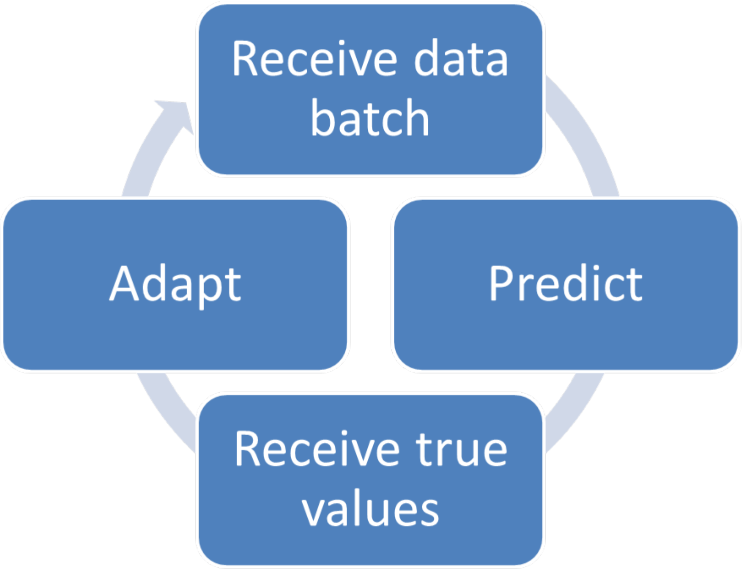


Figure 4.1: *Assumed workflow of the prediction and adaptation on streaming data.*

4.2 Formulation

Recalling Equations 2.1 and 2.2, in the experiments in this and the following chapter, it is assumed that the data is generated by an unknown time varying process which can be formulated as:

$$y_\tau = \psi(\boldsymbol{x}_\tau, \tau) + \xi_\tau, \tag{4.1}$$

where ψ is the unknown function, ξ_τ a noise term, $\boldsymbol{x} \in \mathbb{R}^M$ is an input data instance, and y_τ is the observed output at time τ.[2] Here $\boldsymbol{x}$ represents all measurable/observable variables (for example sensor readings) which are used as inputs to the predictive model as expressed in Equation 4.2. Then the predictive model at a time τ can be considered as a function:

$$\hat{y}_\tau = f_\tau(\boldsymbol{x}_\tau, \Theta_f), \tag{4.2}$$

where $\hat{y}_\tau$ is the prediction, f_τ is an approximation (i.e. the model) of $\psi(\boldsymbol{x}, \tau)$, and Θ_f is the associated parameter set. The estimate, f_τ, evolves via adaptation as each batch of data arrives as is now explained.

4.2.1 Adaptation

In the batch streaming scenario considered in this chapter, data arrives in batches with $\tau \in \{\tau_k \cdots \tau_{k+1} - 1\}$, where τ_k is the start time of the k-th batch. If N_k is the size of the k-th

[2] It should be noted that Equation 4.1 has not intended to and does not explicitly take into account the dynamics of the data generating process as is commonly done in the state-space model representation used in the control engineering.

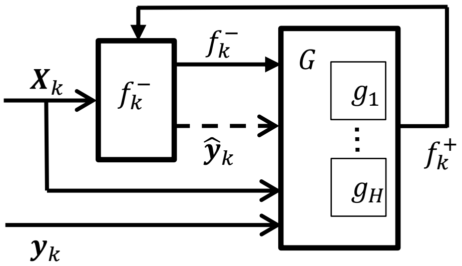


Figure 4.2: *Adaptation with multiple AMs. Optional inputs are shown with dashed lines.*

batch, $\tau_{k+1} = \tau_k + N_k$. It then becomes more convenient to index the model by the batch number k, denoting the inputs as $\boldsymbol{X}_k = \{\boldsymbol{x}_{\tau_k}, \cdots, \boldsymbol{x}_{\tau_{k+1}-1}\}$, the outputs as $\boldsymbol{y}_k = \{y_{\tau_k}, \cdots, y_{\tau_{k+1}-1}\}$.

The *a priori* predictive function at batch k is denoted as f_k^-, and the *a posteriori* predictive function, i.e. the adapted function given the observed output, as f_k^+. An *adaptive mechanism*, $g(\cdot)$, may thus formally be defined as an operator which generates an updated prediction function based on *a priori* predictive function f_k^-, the batch $\boldsymbol{\mathcal{V}}_k = \{\boldsymbol{X}_k, \boldsymbol{y}_k\}$ and other optional inputs. This can be written as:

$$g_k(f_k^-, \boldsymbol{X}_k, \boldsymbol{y}_k, \Theta_g, \hat{\boldsymbol{y}}_k) : f_k^- \to f_k^+. \tag{4.3}$$

or alternatively as $f_k^+ = f_k^- \circ g_k$ for conciseness. Note that $\hat{\boldsymbol{y}}_k$ is optional argument and Θ_g is the set of parameters of g. The function is propagated into the next batch as $f_{k+1}^- = f_k^+$. Predictions on $\boldsymbol{\mathcal{V}}_k$ are always made using the *a priori* function f_k^-.

A situation is examined when a choice of multiple, different AMs, $\{\emptyset, g_1, ..., g_H\} = G$, is available. After the true values $\boldsymbol{y}_k$ for the batch k are received, any AM $g_{h_k} \in G$ can be deployed on it (h_k denotes the AM deployed on batch k). As the history of all adaptations up to the current batch, k, have in essence created f_k^-, that sequence $g_{h_1}, ..., g_{h_k}$ is called an *adaptation sequence*. Note that the option of applying no adaptation denoted by $\emptyset$ is also included. In this formulation, only one element of G is applied for each batch of data. Deploying multiple adaptation mechanisms on the same batch is accounted for with their own symbol in G (for example, it is possible to have $g_{ab} \in G$ where $f \circ g_{ab} = f \circ g_a \circ g_b$ for a predictive model f and $g_a, g_b \in G$). Figure 4.2 illustrates the formulation of adaptation. The workflow, earlier presented in Figure 4.1 can be formulated in Algorithm 2.

Algorithm 2 Assumed workflow of the prediction and adaptation on streaming data.

```
$f_1^-$: initial a priori prediction function
for $k = 1, 2, ...$ do
    Obtain input values of batch $k$, $\boldsymbol{X}_k$
    Predict target values, $\hat{\boldsymbol{y}}_k = f_k^-(\boldsymbol{X}_k, \Theta_f)$
    Obtain true target values of batch $k$, $\boldsymbol{y}_k$
    Adapt prediction function, $g_k(\boldsymbol{X}_k, \boldsymbol{y}_k, \Theta_g, f_k^-, \hat{\boldsymbol{y}}_k) : f_k^- \to f_k^+$.
end for
```

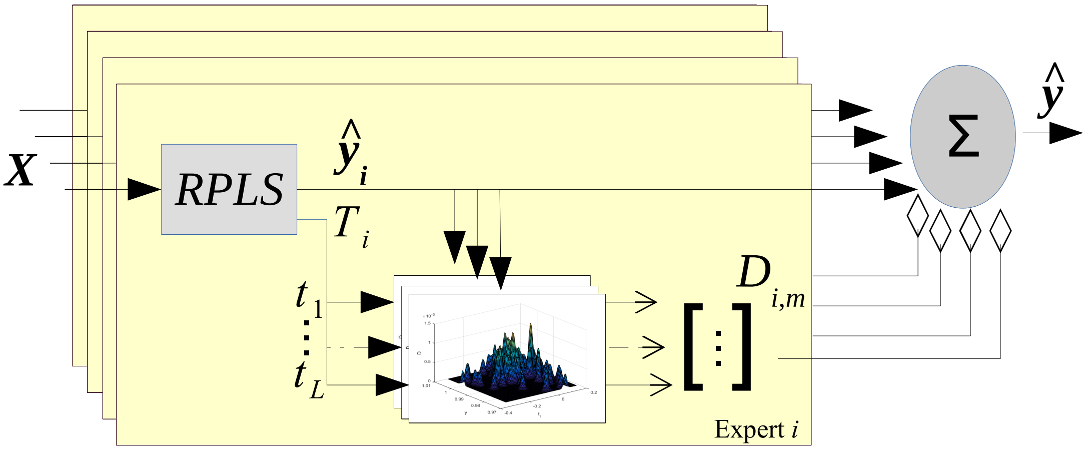


Figure 4.3: *Block diagram of SABLE. Here, $\boldsymbol{T} = [\boldsymbol{t}_1, ..., \boldsymbol{t}_L]$ is the PLS score matrix of i-th expert consisting of corresponding latent vectors (see 2.5.1.2 for more detail).*

4.3 Simple Adaptive Batch Local Ensemble algorithm

To perform experiments, a modelling framework which has the ability to implement several different types of adaptation mechanisms was required. Thus, an adaptive ensemble method was chosen, because, as described in Section 2.5, these methods naturally provide several adaptive mechanisms and because their popularity for learning on non-stationary data. As a part of the research for this thesis, a method called the Simple Adaptive Batch Local Ensemble (Bakirov et al., 2015) was developed extending Incremental Local Learning Soft Sensing Algorithm (ILLSA) (Kadlec & Gabrys, 2011). ILLSA uses an ensemble of models, called *base learners*, with each base learner implemented using a linear model formed by RPLS. To get the final prediction, the predictions of base learners are combined using input/output space dependent weights (i.e. local learning). SABLE differs from ILLSA in that it is designed for batches of data whereas ILLSA works and adapts on the basis of individual data points. SABLE additionally allows the creation and pruning of base learners for adaptation purposes. Moreover, SABLE supports deploying different adaptive strategies. SABLE builds the experts' descriptors (Section 4.3.1) and combines the experts (Section 4.3.2) in the same way as ILLSA. PLS was chosen as a base learner because it is widely used for predictions in chemical processes where high dimensional datasets tend to have low-dimensional embeddings. Figure 4.3 shows the diagram of SABLE model .

4.3.1 Building of experts' descriptors

The relative (to each other) performance of experts varies in different parts of the input/output space. In order to quantify this a *descriptor* is used. Descriptors of experts are distributions of their weights with the aim to describe the area of expertise of the particular local expert.

They describe the mappings from m-th input feature[3], x^m, and output, y, to a weight, denoted $\mathcal{D}_{i,m}(x^m, y)$, for all features $\{x^1, \cdots, x^M\}$ and all experts $\{s_1, \cdots, s_I\}$. The descriptor is constructed using a two-dimensional Parzen window method (Parzen, 1962) as:

$$\mathcal{D}_{i,m} = \frac{1}{||\boldsymbol{\mathcal{V}}_i^{tr}||} \sum_{n=1}^{||\boldsymbol{\mathcal{V}}_i^{tr}||} v(\boldsymbol{x}_n)\Phi(\mu_n^m, \boldsymbol{\Sigma}) \tag{4.4}$$

where $\boldsymbol{\mathcal{V}}_i^{tr}$ is the training data used for i-th expert, $||\boldsymbol{\mathcal{V}}_i^{tr}||$ is the number of instances it includes, $v(\boldsymbol{x}_n)$ is the weight of sample point's contribution which is defined in Equation 4.5, $\boldsymbol{x}_n$ is the n-th sample of $\boldsymbol{\mathcal{V}}_i^{tr}$, $\Phi(\mu_n^m, \boldsymbol{\Sigma})$ is two-dimensional Gaussian kernel function with mean value $\mu = (x_n^m, y_n)$ and variance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times2}$ with the kernel width, σ, at the diagonal positions. σ, is unknown and must be estimated as a hyperparameter of the overall algorithm[4].

The weights $v(\boldsymbol{x}_n)$ for the construction of the descriptors (see Eq. 4.4) are proportional to the prediction error of the respective local expert:

$$v(\boldsymbol{x}_n) = \exp(-(\hat{y}_n - y_n)^2) \tag{4.5}$$

Finally, considering that there are M input variables and I models, the descriptors may be represented by a matrix, $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{M\times I}$ called the *descriptor matrix*. An example of a descriptor is shown on the Figure 4.4.

4.3.2 Combination of experts' predictions

During the run-time phase, SABLE must make a prediction of the target variable given a batch of new data samples. This is done using a set of trained local experts $S = \{s_1, \cdots, s_I\}$ and descriptors $\boldsymbol{\mathcal{D}}$. Each expert makes a prediction $\hat{y}_i$ for a data instance $\boldsymbol{x}$. If each expert s_i produces a prediction $\hat{y}_i$ (i.e. $\hat{y}_i = s_i(\boldsymbol{x}),\ i \in 1\cdots I$), the final prediction $\hat{y}$ is the weighted sum of the local experts' predictions:

$$\hat{y} = \sum_{i=1}^{I} w_i(\boldsymbol{x}, \hat{y}_i)\hat{y}_i \tag{4.6}$$

where $w_i(\boldsymbol{x}, \hat{y}_i)$ is the weight of the i-th local expert's prediction. The weights are calculated using the descriptors, which estimate the performance of the experts in the different regions of the input space. This can be expressed as the posterior probability of the j-th expert given the test sample $\boldsymbol{x}$ and the local expert prediction $\hat{y}_j$:

$$w_j(\boldsymbol{x}, \hat{y}_j) = \mathcal{P}(j|\boldsymbol{x}, \hat{y}_j) = \frac{\mathcal{P}(\boldsymbol{x}, \hat{y}_j|j)\mathcal{P}(j)}{\Sigma_{i=1}^{I}\mathcal{P}(\boldsymbol{x}, \hat{y}_i|i)\mathcal{P}(i)}, \tag{4.7}$$

[3] For the base methods which transform the input space, such as PLS, the transformed input arguments are used instead of original ones.

[4] In this research the inputs are first divided by their standard deviation allowing to assume an isotropic kernel for simplicity and also to reduce the number of parameters to be estimated.

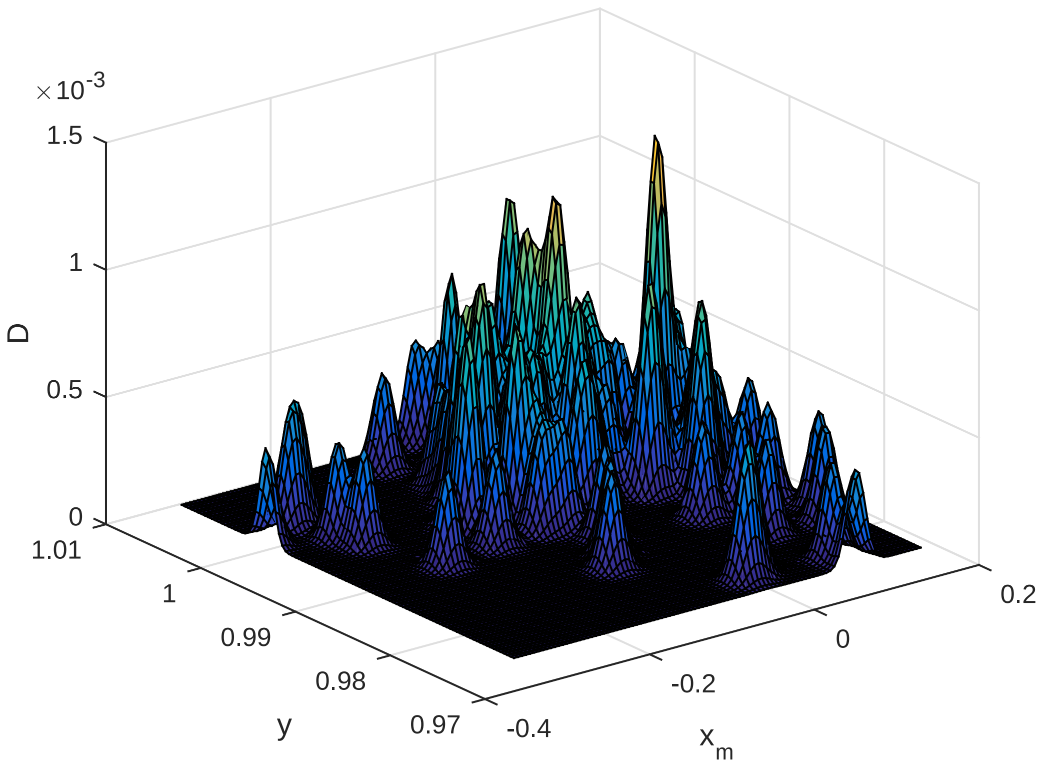


Figure 4.4: *SABLE descriptor.*

where $\mathcal{P}(j)$ is the *a priori* probability of the j-th expert[5], $\Sigma_{i=1}^{I}\mathcal{P}(\boldsymbol{x},\hat{y}_i)\mathcal{P}(i)$ is a normalisation factor and $\mathcal{P}(\boldsymbol{x},\hat{y}_j|j)$ is the likelihood of $\boldsymbol{x}$ given the expert, which can be calculated by reading the descriptors at the positions defined by the sample $\boldsymbol{x}$ and prediction $\hat{y}_j$:

$$\mathcal{P}(\boldsymbol{x},\hat{y}_j|j) = \prod_{m=1}^{M} p(x^m,\hat{y}_j|j) = \prod_{m=1}^{M} mathcalD_{j,m}(x^m,\hat{y}_j). \tag{4.8}$$

Equation 4.8 shows that the descriptors $\mathcal{D}_m$ are sampled at the positions which are given on one hand by m-th feature of the sample point $\boldsymbol{x}$, x^m and on the other hand by the predicted output $\hat{y}_j$ of the local expert corresponding to the j-th receptive field. Sampling the descriptors at the positions of the predicted outputs may result in different outcome than sampling at the positions of correct target values, because the predictions are not necessarily similar to the correct values. However, the correct target values are not available at the time of the prediction. The rationale for this approach is that the local expert is likely to be more accurate if it generates a prediction which conforms with an area occupied by a large number of true values during the training phase.

[5] Equal for all local experts in current implementation, different values could be used for experts' prioritization.

4.3.3 Experts' pruning

To reduce the number of redundant experts, after the processing of batch k, some of those that deliver similar predictions on $\boldsymbol{\mathcal{V}}_k$ can be removed with their descriptors merged. This process is implemented as follows. The prediction vectors of each expert $s_i \in S$ on batch $\boldsymbol{\mathcal{V}}_k$, $\{\hat{\boldsymbol{y}}_1, \cdots, \hat{\boldsymbol{y}}_I\}$ are obtained. The similarities between prediction vectors are pairwise tested using Student's t-test (Student, 1908)[6]. Then p-values of t-test results between each expert pair's prediction values are

$$\boldsymbol{P} = \begin{matrix} p_{1,1} & p_{1,2} & \cdots & p_{1,I} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,I} \\ \vdots & \vdots & \ddots & \vdots \\ p_{I,1} & p_{I,2} & \cdots & p_{I,I} \end{matrix}.$$

The pruning is conducted if $p_{i,j} > \alpha$ where $p_{i,j} = max(\boldsymbol{P})$ (maximum value of $\boldsymbol{P}$) and α is the significance threshold chosen as 0.05. During the pruning, the older of the two experts, s_i and s_j, is removed, while their descriptors are added together to create a merged descriptor. This process is repeated until $p_{i,j} <= \alpha$ for $p_{i,j} = max(\boldsymbol{P})$.

4.4 Adaptive mechanisms

The SABLE algorithm allows the use of different adaptive mechanisms. AMs are deployed as soon as the true values for the batch are available and before predicting on the next batch. The trivial case of no AM deployment is also an option denoted as AM0. The AMs that are used in this work are described in the following sections.

4.4.1 Batch learning

The simplest AM augments existing data with data from the new batch and retrains the expert. Given set of experts $s_i \in S$ and measurements of the actual values, $\boldsymbol{y}$, batch $\boldsymbol{\mathcal{V}}_k$ is partitioned into subsets in the following fashion:

$$(\boldsymbol{x}_j, y_j) \in \boldsymbol{\mathcal{V}}_z \mid z = \underset{i \in 1 \cdots I}{argmin} \langle s_i(\boldsymbol{x}_j), y_j \rangle \tag{4.9}$$

for every instance $(\boldsymbol{x}_j, y_j) \in \boldsymbol{\mathcal{V}}_k$. This creates subsets $\boldsymbol{\mathcal{V}}_i, i \in 1 \cdots I$ such that $\cup_{i=1}^{I} \boldsymbol{\mathcal{V}}_i = \boldsymbol{\mathcal{V}}_k$. Then each expert is updated using the respective dataset $\boldsymbol{\mathcal{V}}_i$. This process updates experts only with the instances where they achieve the most accurate predictions, thus encouraging the specialisation of experts, promoting diversity of the ensemble and ensuring that a single data instance is not included in the training data of multiple experts. The updated is performed using RPLS batch

[6] T-test assumes the normal distribution of prediction vectors, which may not always be the case. In these cases, non-parametric tests which relax this assumption, for example Mann-Whitney U test (Mann & Whitney, 1947) may be considered.

retraining described in Section 2.5.1.2, which does not require storing of the old training data. L, number of latent variables for RPLS is a hyperparameter of SABLE.

Batch learning is a parameter adaptation mechanism (Section 2.4.3) which operates on a single expert level (Section 2.5.1). This AM will be denoted as AM1 in the description of the experiments below.

4.4.2 Batch learning with forgetting

This AM is similar to one described in Section 4.4.1 but uses decay which reduces the weight of the experts historical training data, making the most recent data more important. It is realised via RPLS update with forgetting factor λ (see Section 2.5.1.2). λ is a hyperparameter of SABLE. Similarly to the previous AM, batch learning with forgetting is a parametric adaptation mechanism which operates on a single expert level. This AM will be denoted as AM2.

4.4.3 Descriptors update

This AM recalculates the local descriptors using the new batch as described in the Section 4.3.1 creating a new descriptor set $\boldsymbol{\mathcal{D}}^1$. These are merged with a previous descriptors set, $\boldsymbol{\mathcal{D}}^0$ in the following fashion:

$$\mathcal{D}_{i,m} = \delta_0 \mathcal{D}^0_{i,m} + \delta_1 \mathcal{D}^1_{i,m} \tag{4.10}$$

for all experts $i \in 1 \cdots I$ and features $m \in 1 \cdots M$, where δ_0 and δ_1 are respective update weights associated with old and new descriptors and $\delta_0 + \delta_1 = 1$. This means that when $\delta_0 = 0$, descriptors update is essentially their recalculation using the most recent batch. The descriptor update weights are hyperparameters of SABLE.

Descriptors update is a parameter adaptation mechanism which operates on ensemble combination level (Section 2.5.2). This AM will be denoted as AM3.

4.4.4 Creation of new experts

New expert s_{new} is created from $\boldsymbol{\mathcal{V}}_k$. Then it is checked if any of the experts from $S_{k-1} \cup s_{new}$, where S_{k-1} is the experts pool after processing of batch $k-1$, can be pruned as described in Section 4.3.3. Finally the descriptors of all resulting experts are updated (Section 4.4.3). Creation of new experts is a structural adaptation mechanism (Section 2.4.2) which operates on experts' adding/removal level (Section 2.5.3). This AM will be denoted as AM4.

An example of the general principle of SABLE's operation including few selected adaptation mechanisms is illustrated in the Figure 4.5. It shows how the model changes after deploying AM4, AM2 and AM3 in a sequence.

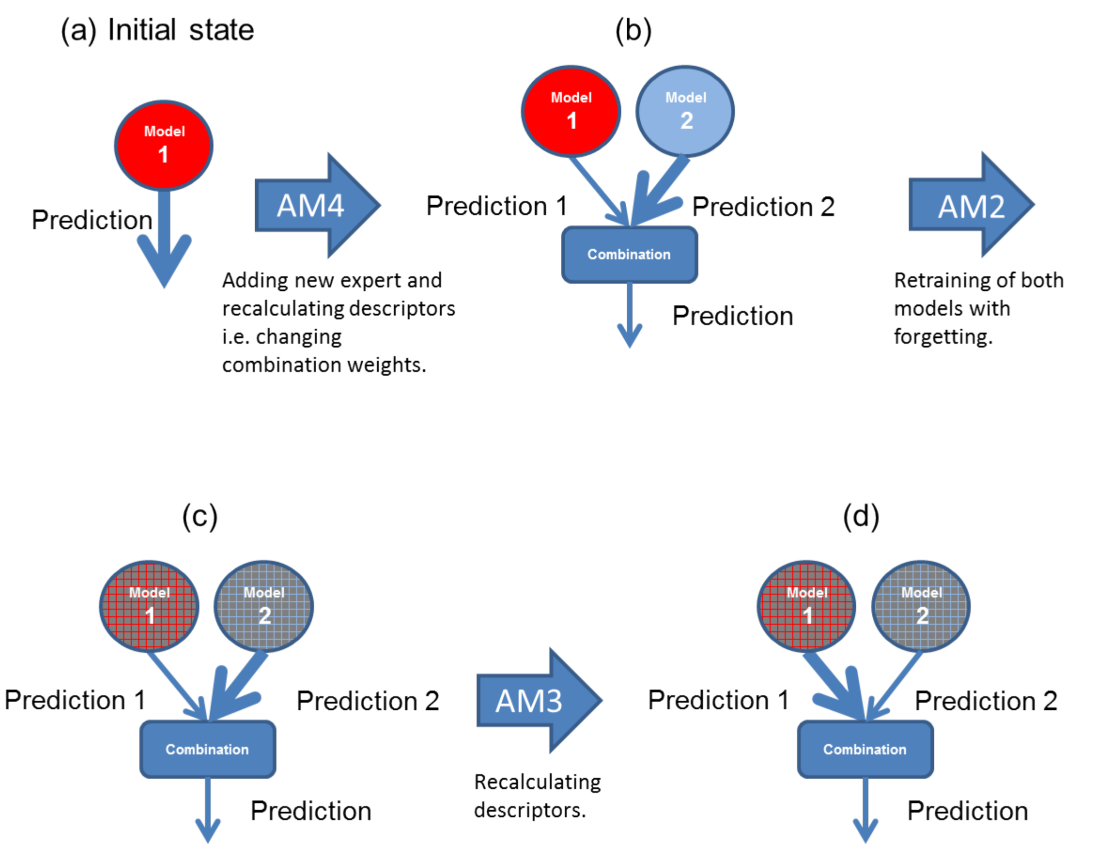


Figure 4.5: *An example of a model adaptation sequence (AM4, AM2, AM3) using SABLE adaptive mechanisms. Here, the weight of the expert corresponds to the thickness of the arrow leading from respective circle.*

4.5 Experiments

4.5.1 Experimental setup

At every batch $\mathcal{V}_k$, an AM g_{h_k} must be chosen to deploy on the current batch of data. The goal of this chapter is to analyze whether the choice of AMs significantly affects the predictive performance of the model. To this end, SABLE was run on each of the datasets with the randomly selected AM deployed on each batch of data, 1000 times. In other words, 1000 random AM sequences were deployed on each dataset. The results were then compared with true values and the benchmark performance.

To obtain a benchmark performance, a *greedy optimal* adaptation strategy[7]

$$f_{k+1}^{-} = f_k^{-} \circ g_{h_k}, \quad h_k = \underset{h_k \in 1 \cdots H}{\operatorname{argmin}} \langle (f_k^{-} \circ g_{h_k})(\boldsymbol{X}_{k+1}), \boldsymbol{y}_{k+1} \rangle \tag{4.11}$$

[7] In this research, the methods of AM selection are called *adaptive strategies*. More strategies will be given in the Chapter 5.

was used, where $\langle\ \rangle$ denotes the chosen error measure[8], f_k^- is the *a priori* predictive function at k-th batch and g_{h_k} is an adaptive mechanism. Since $\boldsymbol{X}_{k+1}, \boldsymbol{y}_{k+1}$ are not yet obtained, this strategy is not applicable in the real life situations. This strategy is referred to as *Optimal*. Note that because of the greedy approach, this may not be the overall optimal strategy which minimizes the error over the whole dataset.

SABLE hyperparameters for experiments on each dataset are listed in the Table 4.1. They were identified using a grid search among the set of hyperparameter candidates.

Dataset	δ_0, δ_1	λ	σ	L
Catalyst50	0, 1	0.5	1	12
Catalyst100	0, 1	0.25	1	12
Catalyst200	0, 1	0.5	1	12
Oxidizer50	0.25, 0.75	0.5	1	3
Oxidizer100	0, 1	0.25	0.01	3
Oxidizer200	0, 1	0.25	0.01	3
Drier50	0, 1	0.25	0.01	16
Drier100	0, 1	0.5	0.1	16
Drier200	0, 1	0.25	0.01	16

Table 4.1: *SABLE parameters for different datasets. Here, δ_0, δ_1 are update weights of descriptors, λ is RPLS forgetting factor, σ is kernel width for descriptor construction and L is the number of RPLS latent variables.*

4.5.2 Results

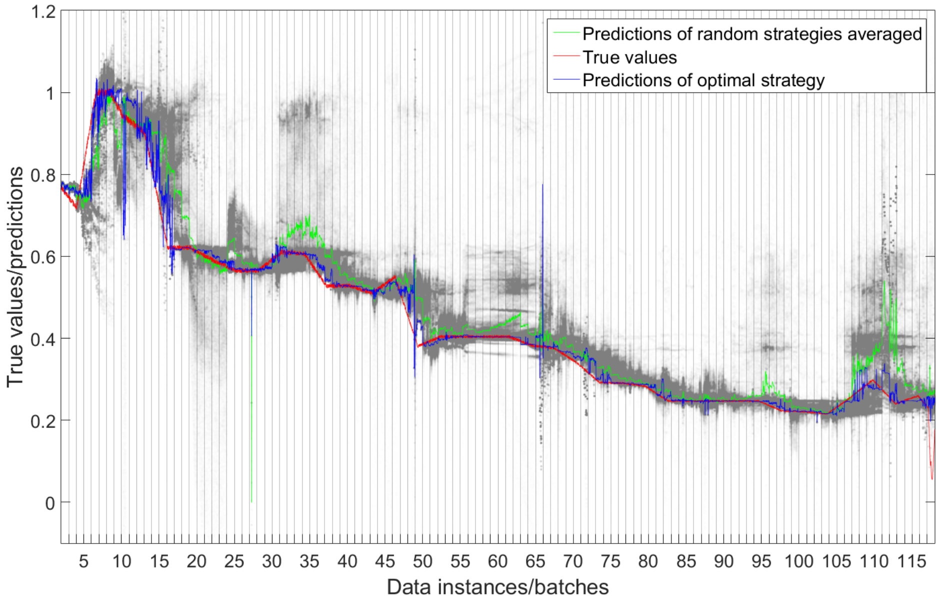
The results of 1000 random runs for each of the datasets are shown in Figures 4.6 - 4.8. Here, the true/predicted values are visualised for the Catalyst and Oxidizer datasets. For Drier dataset, since the differences between the predictions in each run are minimal in comparison to the differences in the output, which makes the different runs impossible to distinguish, error was visualised instead. If the AM sequence choice indeed is an important factor for the predictive accuracy, the results obtained from the optimal choice of AM sequence, must be better than those from the most of the random AM sequences.

Figure 4.6[9] shows the predictions of models resulting from deploying 1000 random adaptation sequences while predicting on Catalyst dataset. When comparing the predictions of the random runs (gray dots) to the predictions of the optimal sequence (blue line), it is easy to see that the prediction accuracy while deploying a random AM sequence can be much lower than when using the optimal sequence. In fact, several different behaviours of the predictive models can be identified. For instance, if the Catalyst100 (Figure 4.6(b)) is considered, it is possible to see that, at some batches, such as #2-#10, #28, #56-#58, optimal AM choice results in drastically more

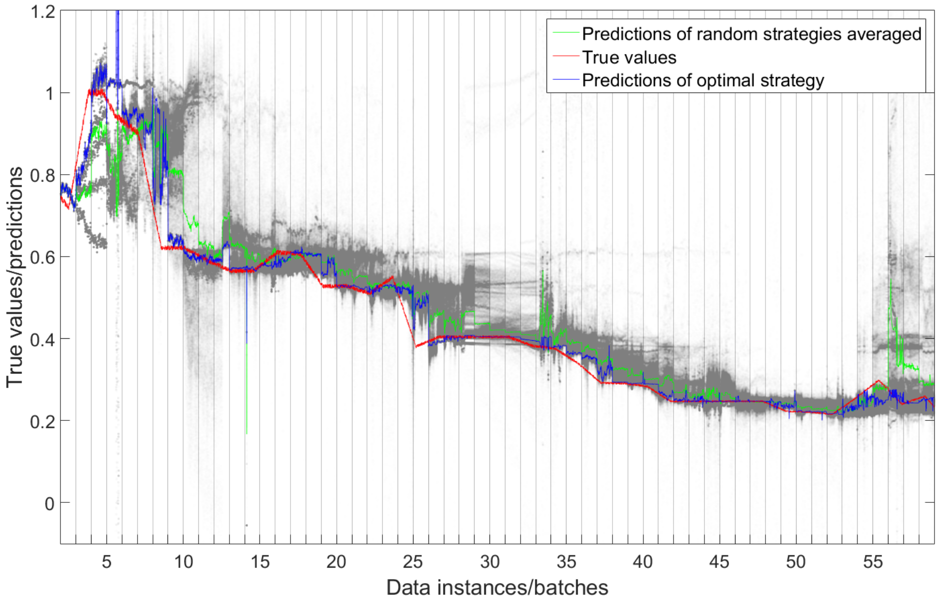
[8] Throughout this thesis, Mean Absolute Error (MAE) is used for this purpose.

[9] Figures 4.6 - 4.8 show plots of the results starting from the second batch, as the first batch is used for training of the initial expert and no predictions on it are made.

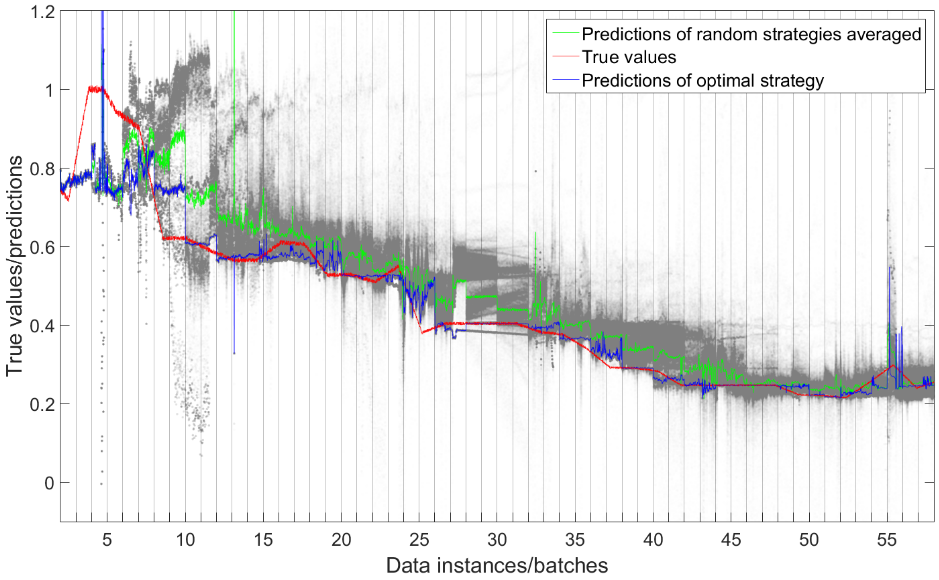
accurate predictions than most of the random runs. In the other areas, such as batches #16-#18, optimal AM sequence seem to result in similar predictions as some of the random runs, however the variance of predictions after different runs is fairly high. It is also possible to notice that the optimal AM sequence results in much faster adaptation than most of the random sequences, which is visible when observing the green line (averaged predictions after the random sequences) in the area between batches #35 and #44. In rare cases such as batch #18, there might be a random sequence which predicts more accurately than the optimal sequence. Catalyst50 and Catalyst200 (Figure 4.6(a) and (c)) show similar behaviour. It is interesting to note that some batches, such as batches around #61 and around #108 for Catalyst50, batch #10 for Ctalyst100 and batch #6 for Catalyst200 exhibit a noticeable bivariate distribution. Comparing to the results on higher batch sizes, results of random runs on Catalyst50 are less robust, exhibiting high variance in more batches. The possible cause of this behaviour is that there is less training data available for new experts, which may lead to relatively untrained experts with a high weight, which in turn have higher error on the subsequent batches. It is also possible to notice that the adaptive methods perform worse on Catalyst200 – this is based on the fact that the adaptation in this case is simply less frequent.



(a) *Catalyst50*



(b) *Catalyst100*



(c) *Catalyst200*

Figure 4.6: *True/predicted values for Catalyst datasets. Predictions of random runs are visualised with gray points.*

Comparison of the minimum and maximum MAE values with RMSE values of corresponding sequences after the random runs on the Catalyst dataset, MAE/RMSE of averaged predictions, MAE/RMSE of a static model with no adaptation (*Sequence0*), MAE/RMSE of simple relearning from scratch on every batch (*Retrain*) and MAE/RMSE after the optimal sequence (*Optimal*) are

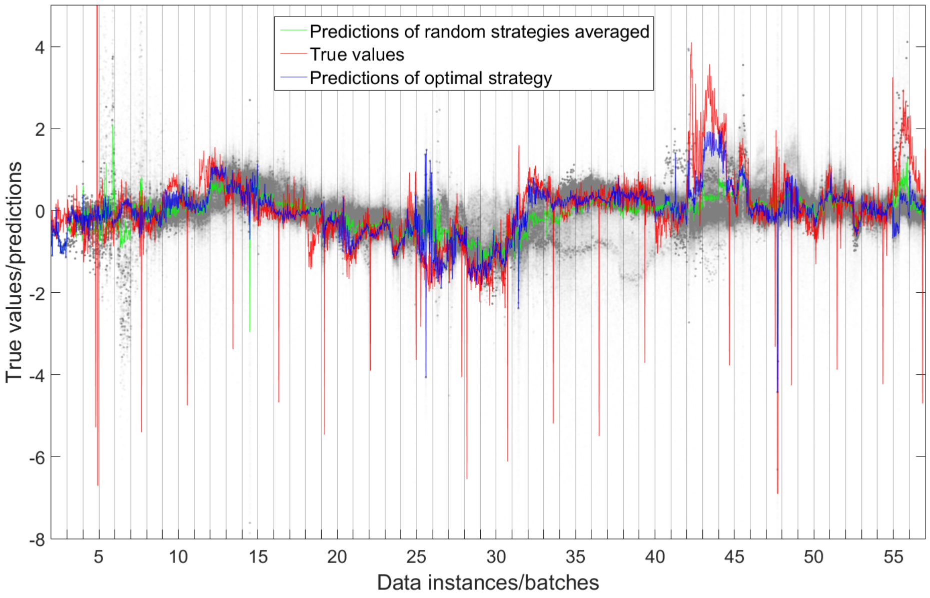
	Catalyst50		**Catalyst100**		**Catalyst200**	
	MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Mean Random1000	**0.0385**	0.0591	**0.0480**	0.0701	**0.0769**	0.1107
Min. Random1000	**0.0236**	0.0424	**0.0337**	0.0528	**0.0480**	0.0792
Max. Random1000	**0.1062**	0.2045	**0.1872**	0.2828	**0.3122**	0.4237
Sequence0	**0.3098**	0.3422	**0.2787**	0.3130	**0.3612**	0.3972
Retrain	**0.0241**	0.0814	**0.0278**	0.0583	**0.0516**	0.1076
Optimal	*0.0149*	*0.0306*	*0.0233*	*0.0467*	*0.0403*	*0.0691*

Table 4.2: *Results of deploying random AM sequences (Random1000) on Catalyst dataset averaged over all batches. MAE values of predictions significantly (according to (Mizrach, 1996), Section 3.2 with $\alpha \leq 0.05$) different from Optimal predictions are marked bold.*

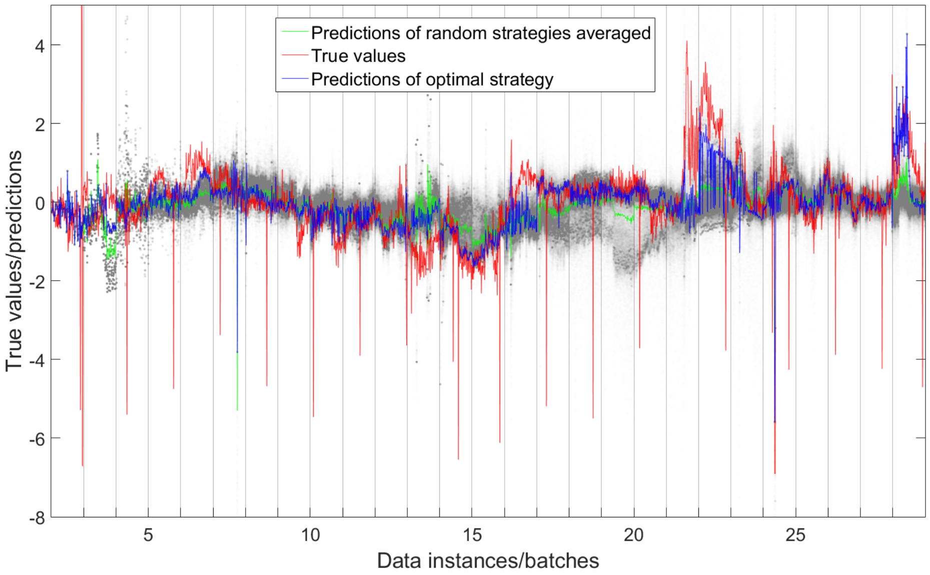
shown in Table 4.2. Significance values of differences between the compared predictions from the predictions of optimal AM sequence, were calculated[10]. If they were different with significance level of $\alpha \leq 0.05$, MAE value in appropriate cell is marked with bold (this also applies to Tables 4.3 and 4.4). The *Optimal* AM sequence provides the smallest error with its predictions significantly different from the rest for every batch size. The differences between minimum and maximum error values from random AM sequences are of several orders of magnitude. Mean error values after the random sequences are higher than those of simple retraining. Finally, even the maximum error values after random sequences are considerably lower than those from the static model.

Figure 4.7 shows the predictions of models resulting from deploying 1000 random adaptation sequences while predicting on Oxidizer dataset. As this dataset is less susceptible to changes, the random AM sequences result in a distribution which is mostly centered on true values, e.g. the batches #6-#12 in Figure 4.7(b). In this figure, there are still several batches, where the Optimal AM sequence results in the predictions are much closer to the true values than most of the random sequences, e.g. batches #17, #26. On some batches, for example #14-#15, #22, #28 it is possible to see that the choice of the *Optimal* AM sequence results in a faster adaptation to the possible changes in comparison to the random sequences. Similar conclusions can be drawn from Figures 4.7(a) and 4.7(c). Bivariate distribution of errors is not noticeable in for this dataset, except slightly in batch #19 for Oxidizer100.

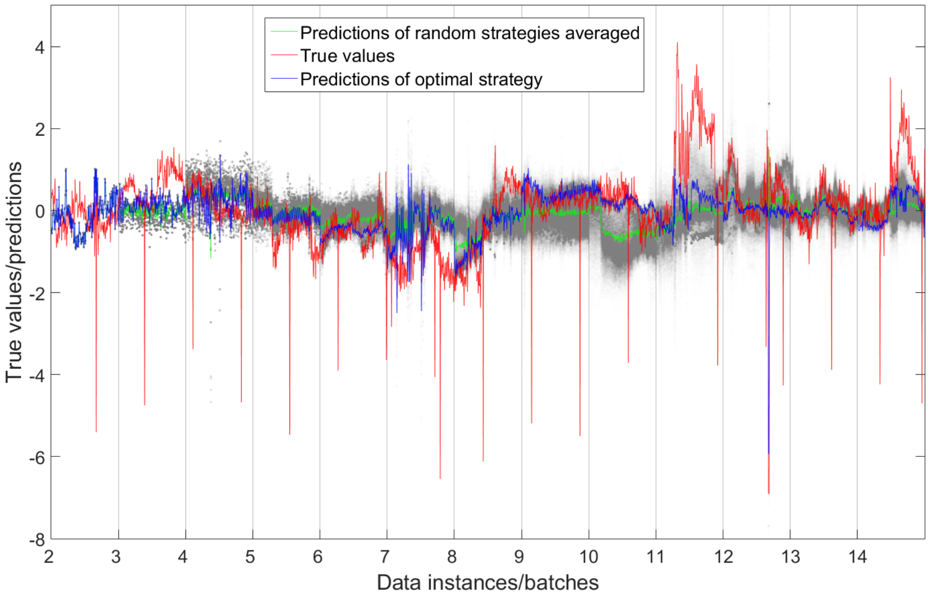
[10] To calculate the significance of differences between the predictions of different strategies here and in Chapter 5, the significance test of difference of two estimators' errors relying on the sample covariance ((Mizrach, 1996), Section 3.2) was used.



(a) *Oxidizer50*



(b) *Oxidizer100*



(c) *Oxidizer200*

Figure 4.7: *True/predicted values for Oxidizer dataset. Predictions of random runs are visualised with gray points.*

Comparison of the minimum and maximum MAE values with RMSE values of corresponding sequences after the random runs on the Oxidizer dataset, MAE/RMSE of averaged predictions, MAE/RMSE of a static model with no adaptation (*Sequence0*), MAE/RMSE of simple relearning from scratch on every batch (*Retrain*) and MAE/RMSE after the optimal sequence (*Optimal*) are shown in Table 4.3. In addition, the errors of the static model and of simple relearning from the scratch on every batch are shown. Similar to the results on the previous dataset, the *Optimal* AM sequence provides the smallest error for every batch size with most of the differences being significant. The differences between minimum and maximum error values from random AM sequences are still very noticeable, but less drastic. Mean error values after the random sequences are lower than those of simple retraining for Oxidizer100 and Oxidizer200. This is an indicator that discarding old models for these datasets is detrimental and that using multiple AMs is indeed more beneficial. Consequently for every batch size, the minimum error values after random AM sequences are smaller than after simple retraining. For this dataset, the maximum error values after random sequences are considerably larger than those from the static model - this is likely connected with the fact that there are less changes in the Oxidizer dataset, so the choice of wrong AMs may result in an “over-adaptation” effect.

Figure 4.8 shows the prediction errors of the models resulting from deploying 1000 random adaptation sequences for the Drier dataset. It is easy to see that while most of the errors of random AM sequences are centered around 0, some unfortunate AM choices result in large error bias. For example, for Drier100 from batch #4 onwards, this bias is observed, also resulting in a bivariate

	Oxidizer50		**Oxidizer100**		**Oxidizer200**	
	MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Mean Random1000	**0.515**	0.864	**0.561**	0.906	**0.609**	0.911
Min. Random1000	**0.491**	0.854	**0.549**	1.023	**0.555**	0.855
Max. Random1000	**1.287**	2.026	**0.925**	1.459	**0.871**	1.216
Sequence0	**0.760**	1.181	0.779	1.218	0.783	1.130
Retrain	0.499	0.854	**0.565**	0.944	**0.678**	1.063
Optimal	*0.416*	*0.766*	*0.481*	*0.822*	*0.530*	*0.804*

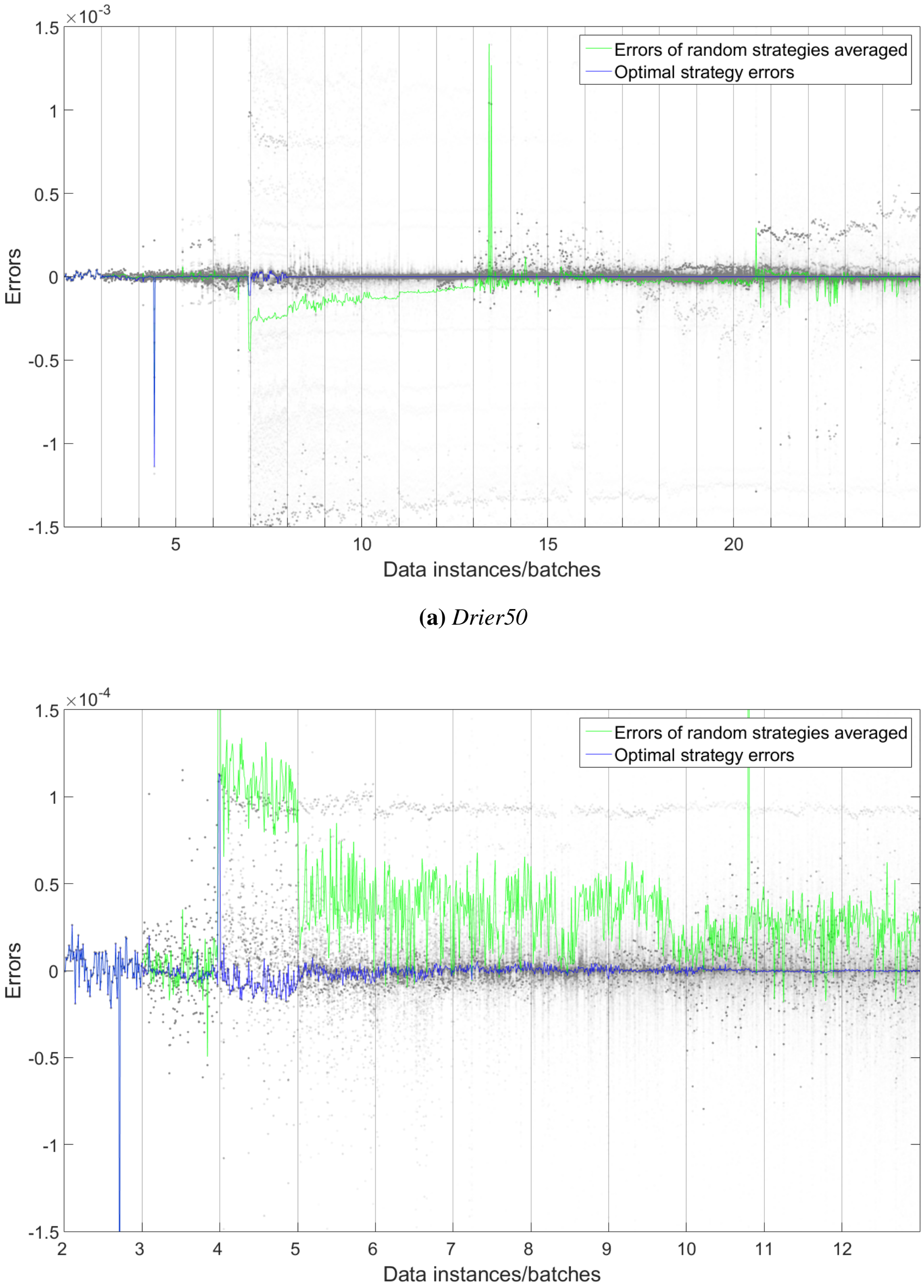
Table 4.3: *Results of deploying random AM sequences (Random1000) on Oxidizer dataset averaged over all batches. MAE values of predictions significantly (according to (Mizrach, 1996), Section 3.2 with $\alpha \leq 0.05$) different from Optimal predictions are marked bold.*

distribution. It is observed that deployment of further AMs on subsequent batches is required, to reduce the errors to stabilize them around 0. The similar behaviour is observed for Drier200.

	Drier50		**Drier100**		**Drier200**	
	MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Mean Random1000	**5.71E-05**	1.07E-04	**3.50E-05**	5.14E-05	**1.11E-04**	1.73E-04
Min. Random1000	9.46E-06	1.64E-05	**3.42E-06**	1.18E-05	4.67E-05	1.43E-04
Max. Random1000	**2.64E-03**	2.98E-03	**1.07E-03**	4.96E-04	**3.87E-04**	4.07E-04
Sequence0	**7.68E-04**	8.67E-04	**5.41E-04**	5.97E-04	**3.87E-04**	4.07E-04
Retrain	**5.86E-05**	3.14E-04	**2.59E-05**	1.54E-04	**5.38E-05**	1.44E-04
Optimal	*3.40E-06*	*3.47E-05*	*3.15E-06*	*1.15E-05*	*4.67E-05*	*1.43E-04*

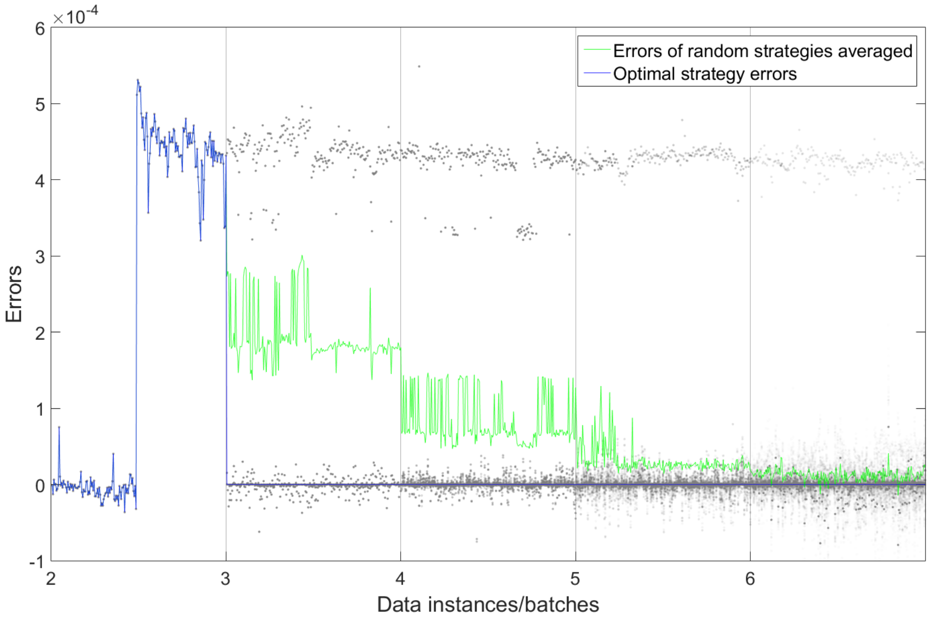
Table 4.4: *Results of deploying random AM sequences (Random1000) on Drier dataset averaged over all batches. MAE values of predictions significantly (according to (Mizrach, 1996), Section 3.2 with $\alpha \leq 0.05$) different from Optimal predictions are marked bold.*

Comparison of the minimum and maximum MAE values with RMSE values of corresponding sequences after the random runs on the Drier dataset, MAE/RMSE of averaged predictions, MAE/RMSE of a static model with no adaptation (*Sequence0*), MAE/RMSE of simple relearning from scratch on every batch (*Retrain*) and MAE/RMSE after the optimal sequence (*Optimal*) are shown in Table 4.4. The *Optimal* AM sequence provides the smallest error with its predictions with most of the differences being significant. As in Oxidizer dataset, for every batch size, the minimum error values after random AM sequences are smaller than after simple retraining. Since the number of deployed AMs in one sequence for Drier200 is only 5, the optimal AM sequence is also deployed among them, so the minimum MAE/RMSE after random AM runs are equal to MAE/RMSE after *Optimal* sequence. Moreover, values for the maximum MAE after the random runs is the same as the one after using a model with no adaptation. It is indeed obtained after running a sequence of AMs which contains only AM0, which means no adaptation.



(a) *Drier50*

(b) *Drier100*



(c) *Drier200*

Figure 4.8: *Error values for Drier dataset. Predictions of random runs are visualised with gray points.*

4.5.3 Discussion

This chapter was focused on investigation of whether the choice of AM sequence influences the predictive accuracy of the model. For this purpose, after the presentation of the necessary formulation in the Section 4.2, an adaptive batch ensemble method, SABLE was introduced in Section 4.3. SABLE can adapt the predictive model both parametrically and structurally by retraining the experts with or without forgetting, updating weights of the experts' contributions, adding new experts and pruning of similar ones, which are at all the levels of ensemble adaptation listed in the Section 2.5. Section 4.4 lists the proposed adaptive mechanisms which are used in the experiments.

To analyze the influence of AM choice, 1000 runs of SABLE with random adaptation sequences were conducted. The results of the experiments are presented in the Section 4.5.2. Analysing the results, several conclusions can be drawn:

- AM choice clearly has a great influence on all of the considered datasets and batch size choices.
- Depending on the dataset, static model or the simple retraining can show worse or better results than random AM sequence. Rapidly changing Catalyst dataset benefits from any random AM sequence which was deployed during the experiment comparing with the static model. However, the simple retraining performed better than averaged predictions after random AM sequences.

- For Oxidizer and Drier datasets, it is seen that not every random AM sequence is as good as the static model. This shows the importance of AM selection as the unfortunate choices might worsen the predictive accuracy instead of improving it. On the other hand, for these datasets averaged predictions after random AM sequences perform better than simple retraining except on Oxidizer50 dataset. This shows the benefits of model's adaptation over retraining and the usefulness of multiple adaptive mechanisms' availability.
- In all of the cases, optimal sequence results in better performance than simple retraining with almost always significant difference in error values. Most of the times, the minimum error of the random runs was better than simple retraining as well. This fact reiterates the benefits of using multiple adaptive mechanisms and confirms the necessity of intelligent AM selection strategies.

It can be thus concluded that the choice of AM sequence plays a decisive role for the methods with multiple adaptive mechanisms. This raises an important question of the method of AM sequence selection. This question is addressed in Chapter 5 of this work, where methods of AM sequence selection or generation are presented.

Chapter 5

Adaptive strategies

5.1 Introduction

In the previous chapter of the thesis, it was shown that the selection of an AM sequence can play an important role in the predictive accuracy of a model on non-stationary streaming data. This raises the questions about which AMs should be selected for deployment and more generally about what strategy should/can be used to choose the AM sequence in order to minimize some cost function (here the prediction error). In this chapter, an *adaptive strategy* is a strategy which identifies the AMs for adaptation on incoming data, and together with the data itself fully specifies the subsequent AM sequence.

To answer these questions, methods to analyse AMs' effects that help understanding the weaknesses and strengths of every AM are suggested in Section 5.3. SABLE AMs are explored in this fashion and it is observed that AM4 performs better than other available AMs in most of the situations.

Subsequently in Section 5.4 it is proceeded to analyse the performance of common strategies which generate fixed AM sequences (i.e. fixed adaptive strategies) and introduce novel strategies which generate flexible AM sequences (i.e. flexible adaptive strategies) based on cross-validatory selection and retrospective reversal of model into the optimal state. Using SABLE as a vehicle, it is shown empirically that in the majority of cases, a flexible adaptive strategy is superior to a fixed one.

In the previous chapter and Section 5.4 it has been shown that the greedy optimal adaptive strategy shows the best results for all datasets. In Section 5.5 it is attempted to predict the adaptation order generated by the optimal strategy using a meta-classifier. It is empirically shown that a certain level of classification accuracy must be attained for this meta-classifier to deliver an AM sequence which would achieve higher accuracy levels than those in Section 5.4. However, it was not possible to achieve these required levels of accuracy in practice and therefore this aspect remains an open subject of further research.

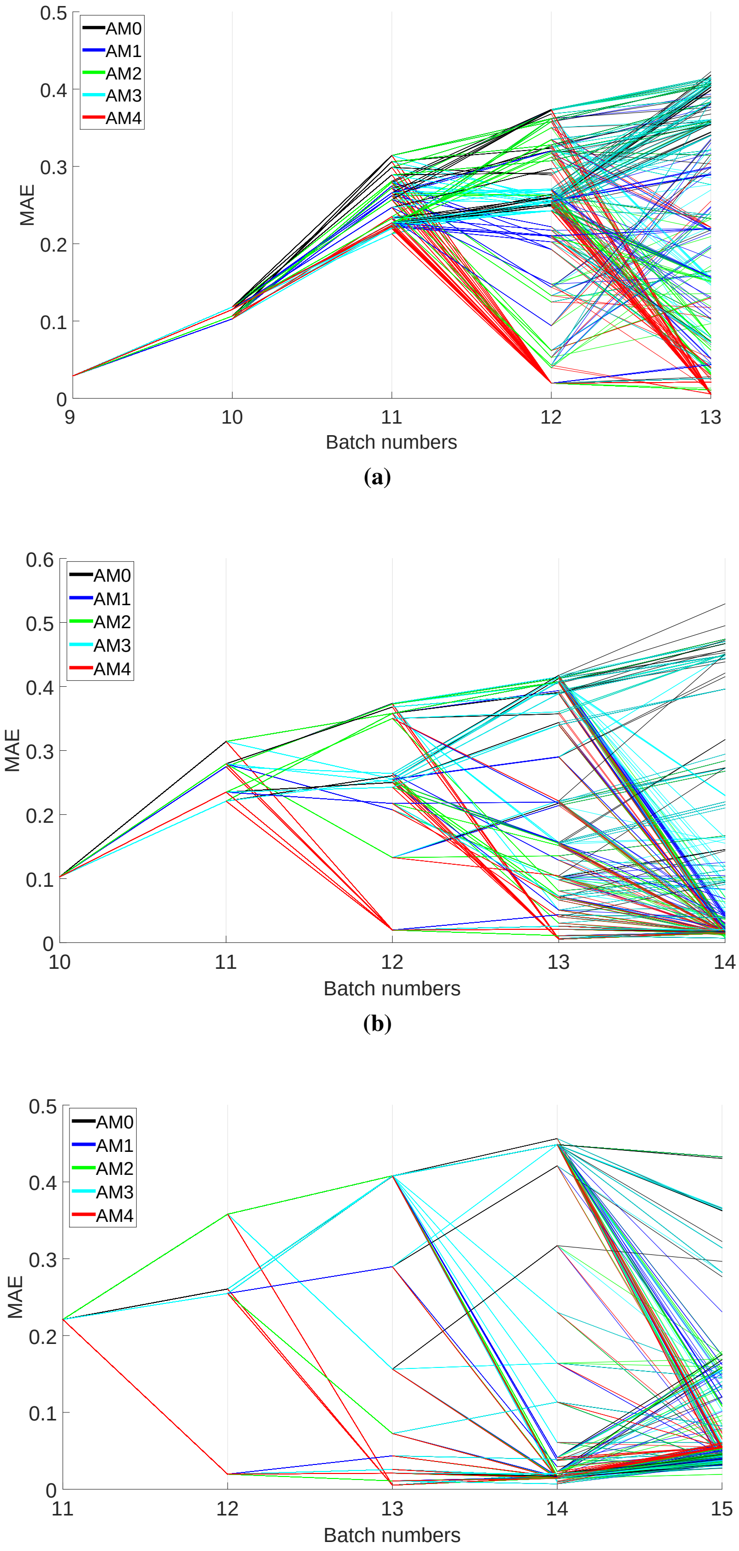
A large part of analysis in this chapter required a substantial amount of data regarding AMs' performance, which was generated using the exhaustive r-step ahead AM deployment technique described in Section 5.2.

5.2 Exhaustive r-step ahead adaptive mechanism deployment

Data on AMs' performance in different situations is required to make generalisations about AMs' behaviour. To generate a substantial amount of this data, a limited r-step ahead exhaustive deployment of all AMs has been implemented. Assume that there are H available AMs, $G = \{g_1, \cdots g_H\}$. At every batch all of these AMs are deployed separately on current predictive model f. The resulting models, $\{f \circ g_1, \cdots, f \circ g_H\}$ are used to predict the values of the next batch, errors are recorded and then these models are adapted using all AMs in G for each of them, etc. for r batches ahead. After this, the model which gives the best performance on the original batch was chosen as the starting point for the next four steps. This process is described in Algorithm 3.

The r-step ahead AM deployment strategy has been applied to SABLE while predicting on the datasets introduced in Chapter 3. For practical reasons discussed below, $r = 4$ was selected. SABLE has 5 AMs (AM0, AM1, AM2, AM3, AM4), therefore $H = 5$. Thus, as seen in Figure 5.1, for every batch a tree of height 4, where all non-leave nodes have 5 children, is generated. The paths from the root of the tree to its leaves represent an exhaustive list of the last 4 AMs (e.g. AM0, AM0, AM0, AM0; AM0, AM0, AM0, AM1;...) deployed to arrive at the updated models providing current prediction. On the leaves of each tree, $5^4 = 625$ forecasts and associated statistics such as MAE are obtained. The data resulting from this limited exhaustive AM deployment was then used for experiments in the following sections.

Note that a limited exhaustive search is used as the number of prediction models grows exponentially (specifically as a power of the number of batches), thus rendering a full exhaustive search nearly impossible, particularly for large datasets with small batch sizes. If a dataset has K batches, for H AMs a full exhaustive deployment tree will have $H^{(K-2)}$ leaves and will require $H^1 + H^2 + \cdots + H^{K-2} = \frac{1-H^{K-1}}{1-H} - 1$ predictions and adaptations (the first batch is used for initial training of the model, from the rest $K - 1$ batches a tree with the height of $K - 2$ can be constructed). For instance, the full tree exhaustive search for the Catalyst50 dataset (117 batches) and $H = 5$ has 2.407×10^{80} leaves and requires 3.009×10^{80} predictions and adaptations. In comparison, for a dataset of K batches, the combined number of leaves for all the trees resulting from applying limited r-step ahead exhaustive AM deployment is $(K-1-r) \times H^r$ (the first batch is used for initial training of the model, the last r batches are not used to generate trees as there are no subsequent batches to evaluate) and the number of required adaptations and predictions is $(K - 1 - r) \times (\frac{1-H^{r+1}}{1-H} - 1)$. For the Catalyst50 with $H = 5$ these are $70,000$ and $87,360$ respectively, making it possible to execute this strategy and operate with the resulting data. In addition, poor adaptation sequences would typically be pruned when detected, so the information generated during those may not be interesting for practical purposes. Thus, limiting the deploy-



(a)

(b)

(c)

Figure 5.1: *Exhaustive 4-step ahead AM deployment on Catalyst100 (a) batch #9, (b) batch #10, (c) batch #11.*

Algorithm 3 Pseudocode of exhaustive r-step ahead AM deployment.

```
function $Limited\_Exhaustive\_Deployment(r, \{\{\boldsymbol{X}_1, \boldsymbol{y}_1\}, \cdots, \{\boldsymbol{X}_{K-r}, \boldsymbol{y}_{K-r}\}\}, G)$
Runs the exhaustive $r$-step ahead AM deployment on the selected dataset, storing the results.
Inputs:
$r$: number of steps to exhaustively deploy all AMs
$\{\{\boldsymbol{X}_1, \boldsymbol{y}_1\}, \cdots, \{\boldsymbol{X}_{K-r}, \boldsymbol{y}_{K-r}\}\}$: input and target data in batches
$G = \{g_1, \cdots g_H\}$: set of available AMs

Build initial training model, $f_1 = f_2^-$ from $\{\boldsymbol{X}_1, \boldsymbol{y}_1\}$
for $k = 2, \cdots, K-r$ do
    $[\hat{\boldsymbol{y}}_k, \{\hat{\boldsymbol{Y}}_{k+1}, \cdots, \hat{\boldsymbol{Y}}_{k+r}\}, \boldsymbol{f}_k^+] =$
        $= AM\_Deployment\_Tree(f_k^-, \{\{\boldsymbol{X}_k, \boldsymbol{y}_k\}, \cdots, \{\boldsymbol{X}_{k+r}, \boldsymbol{y}_{k+r}\}\}, G)$,
        where $\hat{\boldsymbol{y}}_k = f_k^-(\boldsymbol{X}_k)$, $\{\hat{\boldsymbol{Y}}_{k+1}, \cdots, \hat{\boldsymbol{Y}}_{k+r}\}$ are predictions of the models along
        the AM deployment tree on subsequent batches and $\boldsymbol{f}_k^+$ is a set of adapted models
    Store $[\hat{\boldsymbol{y}}_k, \{\hat{\boldsymbol{Y}}_{k+1}, \cdots, \hat{\boldsymbol{Y}}_{k+r}\}, \boldsymbol{f}_k^+]$
    $f_{k+1}^- = f_k^- \circ g_h,\ h = \underset{h \in 1 \cdots H}{argmin} \langle \hat{\boldsymbol{y}}_{k+1}^h, \boldsymbol{y}_{k+1} \rangle$
end for
```

```
$[\hat{\boldsymbol{y}}_a, \{\hat{\boldsymbol{Y}}_{a+1}, \cdots, \hat{\boldsymbol{Y}}_b\}, \boldsymbol{f}_a^+] =$
    function $AM\_Deployment\_Tree(f_a^-, \{\{\boldsymbol{X}_a, \boldsymbol{y}_a\}, \cdots, \{\boldsymbol{X}_b, \boldsymbol{y}_b\}\}, G)$
Generates a single AM deployment tree of the height $b - a$.
Inputs:
$f_a^-$: current predictive model
$\{\{\boldsymbol{X}_a, \boldsymbol{y}_a\}, \cdots, \{\boldsymbol{X}_b, \boldsymbol{y}_b\}\}$: input and target values in batches
$G = \{g_1, \cdots, g_H\}$: set of available AMs
Outputs:
$\hat{\boldsymbol{y}}_a$: predictions of $f_a^-$ on $\boldsymbol{X}_a$
$\hat{\boldsymbol{Y}}_{a+1}, \cdots, \hat{\boldsymbol{Y}}_b$: sets of adapted models' predictions on batches $\boldsymbol{X}_{a+1}, \cdots, \boldsymbol{X}_b$
$\boldsymbol{f}_a^+$: set of predictive models resulting from deploying $\{g_1, \cdots g_H\}$ on $f_a^-$

Predict $\hat{\boldsymbol{y}}_a = f_a^-(\boldsymbol{X}_a)$, store $\hat{\boldsymbol{y}}_a$
if $a < b$ then
    for $h = 1, \cdots, H$ do
        Adapt prediction function, $f_{a+1}^- = f_a^+ = g_h(\boldsymbol{X}_{a+1}, \boldsymbol{y}_{a+1}, \Theta_g, f_a^-, \hat{\boldsymbol{y}}_a) \circ f_a^-$
        $[\hat{\boldsymbol{y}}_{a+1}^h, \{\hat{\boldsymbol{Y}}_{a+2}^h, \cdots, \hat{\boldsymbol{Y}}_b^h\}, \boldsymbol{f}_{a+1}^{+h}] =$
            $= AM\_Deployment\_Tree(f_{a+1}^-, \{\{\boldsymbol{X}_{a+1}, \boldsymbol{y}_{a+1}\}, \cdots, \{\boldsymbol{X}_b, \boldsymbol{y}_b\}\}, G)$
    end for
    $\hat{\boldsymbol{Y}}_{a+1} \leftarrow$ set of all $\hat{\boldsymbol{y}}_{a+1}$ from different tree branches
end if
```

ment to 4-steps ahead was chosen as a practical compromise. The generated data from applying 4-steps AM deployment is shown in Table 5.1.

Dataset	Number of generated leaves	Number of predicted instances per AM	Total number of predicted instances
Catalyst50	70,000	700,000	3,500,000
Catalyst100	33,125	662,500	3,312,500
Catalyst200	15,000	600,000	3,000,000
Oxidizer50	31,875	318,750	1,593,750
Oxidizer100	14,375	287,500	1,437,500
Oxidizer200	5,625	225,000	1,125,000
Drier50	11875	118,750	593,750
Drier100	4375	87,500	437,500
Drier200	625	25,000	125,000

Table 5.1: *Data generated using exhaustive r-step ahead AM deployment.*

5.3 Analysis of adaptive mechanisms' effects

It is evident that the effects of different AMs (in case of SABLE; retraining without forgetting (AM1), retraining with forgetting (AM2), weights recalculation (AM3), and addition of experts (AM4)) can be quite different. For SABLE, the difference between retraining with forgetting as compared to the retraining without forgetting depends directly on the forgetting factor. It is more difficult to quantify the differences between retraining, weights recalculation and creation of new experts, as this is highly dependant on the data and parametrization. Therefore, to gain insight about the relative behaviour of the AMs, the speed/strength of adaptation in addition to the observed profit (or loss) are investigated.

Important factors to consider while analysing the AMs are how strongly the AM can adapt the model, and how accurate the model is after adaptation. Both of these can be measured using the predictions of the adapted models. In this thesis a novel metric for this purpose,

$$RA = \frac{f_0(\boldsymbol{x}) - f_i(\boldsymbol{x})}{f_0(\boldsymbol{x}) - y}, \tag{5.1}$$

where for the data instance $(\boldsymbol{x}, y)$, y is the observed value, $f_0(\boldsymbol{x})$ is the prediction of original model and $f_i(\boldsymbol{x})$ is the prediction after the deployment of the AM $g_i(\cdot)$, is proposed. The numerator shows the difference in the predictions of adapted and unadapted model, that is the *strength of adaptation*. The denominator shows the difference in the prediction of unadapted model and the true value, in other words, the *need for adaptation*. This metric will be referred to as the Relative Adaptation (RA). RA has the following interesting properties:

- $\mathbf{RA} = \mathbf{0}$ means that the adaptation has no effect,
- $\mathbf{RA} = \mathbf{1}$ means that the adaptation is perfect for the particular data instance, i.e. $f_i(\boldsymbol{x}) = y$,
- $\mathbf{RA} = \mathbf{2}$ means that the error has switched its sign, i.e. $f_i(\boldsymbol{x}) - y = -(f_0(\boldsymbol{x}) - y)$,
- $\mathbf{0} < \mathbf{RA} < \mathbf{1}$ means that the adaptation has decreased the error, while the sign of error remains the same. This behavior can be also called under-adaptation, as the adaptation

does correct the model in the right direction but not strongly enough,

- $\mathbf{1 < RA < 2}$ means that the adaptation has decreased the error and has changed the error sign. This behavior can be also called over-adaptation, because the adaptation is correcting the model in the right direction but too strongly.
- $\mathbf{RA < 0}$ means that the error has increased and its sign remains the same, and
- $\mathbf{RA > 2}$ means that the error has increased and its sign is changed. This can be also considered over-adaptation, as the adaptation is correcting the model in the right direction.

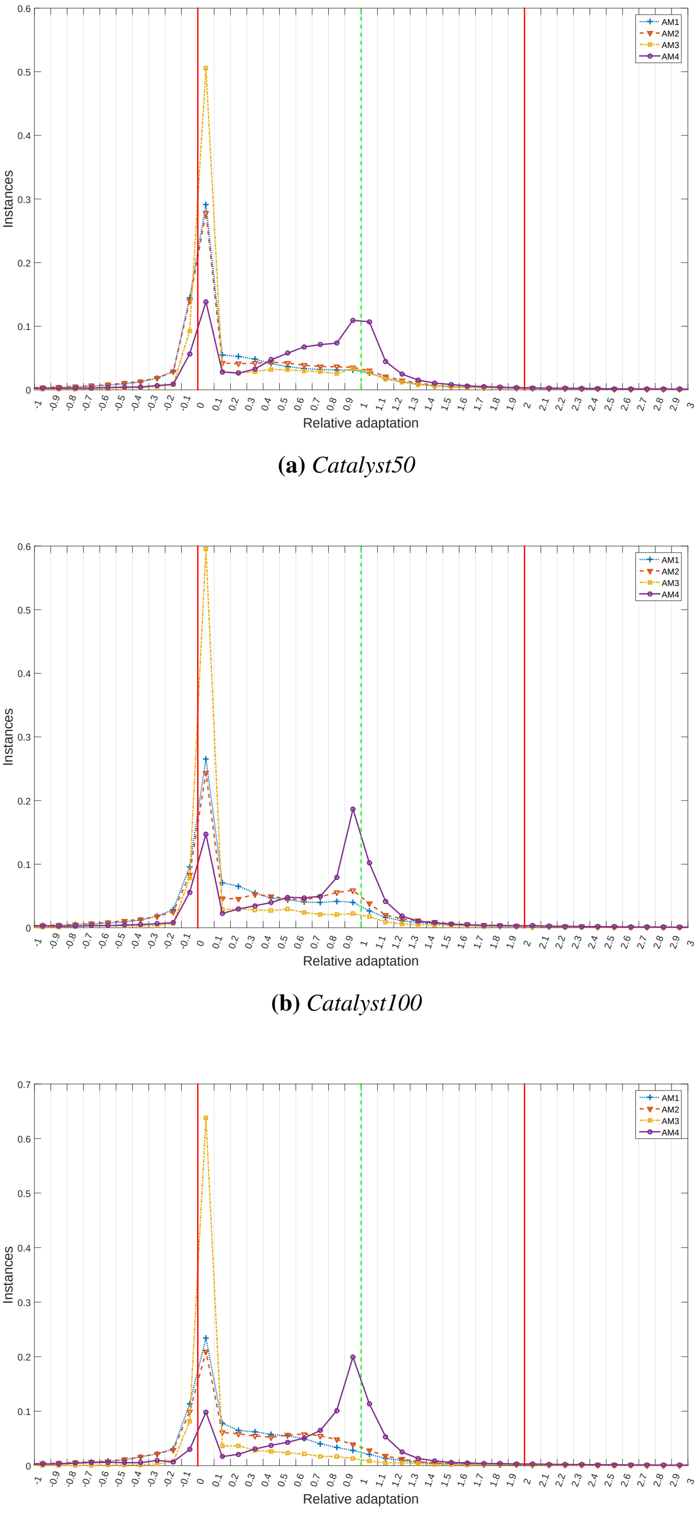
Figure 5.2 presents the normalised relative adaptation histograms for the Catalyst dataset. These and following histograms in this section are obtained using the data generated as shown in Section 5.2. Considering Figure 5.2(a), it can be noted that all of the AMs have a peak at $[0; 0.1)$ interval. AM1, AM2 and AM3 show first significant drop and then gradual decrease when moving away from the peak. AM4 has additional peaks at $[0.9; 1)$ and $[1; 1.1)$ intervals. It can be concluded that AM1, AM2 and AM3 mostly result in minimal adaptation in correct direction[1]. AM4 histogram shows a rise towards RA=1, signifying a considerable improvement in predictive accuracy that it provides. This may be explained by the fact that there are stable periods after the changes in Catalyst dataset, which are relatively easy to predict for the newly created experts.

All three batch sizes exhibit similar behaviour. As the batch size becomes larger, the peaks at $[0; 0.1)$ slightly decrease. This is a natural consequence of having more data for the adaptation - since, as established in Section 3.5, Catalyst dataset changes relatively fast, the larger the batches, the more different they are, which in turn promotes stronger adaptation. Another difference is that AM4 peak at $[0.9; 1)$ rises. This can be explained by more accurate experts which are trained on more data. Although the histograms on this figure, as well as Figures 5.3 and 5.4 were calculated on a sample data, the sample is considered representative for this purpose, for the reasons given in Section 5.2. Bootstrapping based confidence levels of selected histograms, shown in Appendix B, confirm this consideration.

Figure 5.3 presents the relative adaptation histograms for the Oxidizer dataset. As in the previous dataset, all AMs show a peak in $[0; 0.1)$. Differently from the Catalyst there is no defined peak of AM4 around 1. This may be related to the relative difficulty of prediction on this dataset. AM4 still results in the most accurate models. For batch sizes 50 and 200, AM1 and AM2 also show comparable performance. AM3 has much higher peak on $[0; 0.1)$ interval and relatively lower values in other intervals. It can be seen that there are comparatively many instances with $RA < 0$, where the adaptation is conducted in the wrong direction. This may be related to the periodic nature of the dataset - as seen for example in the Figure 3.4. Additionally, batches with extreme values are not good representatives of the subsequent batches, which may make the adaptation based on them detrimental.

Figure 5.4 presents the relative adaptation histograms for the Drier dataset. Batch size of 50

[1] It should be noted that every AM in SABLE can actually result in no change of model's prediction, this is $f_0(\boldsymbol{x}) = f_i(\boldsymbol{x})$; this is related to the local weighting of the input data x - it is possible that an expert which has not been updated is given the weight of 1 for certain data instances. In addition, AM3, which recalculates the descriptors has no effect if the ensemble contains only a single expert.

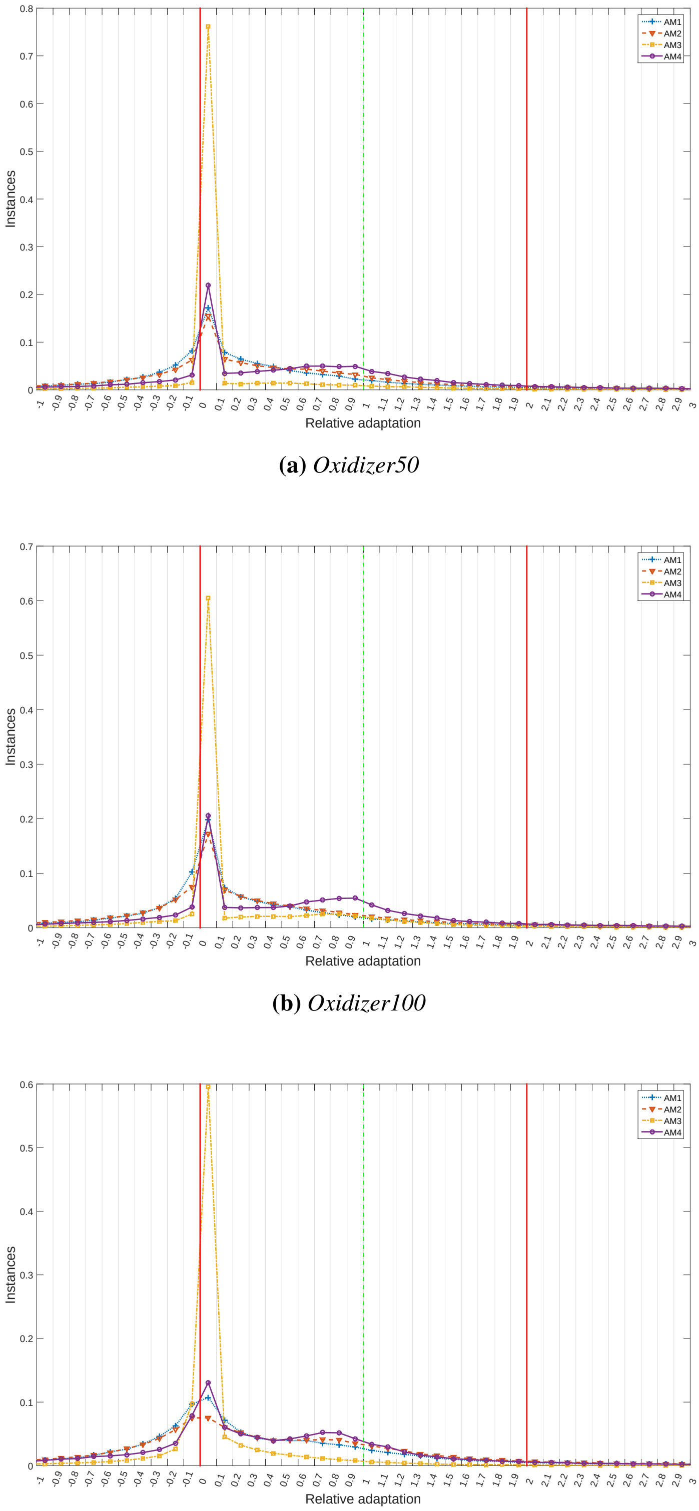


(a) *Catalyst50*

(b) *Catalyst100*

(c) *Catalyst200*

Figure 5.2: *Relative adaptation for Catalyst dataset. RA=0 and RA=2 thresholds are marked with solid red lines and RA=1 with dashed green one. Instances on y axis are normalized to scale [0,1].*



(a) *Oxidizer50*

(b) *Oxidizer100*

(c) *Oxidizer200*

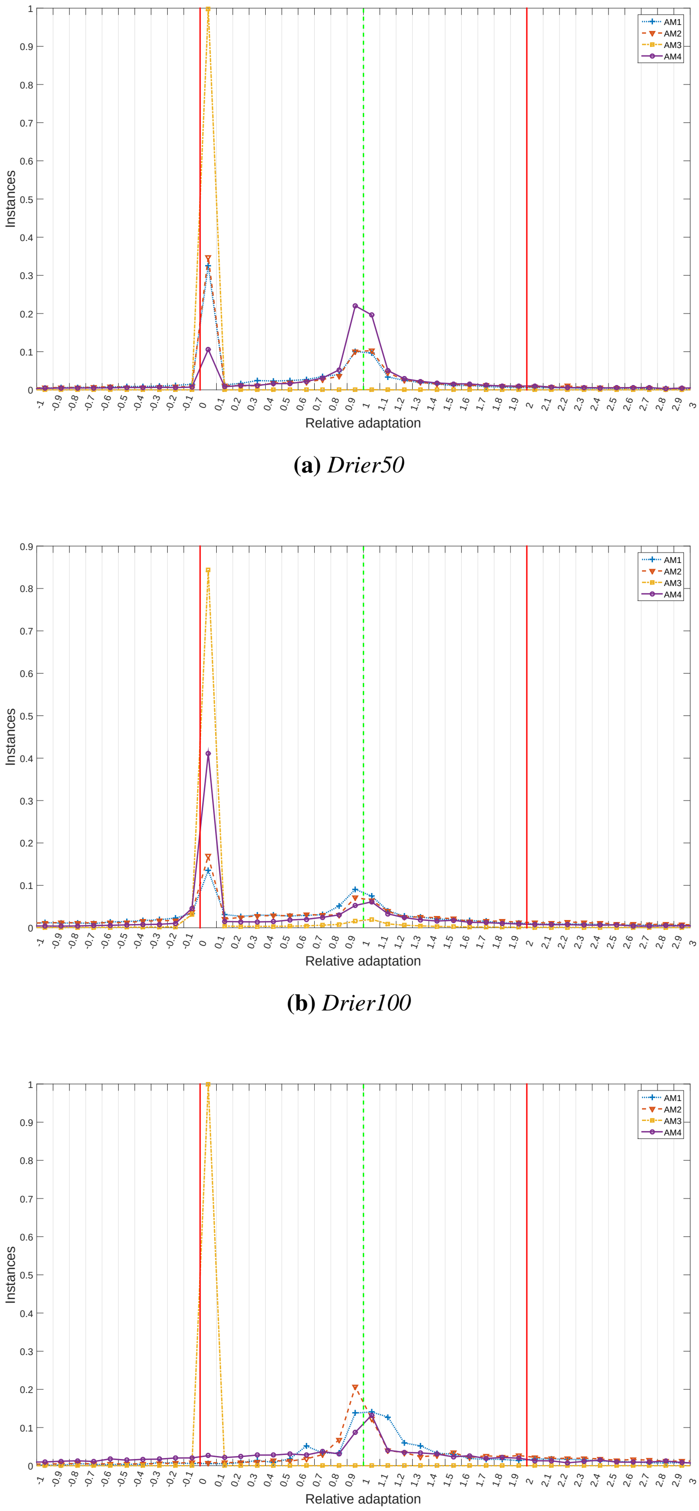
Figure 5.3: *Relative adaptation for Oxidizer dataset. RA=0 and RA=2 thresholds are marked with solid red lines and RA=1 with dashed green one. Instances on y axis are normalized to scale [0,1].*

shows peaks in $[0; 0.1)$, where AM3 has all of its RA values in this interval. AM1 and AM2 have more instances in this interval than AM4. In turn, AM4 has more instances around $RA = 1$ than other AMs. This behaviour is reversed for batch sizes 100 and 200, where AM1 and AM2 have more instances around $RA = 1$ than AM4. Batch size 200 doesn't have any peaks in the $[0; 0.1)$ interval except for AM3. Another interesting observation here is that there are relatively few instances with $0 < RA < 2$. This is a sign that adaptation may be detrimental to the many batches in the Drier dataset. It must be noted that this dataset, especially with the batch size 200 has much less data for building the histogram compared to other cases.

Another aspect of AM performance which was analysed was the relation of the error of adapted model to the error of un-adapted model (i.e. $f_i(\boldsymbol{x}) - y$ versus $f_0(\boldsymbol{x}) - y$). It is investigated whether the error of un-adapted model can be an indicator of an AMs' performance. The analysis is conducted using results from Section 5.2. Figure 5.5 depicts the scatter plots of the adaptive and non-adaptive error for the same data instance for the Catalyst100 dataset. It is seen that for this dataset, AM1 (retraining without forgetting), AM2 (retraining with forgetting) and AM3 (weights update) results are visibly correlated with the non-adaptive error, while AM4 (addition of new experts and weights update) is less so. In other words, the higher is the original error, the higher will be the error after deployment of AM1, AM2 and AM3. This effect is not visible for AM4. For the Oxidizer100 dataset (Figure 5.6), error after AM4 is again the least correlated with the non-adaptive error, however the differences between the AMs are less distinct. For the Drier100 dataset (Figure 5.7), error after AM4 is the most correlated with the non-adaptive error, and AM1 and AM2 are less so. For other batch sizes the conclusions are similar with few exceptions.

As a general conclusion, it appears that all of the AMs have mostly positive impact on the model for Catalyst and Oxidizer datasets, as there are comparatively few instances with $RA < 0$ or $RA > 2$. Drier dataset may not require adaptation as much, because in many cases, adaptation increases the prediction error. AM4, which is creating experts and subsequently changing the weights, is often the best performing AM. This is especially noticeable with Catalyst dataset. AM1 and AM2 often show similar behaviour, which can be expected, since both of them use the same method to retrain the experts, AM1 without and AM2 with forgetting. The forgetting seem to make AM2 adapt models slightly more accurately. In many cases, the adaptation effect is relatively small, as seen by peaks at [0;0.1). The reason may be the fact that the batches on which the errors are calculated can be quite different from the batches used for adaptation.

With some exceptions it was observed that the error values after AM4 are the least correlated with the non-adaptive error among all AMs. This means that most of the times AM4 is a very powerful adaptive mechanism, which can essentially adapt model without being limited by its previous state. This is also intuitively conceivable, as this AM creates a new expert with high weight from scratch and in many cases strongly decreases in the weights of older experts. AM1 and AM2 are limited in this sense, as in their case, even with forgetting, the adapted model will be more influenced by the original model. AM3 can exhibit various behaviours, evidently depending on the existing mix of experts in the ensemble.



(a) *Drier50*

(b) *Drier100*

(c) *Drier200*

Figure 5.4: *Relative adaptation for Drier dataset. RA=0 and RA=2 thresholds are marked with solid red lines and RA=1 with dashed green one. Instances on y axis are normalized to scale [0,1].*

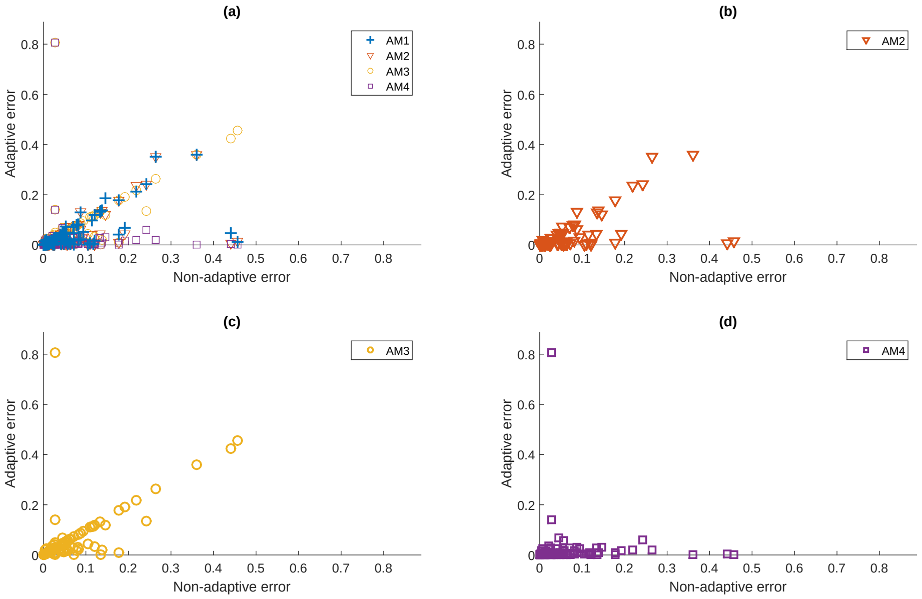


Figure 5.5: *Scatter plot of a random sample of adaptive vs non-adaptive errors for Catalyst100 dataset. (a) Comparison of all AMs, (b) AM2 separately, (c) AM3 separately, (d) AM4 separately.*

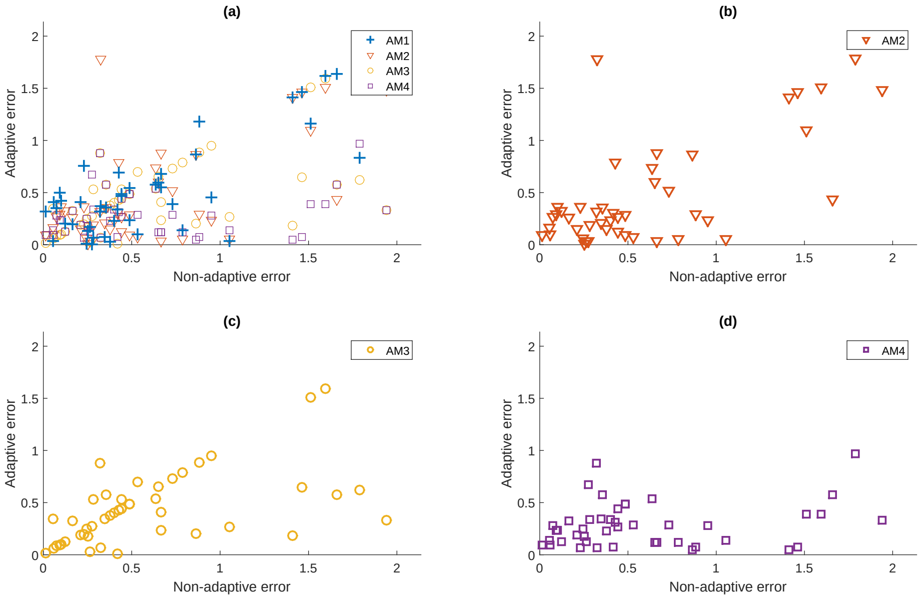


Figure 5.6: *Scatter plot of a random sample of adaptive vs non-adaptive errors for Oxidizer100 dataset. (a) Comparison of all AMs, (b) AM2 separately, (c) AM3 separately, (d) AM4 separately.*

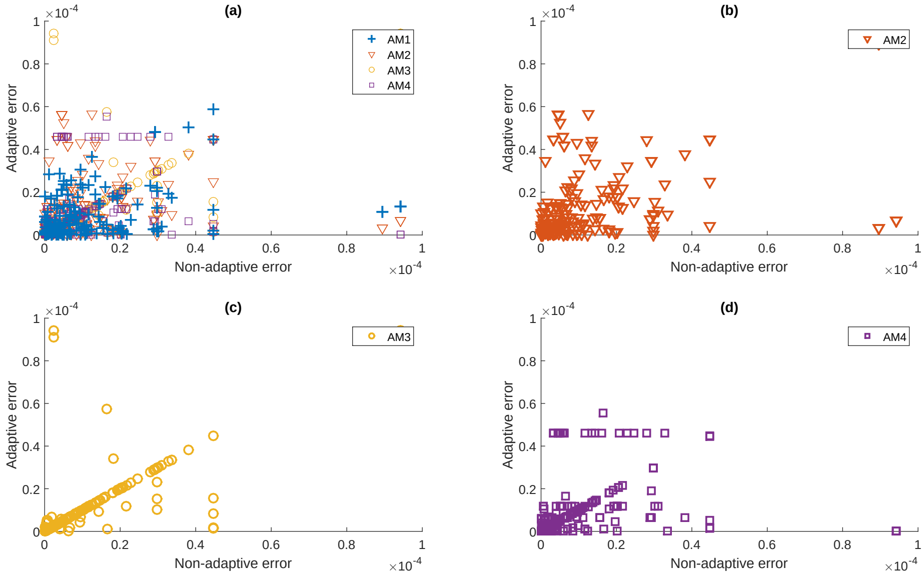


Figure 5.7: *Scatter plot of a random sample of adaptive vs non-adaptive errors for Drier100 dataset. (a) Comparison of all AMs, (b) AM2 separately, (c) AM3 separately, (d) AM4 separately.*

5.4 Adaptive mechanism selection

To run the predictive method on a batch of streaming data, an adaptive strategy for AM selection must be specified. After the analysis of AMs in the previous section, this section proceeds to introduce certain adaptive strategies. The aim of these adaptive strategies is the generation of an AM sequence which would try to minimize the error during the operation of the predictive method.

The simplest adaptive strategies are constant deployment of a certain AM set. These will be also called fixed adaptive strategies. This includes deployment of single AMs, effectively removing the multiple adaptive mechanism component from the system. A popular method which falls in this category is deployment of all the available AMs at every time step. As discussed earlier in Section 2.6, this is employed for many adaptive prediction methods.

5.4.1 Using cross-validation for adaptive mechanism selection

As opposed to fixed AM strategies, flexible adaptive strategies may deploy different AM combinations on different batches of data. With some assumptions about the data and how it changes, it may be possible to theoretically find the optimal AM to predict on the next batch of the data and even an optimal AM sequence of the whole dataset or its subset. In absence of such assumptions, the AMs can be selected based on previously seen data. One way of doing this is using the data from the last seen batch.

More formally, as introduced in (Bakirov et al., 2015), it is also possible to use batch $\boldsymbol{\mathcal{V}}_k$

for the choice of AM g_{h_k}. Given the observation, the *a posteriori* prediction error on $\boldsymbol{\mathcal{V}}_k$ is $\langle (f_k^- \circ g_{h_k})(\boldsymbol{X}_k), \boldsymbol{y}_k\rangle$, where f_k^- is the *a priori* predictive function at k-th batch. However, this is effectively an in-sample error as g_{h_k} is a function of $\{\boldsymbol{X}_k, \boldsymbol{y}_k\}$.[2] To obtain a generalised estimate of the prediction error 10-fold cross validation is applied. The cross-validatory adaptation strategy (denoted as *XVSelect*) uses a subset (fold), $\mathcal{S}$, of $\{\boldsymbol{X}_k, \boldsymbol{y}_k\}$ to adapt; i.e. $f_k^+ = f_k^- \circ g_{h_k}(\{\boldsymbol{X}_k, \boldsymbol{y}_k\}_{\in\mathcal{S}})$ (f_k^+ is the *a posteriori* predictive function at k-th batch) and the remainder, $\not{\mathcal{S}}$, is used to evaluate, i.e. find $\langle f_k^+(\boldsymbol{X}_k)_{\in\not{\mathcal{S}}}, \boldsymbol{y}_{k\in\not{\mathcal{S}}}\rangle$. This is repeated 10 times resulting in 10 different error values and the AM, $g_{h_k} \in G$, with the lowest average error measure is chosen. In summary:

$$f_{k+1}^- = f_k^- \circ g_{h_k}, \quad h_k = \underset{h_k\in 1\cdots H}{\operatorname{argmin}}\langle (f_k^- \circ g_{h_k})(\boldsymbol{X}_k), \boldsymbol{y}_k\rangle^{\times} \tag{5.2}$$

where $\langle\ \rangle^{\times}$ denotes the cross validated error.

5.4.2 Retrospective model correction

Recall that after batch $\boldsymbol{\mathcal{V}}_{k-1}$ has been obtained, any AM can be chosen to be deployed on $\boldsymbol{\mathcal{V}}_{k-1}$ to adapt f_{k-1}^- and obtain f_k^-. After $\boldsymbol{y}_k$ has been obtained, it is possible to identify the optimal AM which *could have* been deployed on $\boldsymbol{\mathcal{V}}_{k-1}$ as $g_{h_{k-1}^{opt}}$ where

$$h_{k-1}^{opt} = \underset{h_{k-1}\in 1\cdots H}{\operatorname{argmin}}\langle (f_{k-1}^- \circ g_{h_{k-1}})(\boldsymbol{X}_k), \boldsymbol{y}_k\rangle. \tag{5.3}$$

By definition, deploying $g_{h_{k-1}^{opt}}$ results in the most accurate predictions on batch $\boldsymbol{\mathcal{V}}_k$, $\hat{\boldsymbol{y}}_k$ among all AMs $g_{h_{k-1}} \in G$. Additionally, it has been observed that this often has a positive effect on the accuracy of model on the next batch, irrespective of subsequently deployed AM, g_{h_k}. This observation is in line with the conclusion Section 5.3, that AMs are influenced by the previous state of the model and deploying $g_{h_{k-1}^{opt}}$ results in the optimal model f_k^{-opt}.

Inspired by this observation, the strategy described in this section reverts the model to the optimal state f_k^{-opt} once $\boldsymbol{y}_k$ has been obtained. This is called the *retrospective model correction* (Bakirov et al., 2016). Specifically, it is identified which adaptation at batch $\boldsymbol{\mathcal{V}}_{k-1}$ would have produced an optimal estimate for batch $\boldsymbol{\mathcal{V}}_k$ and then it is deployed on f_{k-1}^-, before deploying the next AM:

$$f_{k+1}^- = f_{k-1}^- \circ g_{h_{k-1}} \circ g_{h_k}, \quad h_{k-1} = \underset{h_{k-1}\in 1\cdots H}{\operatorname{argmin}}\langle (f_{k-1}^- \circ g_{h_{k-1}})(\boldsymbol{X}_k), \boldsymbol{y}_k\rangle \tag{5.4}$$

Here, f_{k-1}^- is the *a priori* predictive function and $g_{h_{k-1}}$ is the AM for $k-1$-th batch, g_{h_k} is the AM for k-th batch and f_{k+1}^- is the *a priori* predictive function at $k+1$-th batch. Using cross-validated

[2] As a solid example consider the case where f_k^+ is f_k^- retrained using $\{\boldsymbol{X}_k, \boldsymbol{y}_k\}$. In this case $\boldsymbol{y}_k$ are part of the training set and so evaluating the goodness of fit on $\boldsymbol{y}_k$ is prone to overfitting.

Strategy	Description
Sequence0	Deploy AM0 on every batch. This means that only the first batch of data is used to create an expert.
Sequence1	Deploy AM1 on every batch.
Sequence2	Deploy AM2 on every batch.
Sequence3	Deploy AM3 on every batch.
Sequence4	Deploy AM4 on every batch.
Retrain	A new model is trained from the current batch and the old one is discarded.
Joint	Deploy AM2 and AM4 (in this order) on every batch. This strategy includes all of the available adaptation methods (batch learning, addition of new experts and change of weights)
XVSelect	Select AM based on the current data batch using the cross-validatory approach described in the Section 5.4.1.
Optimal	Select AM based on the next data batch as described in the Section 4.5.1. Used for benchmarking.

Table 5.2: *Adaptive strategies.*

error measure in Equation 5.4 is not necessary, because $g_{h_{k-1}}$ is independent of $\boldsymbol{y}_k$. Also note the presence of g_{h_k}; retrospective correction does not in itself produce a f_{k+1} and so cannot be used for prediction unless it is combined with another strategy (g_{h_k}).

Similarities may be found between retrospective correction and weight update backtrack mechanism, a key feature of Resilient Propagation neural network learning algorithm (Riedmiller & Braun, 1993). In fact, this mechanism can be considered as a special case of retrospective correction. However, to the best of author's knowledge, the general notion and analysis of retrospective model correction strategy have not been given in the adaptation context before.

Retrospective model correction strategy can be extended to consider the sequence of r AMs while choosing the optimal state for the current batch, which will be called r-step retrospective correction:

$$\begin{aligned} f^{-}_{k+1} &= f^{-}_{k-r} \circ g_{h_{k-r}} \circ \cdots \circ g_{h_{k-1}} \circ g_{h_k}, \\ \{h_{k-r} \cdots h_{k-1}\} &= \underset{h_{k-r} \cdots h_{k-1} \in 1 \cdots H}{\operatorname{argmin}} \langle (f^{-}_{k-r} \circ g_{h_{k-r}} \circ \cdots \circ g_{h_{k-1}})(\boldsymbol{X}_k), \boldsymbol{y}_k \rangle \end{aligned} \tag{5.5}$$

5.4.3 Results

To assess the usefulness of different adaptive strategies, the strategies shown in Table 5.2, with and without retrospective correction, were evaluated on the Catalyst, Oxidizer and Drier datasets.

In Table 5.3 the results on the Catalyst dataset with batch sizes of 50, 100 and 200, (Catalyst50, Catalyst100 and Catalyst200) are shown. The best result in terms of MAE among the methods with flexible AM deployment order are denoted with a †, and among the methods with fixed order with ‡. It is checked whether the error values of these two strategies are significantly different with a significance level $\alpha \leq 0.05$, and if so the most accurate strategy is marked with a bold font. *RC* at the end of the strategy name denotes that the retrospective model correction was used. Using the benchmark (*Optimal*) strategy AM, which minimizes MAE for the incoming batch of data, always led to the lowest MAE for the whole dataset.

		Catalyst50		**Catalyst100**		**Catalyst200**	
	Strategy	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Fixed Order	**Sequence0**	0.3098	0.3422	0.2787	0.3130	0.3612	0.3972
	Sequence1	0.1376	0.2481	0.1466	0.2596	0.1606	0.2879
	Sequence2	0.0227‡	0.0537	0.0309	0.0674	0.0585	0.1394
	Sequence4	0.0368	0.1017	0.0310	0.0625	0.0521	0.0952
	Joint	0.0301	0.0680	0.0349	0.0775	0.0495‡	0.0847
	Retrain	0.0241	0.0814	0.0278‡	0.0583	0.0516	0.1076
Flexible Order	**Sequence0 RC**	0.0306	0.0609	0.0449	0.0793	0.0725	0.1090
	Sequence1 RC	0.0256	0.0521	0.0385	0.0698	0.0730	0.1246
	Sequence2 RC	0.0263	0.0530	0.0354	0.0668	0.0727	0.1226
	Sequence3 RC	0.0261	0.0467	0.0452	0.0804	0.0673	0.0938
	Sequence4 RC	0.0211	0.0430	0.0307	0.0615	0.0534	0.0966
	Joint RC	0.0196	0.0409	0.0325	0.0708	0.0519	0.0952
	XVSelect	0.0205	0.0481	0.0301	0.0598	**0.0492**†	0.0963
	XVSelect RC	**0.0171**†	0.0347	0.0295†	0.0613	0.0495	0.0959
	Optimal	*0.0149*	*0.0306*	*0.0233*	*0.0467*	*0.0403*	*0.0691*

Table 5.3: *Catalyst dataset results averaged over all batches. The least MAE per batch size among the methods with flexible AM deployment order are denoted with a †, and among the methods with fixed order with ‡. If the error values of these two strategies are significantly different (according to (Mizrach, 1996), Section 3.2 with $\alpha \leq 0.05$) the most accurate strategy is marked with a bold font.*

For Catalyst50 and Catalyst100, the best flexible strategy is *XVSelect RC* and the best fixed strategies are respectively *Sequence2* and *Retrain*. For Catalyst200 *XVSelect* and *Joint* perform better in the respective groups. For Catalyst50 and Catalyst200, the errors of the most accurate flexible AM order are significantly different from the errors of the most accurate fixed AM order generated by investigated adaptive strategies. The distribution of the AMs in the *Optimal* strategy and to what extent *XVSelect RC* AMs match with them is shown in the Figure 5.8. It is noticeable that AM4 is the most common AM in the *Optimal* strategy, meaning that it often delivers the most accurate results. This is also the reason why it is often selected by *XVSelect RC*. On Catalyst100 dataset, *XVSelect RC* and *Optimal* have the least common AMs on average (Figure 5.8). This can be the reason why *Retrain* is more accurate in this case.

The importance of the proper selection of AMs is illustrated in Figure 5.9 which presents MAE values after exhaustively deploying all possible combinations of AMs for four steps ahead at every batch on the Catalyst50 dataset as described in Section 5.2. From this figure it is possible to see that the choice of the wrong AM can result in a drastic increase in the prediction error.

Additional experiments were performed with 0 to 3 steps retrospective correction as described in Equation 5.5 for *XVSelect* for batch sizes 50 to 200. Figure 5.10(a) shows the results of these

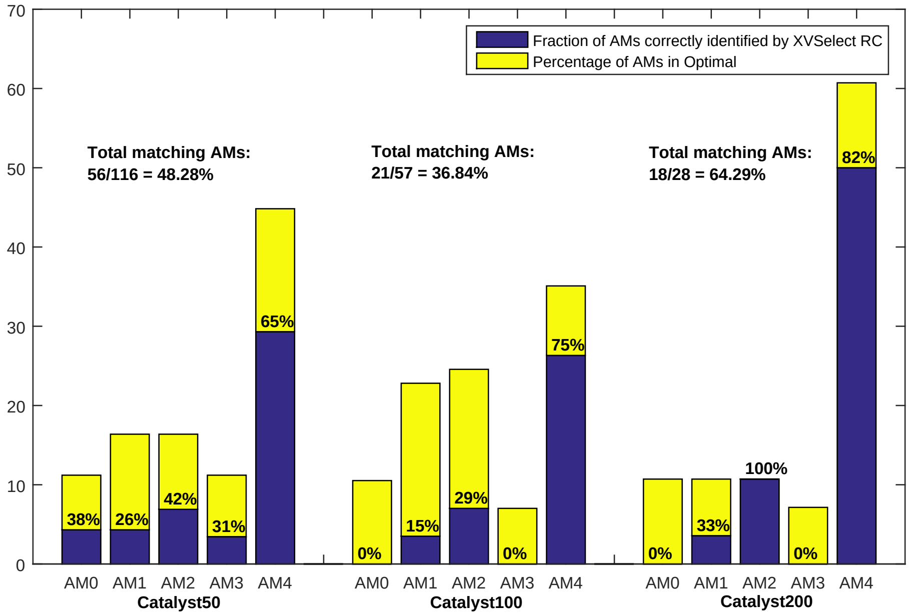


Figure 5.8: *Adaptive mechanisms generated by Optimal strategy for the Catalyst dataset.*

experiments. It can be seen that while one step correction generally improves the performance, using more than one step retrospective correction usually does not bring improvement to the

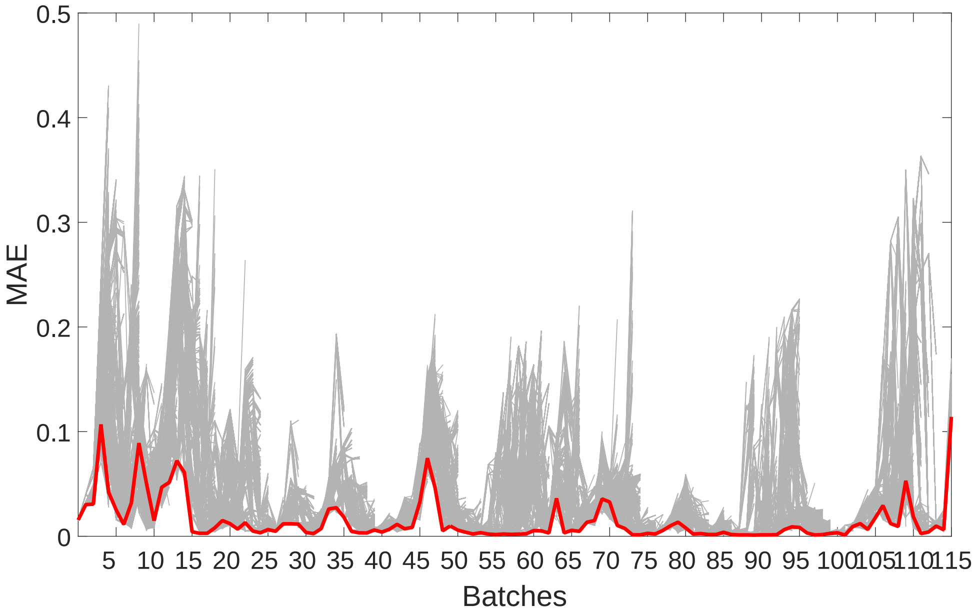


Figure 5.9: *4 step ahead exhaustive AM deployment on all batches of Catalyst50 dataset as described in Section 5.2. Red line shows MAE values of Optimal strategy.*

predictive accuracy, and in fact often decreases it. This could be related to the fact that using more retrospective correction steps may cause the overfitting of the model to the current batch. For higher batch sizes even one step retrospective correction is detrimental. This also can be related to the overfitting issues, which are exacerbated with the batch size increase, as the batches become less similar.

Figure 5.11[3] compares the true target values and predicted values of *Optimal*, *XVSelect RC* and *Sequence2* on *Catalyst50* dataset. The AMs deployed by *Optimal* and *XVSelect* are also shown, marked red when they differ. It can be seen that in the beginning of the dataset, where the target value changes quite fast, flexible strategies perform noticeably better. In the more stable parts of the data, such as batches #50-#65 or #85-#94, the differences are much less drastic. *Optimal* and *XVSelect RC* can suffer from fluctuations (e.g. batches #107-#108) which may be an artefact of the SABLE algorithm, depending on its settings.

[3] Figures 5.11, 5.14 and 5.17 show plots of the results starting from the second batch, as the first batch is used for training of the initial expert and no predictions on it are made.

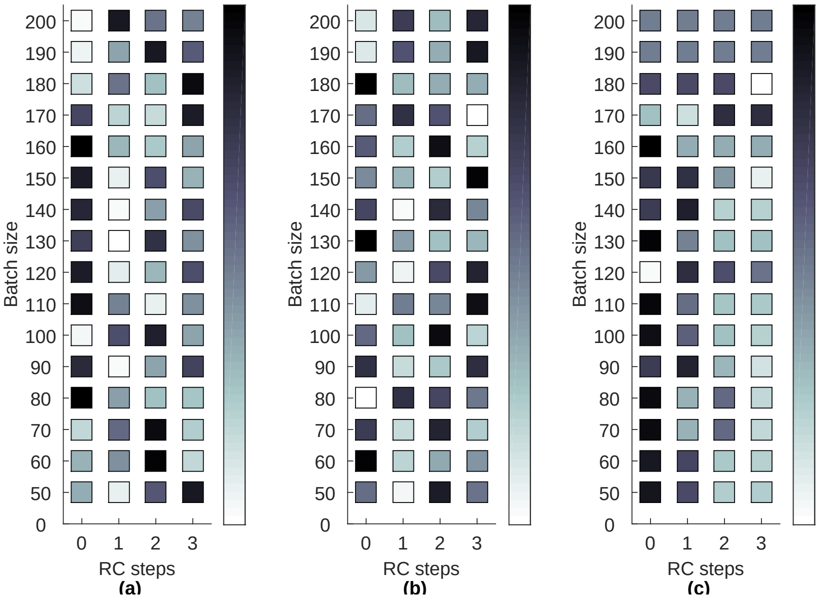


Figure 5.10: *Normalized XVSelect results' comparison with different batch sizes. White is the minimal and black is the maximal error. (a) Catalyst dataset (b) Oxidizer dataset (c) Drier dataset.*

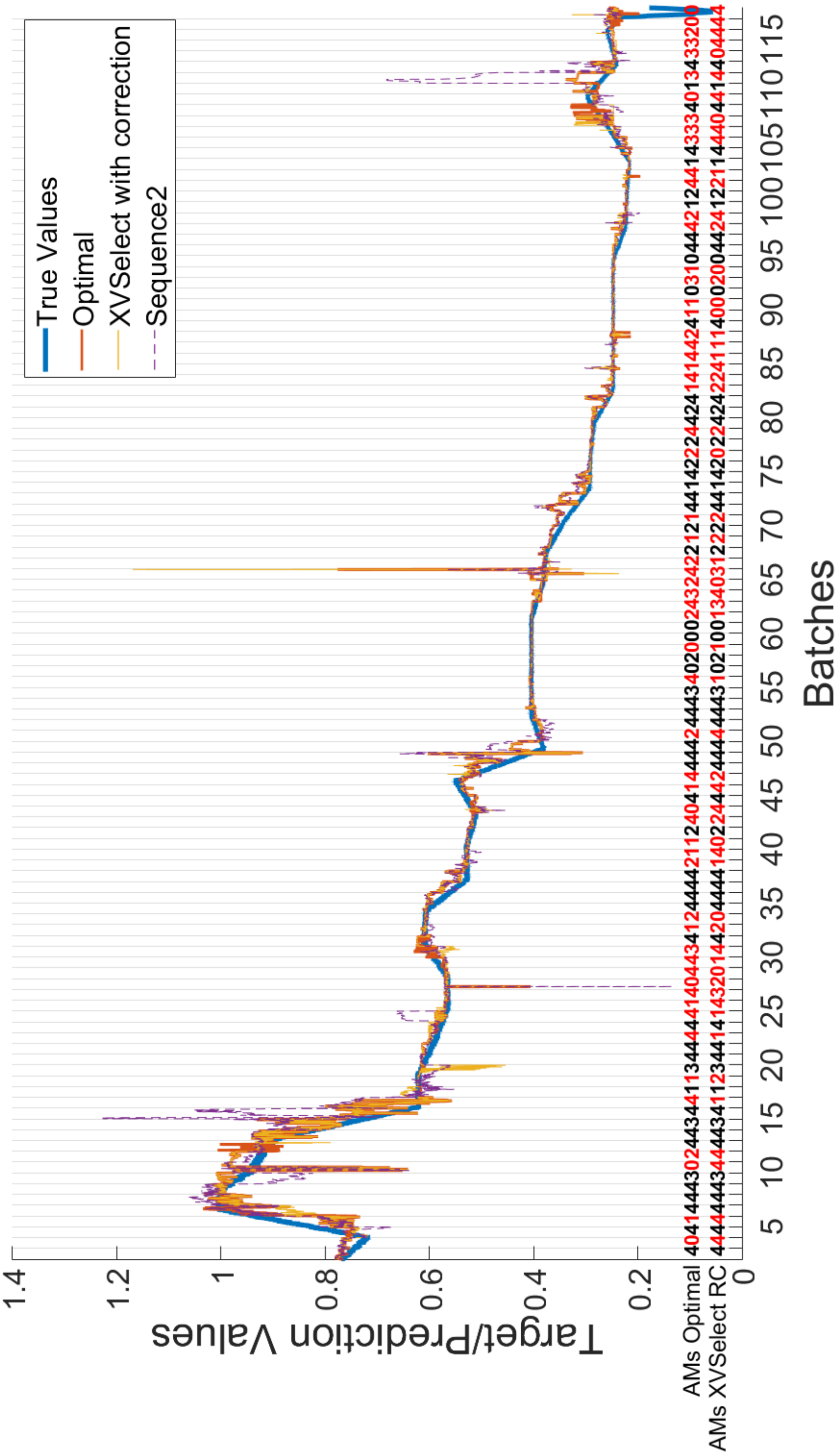


Figure 5.11: *Predictions on Catalyst50 dataset.*

		Oxidizer50		**Oxidizer100**		**Oxidizer200**	
	Strategy	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Fixed Order	**Sequence0**	0.760	1.181	0.779	1.218	0.783	1.130
	Sequence1	0.640	0.998	0.662	1.043	0.613	0.891
	Sequence2	0.490‡	0.838	0.564	0.966	0.685	1.009
	Sequence4	0.504	0.851	0.543‡	0.929	0.620	0.961
	Joint	0.494	0.852	0.590	0.975	0.612‡	0.988
	Retrain	0.499	0.854	0.565	0.944	0.678	1.063
Flexible Order	**Sequence0 RC**	0.538	0.981	0.636	0.984	0.687	0.993
	Sequence1 RC	0.522	0.962	0.650	0.997	0.679	0.972
	Sequence2 RC	0.515	0.925	0.648	1.016	0.677	0.981
	Sequence3 RC	0.551	0.993	0.589	0.927	0.674	0.918
	Sequence4 RC	0.487	0.842	0.533	0.906	0.613	0.924
	Joint RC	0.478	0.831	0.570	0.974	**0.611**†	0.969
	XVSelect	0.499	0.868	0.555	0.908	0.631	0.995
	XVSelect RC	**0.468**†	0.823	**0.529**†	0.871	0.676	0.988
	Optimal	*0.416*	*0.766*	*0.481*	*0.822*	*0.530*	*0.804*

Table 5.4: *Oxidizer dataset results averaged over all batches. The least MAE per batch size among the methods with flexible AM deployment order are denoted with a †, and among the methods with fixed order with ‡. If the error values of these two strategies are significantly different (according to (Mizrach, 1996), Section 3.2 with* $\alpha \leq 0.05$*) the most accurate strategy is marked with a bold font.*

In Table 5.4 the results on Oxidizer dataset with batch sizes of 50, 100 and 200 (Oxidizer50, Oxidizer100, Oxidizer200) are presented. The best results in this dataset are achieved with flexible adaptive strategies – *XVSelect RC* (for Oxidizer50 and Oxidize100) and *Joint RC* (for Oxidize200). They all result in significantly different error values from the ones obtained after deploying the most accurate fixed strategies; *Sequence2*, *Sequence4* and *Joint* for Oxidizer50, Oxidizer100, Oxidize200 respectively. Except for Oxidizer200, as shown in the Figure 5.12, AM4 is most often the best AM. It is however less dominant than in the Catalyst dataset. For this dataset, the distribution of AMs in *Optimal* sequence is more uniform than for the Catalyst data. Oxidizer200 has the lowest number of matching AMs in *XVSelect RC* and *Optimal* sequences. This is reflected in comparatively low predictive accuracy of the *XVSelect RC* for that case.

As seen in Figure 5.13 the order of AMs makes a large difference for predicting on the Oxidizer dataset as well. From the Figure 5.10(b) similar observations as for previous dataset could be made; one step restrospective correction generally improves the accuracy, but using more than one step retrospective correction in most cases does not improve it further, and often causes its deterioration. Figure 5.14 compares the true target values and predicted values of *Optimal*, *XVSelect RC* and *Joint* on Oxidizer50 dataset. The AMs deployed by *Optimal* and *XVSelect* are also shown and marked red when they differ. It can be observed that the Oxidizer dataset has a cyclic characteristic, with extreme values roughly every 5 batches. Often after these extreme values, *Joint* prediction values show higher errors (e.g. batches #4, #13, #30, #44 etc.). This can be related to the strong adaptation of this strategy, which likely overfits the model to the extreme values. *XVSelect* in contrast behaves more stable with less obvious jumps to extreme values.

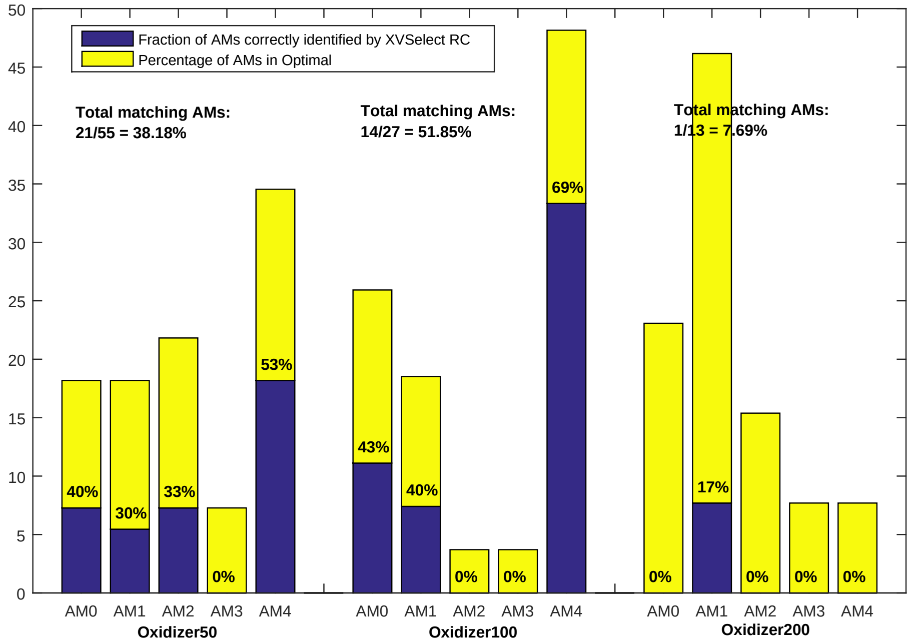


Figure 5.12: *Adaptive mechanisms generated by Optimal strategy for the Oxidizer dataset.*

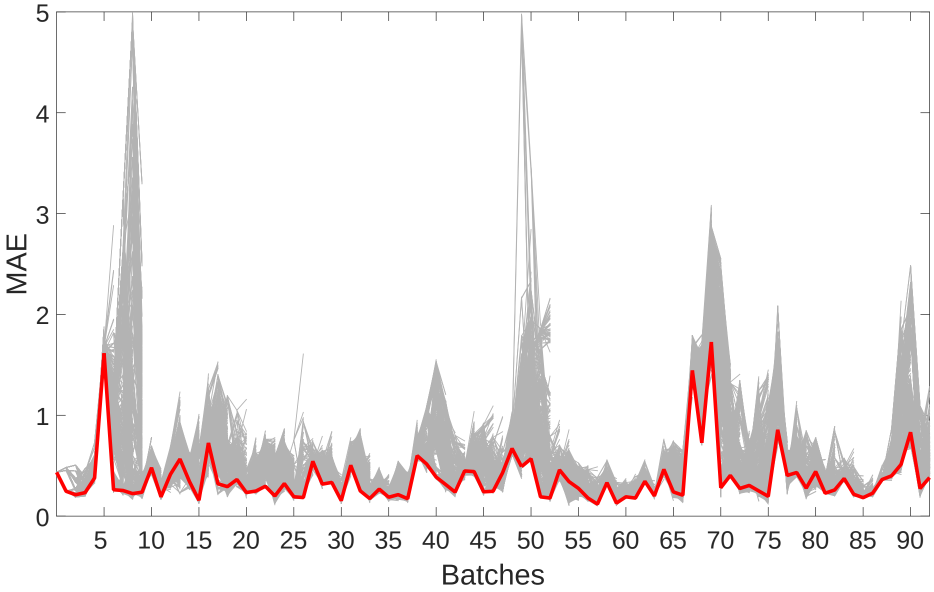


Figure 5.13: *4 step ahead exhaustive AM deployment on all batches of Oxidizer50 dataset as described in Section 5.2. Red line shows MAE values of Optimal strategy.*

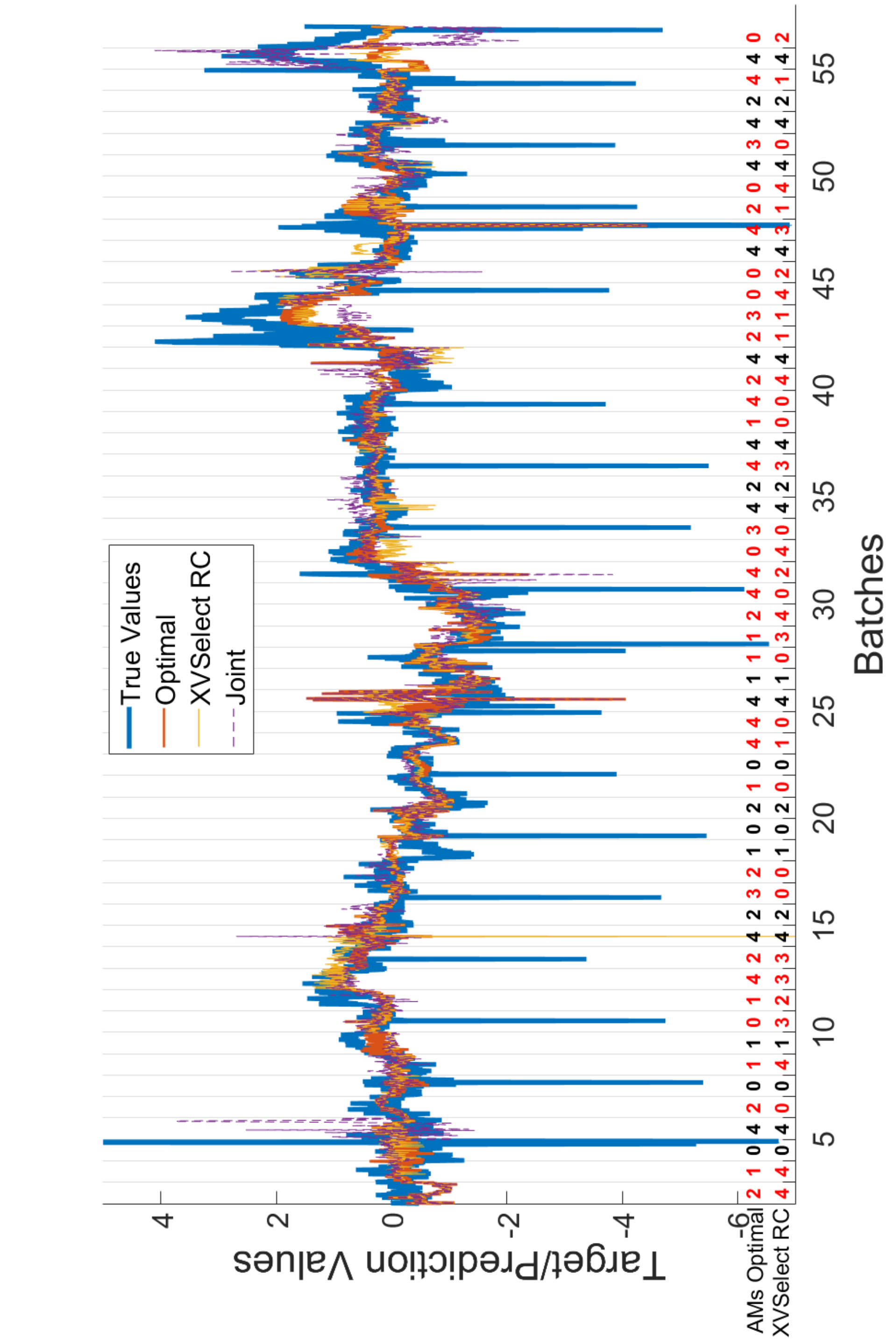


Figure 5.14: *Predictions on Oxidizer50 dataset.*

		Drier50		**Drier100**		**Drier200**	
	Strategy	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Fixed Order	**Sequence0**	7.68E-04	8.67E-04	5.41E-04	5.97E-04	3.87E-04	4.07E-04
	Sequence1	8.98E-06‡	2.57E-05	**8.09E-06‡**	2.04E-05	5.06E-05	1.43E-04
	Sequence2	3.04E-05	2.04E-04	1.75E-05	1.15E-04	5.28E-05	1.44E-04
	Sequence4	9.86E-05	3.41E-04	1.43E-05	8.30E-05	8.77E-05	1.92E-04
	Joint	4.06E-05	2.40E-04	1.34E-05	8.28E-05	5.01E-05‡	1.43E-04
	Retrain	5.86E-05	3.14E-04	2.59E-05	1.54E-04	5.38E-05	1.44E-04
Flexible Order	**Sequence0 RC**	9.78E-06	7.02E-05	1.43E-05	5.39E-05	1.34E-04	2.42E-04
	Sequence1 RC	1.02E-05	4.56E-05	8.96E-06†	2.09E-05	5.41E-05	1.43E-04
	Sequence2 RC	3.02E-05	1.41E-04	1.79E-05	1.15E-04	5.09E-05	1.43E-04
	Sequence3 RC	9.78E-06	7.02E-05	1.44E-05	5.39E-05	1.34E-04	2.42E-04
	Sequence4 RC	4.06E-05	2.40E-04	1.34E-05	8.28E-05	6.41E-05	1.64E-04
	Joint RC	4.16E-05	2.40E-04	1.37E-05	8.30E-05	5.01E-05	1.43E-04
	XVSelect	9.27E-06	4.02E-05	1.20E-05	3.08E-05	4.67E-05†	1.43E-04
	XVSelect RC	6.95E-06†	3.99E-05	1.12E-05	3.06E-05	4.67E-05†	1.43E-04
	Optimal	*3.40E-06*	*3.47E-05*	*3.15E-06*	*1.15E-05*	*4.67E-05*	*1.43E-04*

Table 5.5: *Drier dataset results averaged over all batches. The least MAE per batch size among the methods with flexible AM deployment order are denoted with a †, and among the methods with fixed order with ‡. If the error values of these two strategies are significantly different (according to (Mizrach, 1996), Section 3.2 with* $\alpha \leq 0.05$*) the most accurate strategy is marked with a bold font.*

In Table 5.5 the results of experiments on Drier dataset, with batch sizes of 50, 100 and 200 (Drier50, Drier100 and Drier200) are presented. From the results, it is clear that the Drier dataset is the least changing dataset. Simple RPLS online update *Sequence1* performs not significantly different for Drier50 and better for Drier100 than strategies with stronger adaptation or the best flexible AM orders (respectively *XVSelect RC* and *Sequence1 RC*). For the largest batch size of Drier200, the errors of the best performing flexible strategies *XVSelect* and *XVSelect RC* (which in this case deploy exactly the same AMs as *Optimal*) do not not significantly differ from the errors obtained after deploying *Joint*. It is worth noting that for this dataset there are only 5 batches of the test data for the batch size of 200 available. As seen from the Figure 5.15, AM0 is prevalent AM for this dataset. This can be related to the lack of changes in the data.

Figure 5.16 also confirms that the order of AMs makes less difference on the predictive accuracy for this dataset, except around the batches #6-#7. Figure 5.10(c) shows that as opposed to the previous datasets, predictive accuracy on Drier data usually improves when increasing retrospective correction steps. This can be also related to the stability of the dataset, where stronger optimization of the model towards the current batch does not cause overfitting issues.

This space has been intentionally left blank.

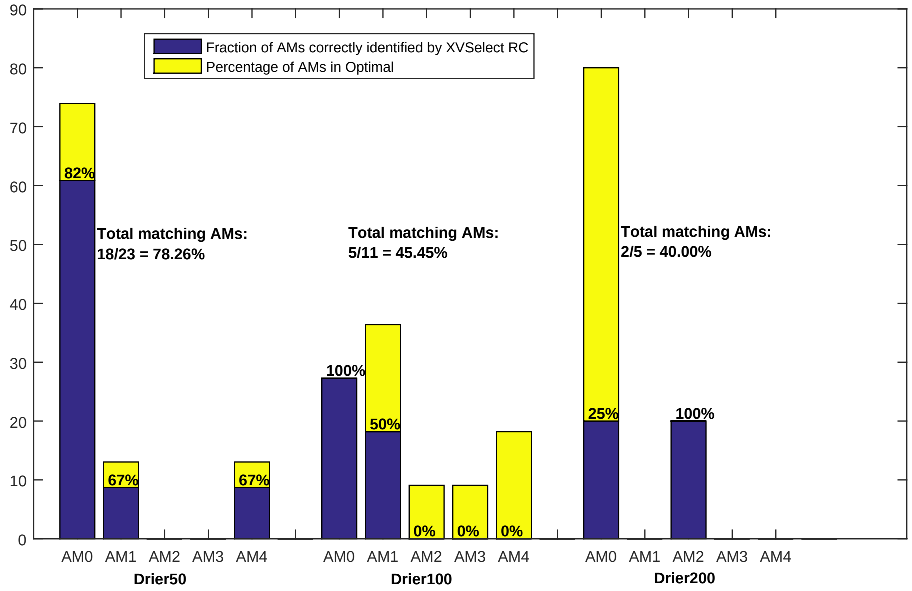


Figure 5.15: *Adaptive mechanisms generated by Optimal strategy for the Drier dataset.*

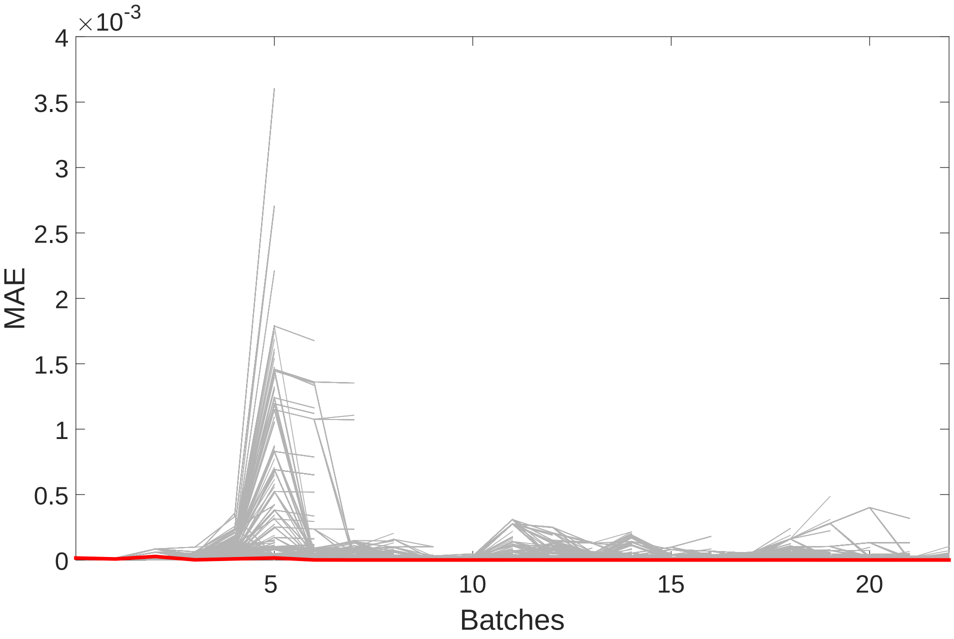


Figure 5.16: *4 step ahead exhaustive AM deployment on all batches of Drier50 dataset as described in Section 5.2. Red line shows MAE values of Optimal strategy.*

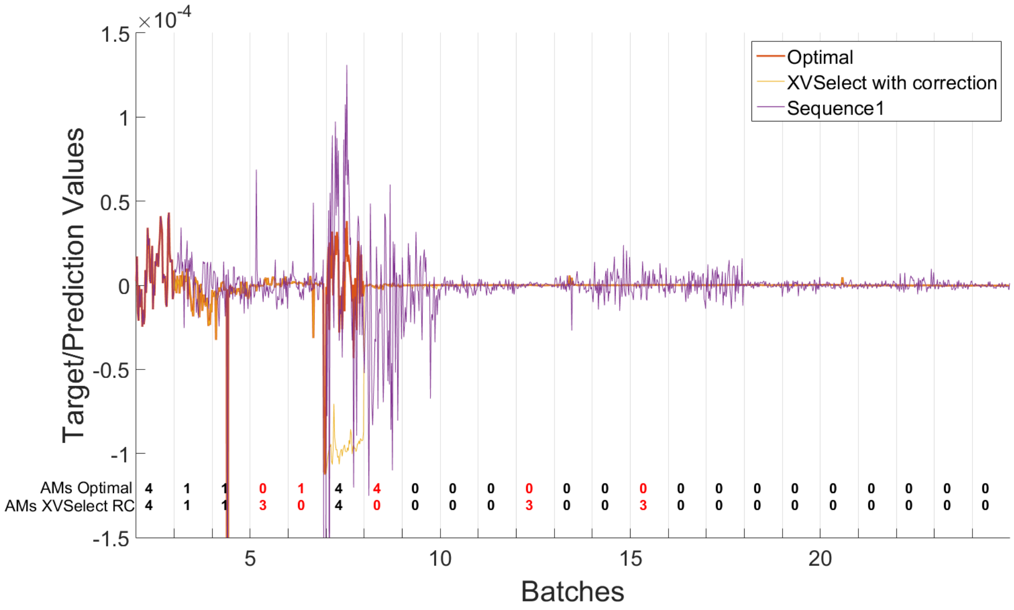


Figure 5.17: *Drier50 dataset error values (separately on every sample).*

Figure 5.17 compares error values of *Optimal*, *XVSelect RC* and *Sequence1* on Drier50 dataset[4]. The figure shows that the single wrong choice of AM at batch #5 causes a noticeable difference between predictions of *Optimal* and *XVSelect RC* predictions at batch #6. Otherwise the performances of these two strategies are similar. The advantages of using flexible multiple adaptive mechanisms are clear as well; flexible sequences visibly benefit from deployment of AM4 on batch #6 which reduces and stabilizes the error much more quickly than AM1 (simple learning of new data using RPLS online update) employed by *Sequence1*. In later batches, there is no apparent need for training, and flexible strategies deploy AM0 keeping the error stable, whereas using AM1 is causing noticeable fluctuations.

5.5 Prediction of optimal adaptive mechanism

In tables in section 5.4.3 it was shown that the one step ahead optimal AM sequence achieves better predictive performance than any of the other adaptive strategies. It was also shown that the *XVSelect*, in other words, cross-validation on the previous batch, is not very successful in predicting which AM will be optimal to deploy to get the least error on the next batch.

This section attempts to use a classification method to predict the next optimal AM, using certain meta-data about the prediction process. More formally, the classification can be expressed as:

$$\hat{h}_k = \upsilon(\boldsymbol{\chi}_k), \tag{5.6}$$

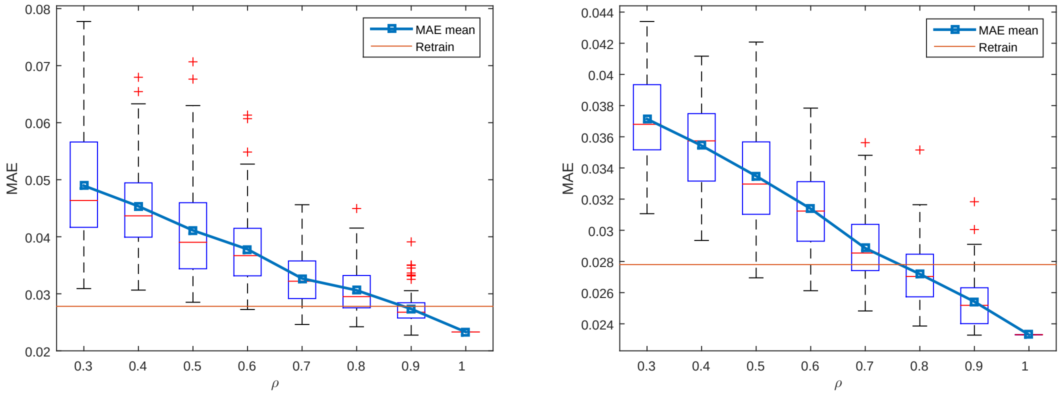
[4] The predictions for this dataset are not shown, as their errors are much smaller than the axis scale, making them impossible to distinguish from the target values.

where $\hat{h}_k$ is an estimate of the best AM to choose, and is dependent on the current and previous batches, $\boldsymbol{\mathcal{V}}_k, \boldsymbol{\mathcal{V}}_{k-1}, ...$, through a vector of meta-data features χ_k (discussed below) and υ is a classification function. Following the deployment of $g_{\hat{h}_k}$ on f_k, note that the performance is evaluated on batch $\boldsymbol{\mathcal{V}}_{k+1}$. After $\boldsymbol{\mathcal{V}}_{k+1}$ is available, the optimal AM,

$$g_{h_k^{opt}} = \underset{h_k \in 1 \cdots H}{\operatorname{argmin}} \langle (f_k^- \circ g_{h_k})(\boldsymbol{X}_{k+1}), \boldsymbol{y}_{k+1} \rangle \tag{5.7}$$

among all available AMs deployed on k-th batch, $h_k \in 1 \cdots H$, will be known. Then, if $h_k^{opt} = \hat{h}_k$ the prediction of meta-classifier was correct, and otherwise false.

In Section 5.4 promising results while using certain adaptive strategies were observed. Here, how accurate a classifier would need to be to improve upon those algorithms is investigated. A *pseudo-classifier*, which is a method that selects the correct class with a defined probability of ρ is used for this purpose. Note that obviously the correct classes must be known to construct such a classifier. Experiments using 100 random runs with pseudo-classifiers with accuracy levels varying from $\rho = 0.3$ to 0.9 on the Catalyst100, Oxidizer100, and Drier100 datasets were conducted.



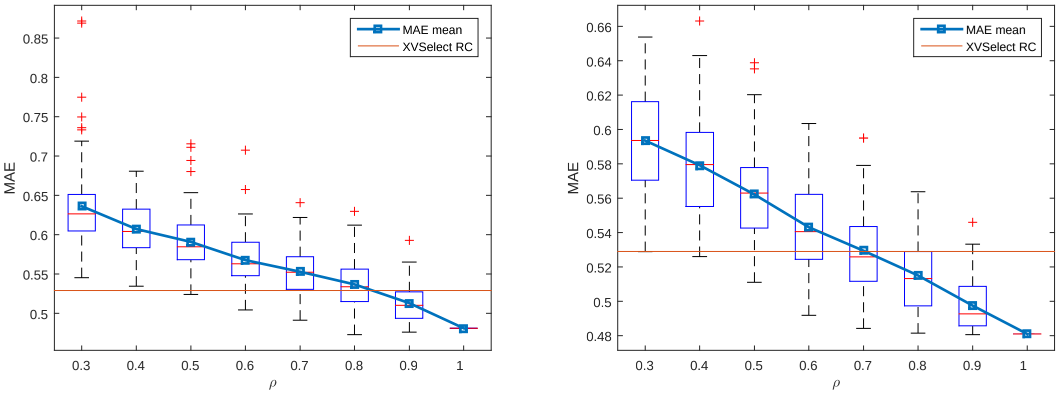
(a) *Catalyst100* **(b)** *Catalyst100 RC.*

Figure 5.18: *Pseudo-classifier MAE for Catalyst100 dataset for different values of ρ, with and without retrospective correction.*

Figure 5.18 shows box plots of MAE values on the whole dataset for the different values of ρ for the Catalyst100 dataset, with and without retrospective correction. It is seen that for this data set, without retrospective correction, one would require a classifier with $\rho > 0.9$ to achieve a MAE that is lower than the best adaptive strategy (see Section 5.4.3), here the *Retrain* strategy.[5] Using retrospective correction generally improves prediction models and so lowers the classification accuracy one would require to achieve better performance than for the *Retrain*

[5] This is an oversimplification, as the effects of misclassification of different AMs can be different. This is discussed in more detail in the Section 5.6.1

strategy to $\rho = 0.8$. An interesting observation is that in Figure 5.18(a), some MAE values for $\rho = 0.9$ are in fact lower than for $\rho = 1$, which is the *Optimal* adaptive strategy. This is an example of the fact stated above, that the *Optimal* sequence is only optimal for minimizing the error one step ahead, and may not be optimal for a multi-step ahead sequence or to minimize the error on the whole dataset.



(a) *Oxidizer100* **(b)** *Oxidizer100 RC.*

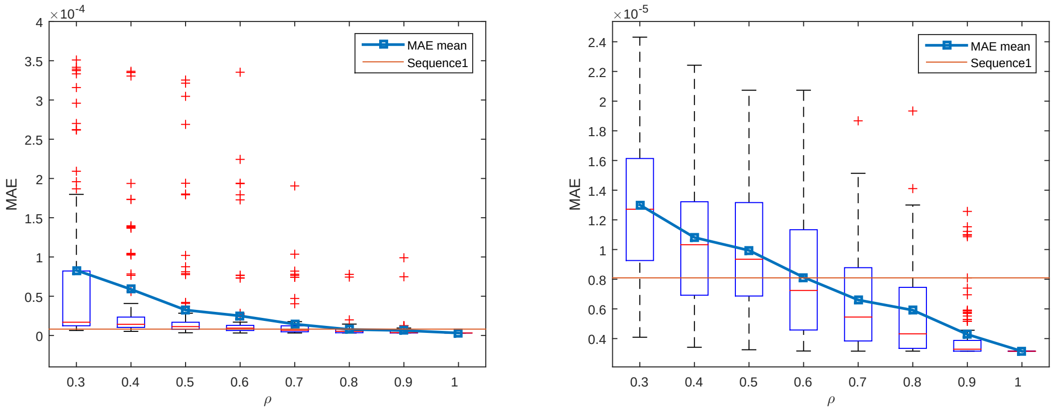
Figure 5.19: *Pseudo-classifier MAE for Oxidizer100 dataset for different values of ρ, with and without retrospective correction.*

Figure 5.19 shows box plots of MAE values on the whole dataset for the different values of ρ for Oxidizer100 dataset, with and without retrospective correction. It is seen that for this data set, without retrospective correction, only when $\rho > 0.9$, MAE values are mostly higher than the best result for this dataset from the Section 5.4.3, *Sequence4 RC*. Using retrospective correction lowers the accuracy needed to achieve similar on average, or better performance than *Sequence4 RC* to $\rho = 0.7$. As in the previous figure, an observation can be made that in Figure 5.19(a), some MAE values when using accuracy levels of $\rho = 0.8$ and $\rho = 0.9$ are lower than when using accuracy level $\rho = 1$, for the same reason as described above.

Figure 5.20 shows box plots of MAE values on the whole dataset for the different values of ρ for Drier100 dataset, with and without retrospective correction. For this dataset, without retrospective correction, only when $\rho > 0.8$, MAE values are mostly higher than the best result for this dataset from the Section 5.4.3, *Sequence1*. With retrospective correction the accuracy needed to achieve similar on average, or better performance than *Sequence1* is $\rho = 0.6$.

5.5.1 Meta-features for adaptive mechanisms' prediction

It has been shown, for example in Lemke & Gabrys (2010) or Lemke et al. (2015) that meta-learning is a powerful tool which can increase predictive accuracy of the algorithm using meta-knowledge about its behaviour. In this light it is attempted to construct a meta-classifier which can match the performance requirements identified in the previous Section. For this



(a) *Drier100 (some extreme values are not shown)*

(b) *Drier100 RC.*

Figure 5.20: *Pseudo-classifier MAE for Drier100 dataset for different values of ρ, with and without retrospective correction.*

purpose the set of meta-features shown in Table 5.6 is used. These features can be grouped into three categories:

A ***Features 1-6, optimal/deployed AMs for previous batches:*** These features provide information whether the optimal AMs were in fact deployed on the previous batches. This could influence the need for a strong adaptation on the subsequent batch, as seen previously for example in Figure 4.8.

B ***Features 7-17, statistics about the error values in previous batches:*** These can indicate whether there have been changes in the error and consequently changes in the data recently.

C ***Features 18-24, symmetric Kullback-Leibler divergences between input features and outputs:*** these values are also indicative of the changes in the process. *KLDATA0* should be especially noted, as it is the only feature with information about the next data batch, however only the input data $\boldsymbol{X}_{k+1}$.

To construct the meta-features table, the results from the exhaustive AM deployment as described in Section 5.2 were used.

5.6 Adaptive mechanism classification results

The accuracy of classification was tested using a *leave-batch-out* technique. That is, if the dataset consists of K batches, $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \mathcal{V}_K$, every batch in succession is used as the test data for the evaluation of the classification error, while the rest of the dataset is used as the training data for the classifier. This was done to avoid having the instances from the same batch both in training and test data; because they share certain features this could have led to the reporting of overly optimistic accuracy rates. The accuracy rates are then averaged to obtain the final classifier

Number	Abbreviation	Feature
1	OAM3	Optimal AM for $\mathcal{V}_{k-2}$
2	OAM2	Optimal AM for $\mathcal{V}_{k-1}$
3	OAM1	Optimal AM for $\mathcal{V}_{k}$
4	AM3	AM deployed on $\mathcal{V}_{k-2}$
5	AM2	AM deployed on $\mathcal{V}_{k-1}$
6	AM1	AM deployed on $\mathcal{V}_{k}$
7	MAE4	MAE on $\mathcal{V}_{k-3}$
8	MAE3	MAE on $\mathcal{V}_{k-2}$
9	MAE2	MAE on $\mathcal{V}_{k-1}$
10	MAE1	MAE on $\mathcal{V}_{k}$
11	STD4	STD of errors on $\mathcal{V}_{k-3}$
12	STD3	STD of errors on $\mathcal{V}_{k-2}$
13	STD2	STD of errors on $\mathcal{V}_{k-1}$
14	STD1	STD of errors on $\mathcal{V}_{k}$
15	MAEDIFF3	Difference between MAE on $\mathcal{V}_{k-3}$ and MAE on $\mathcal{V}_{k-2}$
16	MAEDIFF2	Difference between MAE on $\mathcal{V}_{k-2}$ and MAE on $\mathcal{V}_{k-1}$
17	MAEDIFF1	Difference between MAE on $\mathcal{V}_{k-1}$ and MAE on $\mathcal{V}_{k}$
18	KLDATA3	Symmetric KL divergence between $\boldsymbol{X}_{k-3}$ and $\boldsymbol{X}_{k-2}$
19	KLDATA2	Symmetric KL divergence between $\boldsymbol{X}_{k-2}$ and $\boldsymbol{X}_{k-1}$
20	KLDATA1	Symmetric KL divergence between $\boldsymbol{X}_{k-1}$ and $\boldsymbol{X}_{k}$
21	KLDATA0	Symmetric KL divergence between $\boldsymbol{X}_{k}$ and $\boldsymbol{X}_{k+1}$
22	KLVAL3	Symmetric KL divergence between $\boldsymbol{y}_{k-3}$ and $\boldsymbol{y}_{k-2}$
23	KLVAL2	Symmetric KL divergence between $\boldsymbol{y}_{k-2}$ and $\boldsymbol{y}_{k-1}$
24	KLVAL1	Symmetric KL divergence between $\boldsymbol{y}_{k-1}$ and $\boldsymbol{y}_{k}$

Table 5.6: *Meta-features for adaptive mechanism prediction.*

accuracy value. Three classification methods; Random Forests (RF) (Breiman, 2001) using the Matlab TreeBagger implementation, with 50 trees and the minimum number of elements in a node set to 1, the K nearest neighbours algorithm (Altman, 1992) with three neighbours using Weka (Hall et al., 2009) implementation, and the Naive Bayes classifier (Matlab implementation) were tested as meta-classifier algorithms. The results are presented in Table 5.7.[6] It can be seen that Random Forests provides the highest AM classification accuracy. However, for Catalyst and Oxidizer datasets, the accuracy of RF is never higher than the naive majority classifier. The accuracy on Drier100 dataset, while higher than naive majority is still considerably lower than the value of ρ, identified in Figure 5.20. Therefore it is not expected that using a meta-classifier will improve the prediction accuracy on the data. This is confirmed in Table 5.8 which compares the MAEs on the datasets using the AM prediction meta-classifier with and without retrospective correction to the best performing adaptive strategy.[7] Meta-classifier shows higher accuracy than

[6] Drier200 is not included because of scarcity of training data for this batch size.

[7] These are different from the ones shown in Tables 5.3, 5.4 and 5.5, because using an AM meta classifier requires starting the prediction process from the batch #6 to be able to calculate all the required features. Before batch #6, optimal AMs are used and the errors on these batches are not included in the MAE shown in Table 5.8.

previously identified best adaptive strategy only for Catalyst100. The confusion matrices which for the RF classifier are shown in Tables 5.9-5.16. It is clear that AM4 is most often the optimal AM for the Catalyst and Drier datasets. From the confusion matrices it is seen that the predictions for these datasets are even more skewed towards AM4. The distribution of target classes is more uniform for the Drier dataset. This may be the reason why the prediction of AMs works better for this case.

Dataset	**Batches**	**Accuracy RF**	**Accuracy KNN 3**	**Accuracy NB**	**Majority Class**
Catalyst50	112	0.57	0.26	0.12	**0.59**
Catalyst100	53	0.56	0.30	0.23	**0.58**
Catalyst200	24	0.70	0.66	0.41	**0.72**
Oxidizer50	51	0.50	0.32	0.27	**0.52**
Oxidizer100	23	0.44	0.37	0.34	**0.52**
Oxidizer200	9	0.21	0.20	0.19	**0.37**
Drier50	19	**0.47**	0.31	0.19	0.34
Drier100	7	**0.32**	0.31	0.17	0.28

Table 5.7: *AM classifier average accuracy values, obtained using leave-batch-out technique. Highest values are marked bold.*

Dataset	**MAE Optimal**	**MAE Meta**	**MAE Meta RC**	**MAE Best**	**Best AS**
Catalyst50	*0.015*	0.031	0.024	**0.017**	XVSelect RC
Catalyst100	*0.021*	**0.026**	0.026	0.026	Retrain
Catalyst200	*0.026*	0.028	0.029	**0.027**	XVSelect
Oxidizer50	*0.404*	0.485	0.477	**0.444**	XVSelect
Oxidizer100	*0.472*	0.564	0.564	**0.515**	Sequence4 RC
Oxidizer200	*0.538*	0.716	0.756	**0.593**	Sequence4
Drier50	*1.41E-06*	1.47E-05	1.17E-05	**5.49E-06**	XVSelect RC
Drier100	*1.28E-06*	8.19E-06	3.06E-06	**1.61E-06**	Sequence0 RC

Table 5.8: *Results averaged over all batches obtained using AM meta-classifier with and without retrospective correction (MAE Meta and MAE Meta RC) and MAE value of the best AS (MAE Best). Results from Optimal strategy are in italics, lowest values from other strategies are in bold.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	51	159	113	6	529	858
AM1	77	198	172	9	965	1421
AM2	69	172	199	12	1464	1916
AM3	32	113	74	49	1311	1579
AM4	84	223	284	118	7517	8226
Total Predicted	313	865	842	194	11786	

Table 5.9: *Confusion matrix of Random Forest AM predictions for Catalyst50 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	5	35	27	1	332	400
AM1	16	136	115	1	552	820
AM2	5	103	202	18	839	1167
AM3	1	29	29	0	335	394
AM4	34	162	223	37	3388	3844
Total Predicted	61	465	596	57	5446	

Table 5.10: *Confusion matrix of Random Forest AM predictions for Catalyst100 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	0	2	2	0	197	201
AM1	0	0	3	0	221	224
AM2	0	1	1	0	318	320
AM3	0	0	0	0	93	93
AM4	28	12	35	0	2087	2162
Total Predicted	28	15	41	0	2916	

Table 5.11: *Confusion matrix of Random Forest AM predictions for Catalyst200 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	63	10	19	6	595	693
AM1	18	22	18	4	770	832
AM2	41	12	45	2	1026	1126
AM3	28	13	23	3	369	436
AM4	102	33	114	5	3034	3288
Total Predicted	252	90	219	20	5794	

Table 5.12: *Confusion matrix of Random Forest AM predictions for Oxidizer50 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	7	3	2	59	311	382
AM1	15	0	3	29	341	388
AM2	35	1	2	41	220	299
AM3	5	5	10	25	268	313
AM4	139	23	25	72	1234	1493
Total Predicted	201	32	42	226	2374	

Table 5.13: *Confusion matrix of Random Forest AM predictions for Oxidizer100 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	0	13	27	0	120	160
AM1	2	11	49	0	143	205
AM2	2	63	45	0	160	270
AM3	0	4	14	3	54	75
AM4	3	61	174	1	176	415
Total Predicted	7	152	309	4	653	

Table 5.14: *Confusion matrix of Random Forest AM predictions for Oxidizer200 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	541	80	25	5	146	797
AM1	122	45	31	0	211	409
AM2	32	57	68	0	180	337
AM3	30	0	0	0	12	42
AM4	135	63	120	0	472	790
Total Predicted	860	245	244	5	1021	

Table 5.15: *Confusion matrix of Random Forest AM predictions for Drier50 obtained using leave-batch-out technique.*

True\Predicted	AM0	AM1	AM2	AM3	AM4	Total True
AM0	113	50	5	11	33	212
AM1	43	85	5	1	109	243
AM2	14	49	9	6	45	123
AM3	28	35	4	0	9	76
AM4	27	109	9	4	72	221
Total Predicted	225	328	32	22	268	

Table 5.16: *Confusion matrix of Random Forest AM predictions for Drier100 obtained using leave-batch-out technique.*

5.6.1 Cost-sensitive adaptive mechanism classification

It is possible to attempt to improve the bias of above confusion matrices by specifying a misclassification cost matrix during the classification of AMs. The cost matrix should reflect the cost of AM misclassification when the optimal AM is misclassified or $h_k^{opt} \neq \hat{h}_k$.

It is proposed to set the cost of an AM classification of a single instance to the difference of the absolute values of the model's error after the deployment of selected AM and model's error after deployment of optimal AM. Thus, for an instance $(\boldsymbol{x}_\tau, y_\tau) \in \boldsymbol{\mathcal{V}}_k$:

$$\mathcal{C}_{i,opt}^{\tau} = |(f_i(\boldsymbol{x_\tau}) - y_\tau)| - |(f_{opt}(\boldsymbol{x_\tau}) - y_\tau)| \tag{5.8}$$

where $f_i(\boldsymbol{x})$ is the prediction after the deployment of AM $g_i(\cdot)$ and $f_{opt}(\boldsymbol{x})$ is the prediction after the deployment of $g_{h_k^{opt}}(\cdot)$. To get the overall classification costs matrix, $\mathcal{C}$, every combination of the cost for all h_i and h_j are averaged over the whole dataset as:

$$\mathcal{C}_{i,j} = \frac{\sum_{\tau=1}^{N} \mathcal{C}_{i,j}^{\tau} | j = h_\tau^{opt}, i \neq j}{N} \tag{5.9}$$

where $\mathcal{C}_{i,j}$ $(i, j - th$ element of $\mathcal{C})$ is the cost of deploying $g_i(\cdot)$ when the optimal AM is $g_j(\cdot)$, and $h_\tau^{opt} = h_k^{opt}$ for all $(\boldsymbol{x_\tau}, y_\tau) \in \boldsymbol{\mathcal{V}}_k$. The cost matrices for each dataset are shown in the Appendix C.

It should be noted that for every batch k the cost matrix given in this way serves to minimize the MAE on this batch. In fact, identifying cost and confusion matrices, allows to estimate the MAE when using *non-cost sensitive* AM prediction with retrospective correction without running the predictive algorithm on the dataset. Confusion matrices allow to calculate the probabilities of predicted optimal AM given the true optimal AM, $\mathcal{P}_{i,j}^{pred} = \mathcal{P}(\hat{h}_k = i | h_k^{opt} = j)$. Then, the estimated MAE after using AM prediction mechanism, is

$$MAE_{pred} = MAE_{opt} + \sum_{i,j=1}^{H} \mathcal{P}_{i,j}^{pred} \mathcal{C}_{i,j}, \tag{5.10}$$

where MAE_{opt} is the MAE of optimal adaptive strategy, and $\mathcal{C}_{i,j}$ is the entry of the cost matrix at at i-th row and j-th column. The experimentation has shown that these MAE estimates obtained using Equation 5.10 are generally close to real MAE values.

Table 5.17 gives the comparison of prediction MAE when using and not using the cost matrix[8]. It is seen that for some datasets using cost-sensitive AM classification improves the prediction results.

[8] Since the cost matrices were estimated from the same datasets on which then the predictive models were used, slight over-fitting is possible

	Dataset	**Meta-classifier**	**Cost sensitive meta-classifier**
No RC	Catalyst50	0.031	**0.031**
	Catalyst100	**0.026**	0.030
	Catalyst200	**0.028**	0.029
	Oxidizer50	**0.485**	0.490
	Oxidizer100	**0.564**	0.566
	Oxidizer200	**0.716**	0.719
	Drier50	1.47E-05	**1.18E-05**
	Drier100	8.19E-06	**7.53E-06**
RC	Catalyst50	0.024	**0.024**
	Catalyst100	**0.026**	0.027
	Catalyst200	0.029	**0.027**
	Oxidizer50	**0.477**	0.481
	Oxidizer100	**0.564**	0.570
	Oxidizer200	0.756	**0.751**
	Drier50	1.17E-05	**1.17E-05**
	Drier100	**3.06E-06**	3.59E-06

Table 5.17: *MAE values after simple and cost sensitive meta-classification. Smaller MAE values are marked bold.*

5.7 Discussion

Firstly this chapter explored the used AMs in greater detail. A limited 4-step exhaustive deployment of AMs was conducted on all datasets to generate data for AMs' analysis. For the generated data, values of novel *relative adaptation* measure were calculated and the errors of adapted models to those of un-adapted models were compared. It is observed that in majority of cases, SABLE AM4 (creating new experts and adapting experts' weights) performs better than the rest of AMs, except for the Drier dataset. AM3 (only adapting the weights) and AM0 (no adaptation) are often the worst AMs. How well the AMs perform is however very dependant on the dataset.

Subsequently, this chapter investigated the problem of AM selection during the prediction of streaming data in batch mode. Different adaptive strategies were introduced for this purpose. These could be grouped into fixed adaptive strategies and flexible ones. Fixed adaptive strategies deploy the same set of AMs on every batch, whereas flexible adaptive strategies may deploy different sets of AMs on different batches. Flexible adaptive strategies such as greedy optimal AM deployment, using cross-validation for AM selection, retrospective model correction and prediction of optimal AM were introduced. Optimal adaptive strategy, that deployed the AM which minimizes the error on each subsequent batch was the best strategy in all of the cases, however it is unattainable in practice, as the optimal AMs are not known. Retrospective model correction can be used together with other adaptation strategies and often improves predictive error, especially for slower adaptive mechanisms (Bakirov et al., 2016). Using cross-validation on the previous batch to select the AM for the subsequent batch, often coupled with retrospective correction, has proved to be the most successful adaptive strategy. It performed universally well and was often the best strategy for all the datasets. As it was seen that the optimal adaptive strategy provides the best results, a meta-classifier which predicts these optimal AMs was constructed using several popular classification algorithms. However, almost none of these could achieve the accuracy rates needed to surpass the best performing adaptive strategies.

Chapter 6

Conclusions

6.1 Thesis summary

In recent times many advances have been made in adaptive predictive modeling for streaming data. Powerful adaptive mechanisms are being developed and integrated into predictive algorithms. Employing several adaptive mechanisms, often operating on different levels, for a single predictive algorithm has become a current trend. This has allowed the development of powerful and flexible adaptive predictive methods.

Despite these advances and popularization of the usage of multiple AMs, there is a noticeable gap in research about the reasons of AMs choices and their synergy while used together. In other words the design of adaptation element of predictive methods is often intuitive or ad-hoc. This fact creates a concern that, even while the predictive methods often show good results on test data, their adaptation strategies may be not optimal and that better results may be achieved with other, more appropriate strategies.

The aim of the thesis is to offer a multifaceted research with the objective of identifying ways to improve the predictive accuracy of the methods with multiple AMs. This thesis offers background information on learning and adaptation on streaming data and categorization of adaptive mechanisms, formalises the adaptive mechanisms, investigates the necessity of a careful selection of adaptive strategy and proposes novel adaptive strategies and techniques. In addition to offering the results which can be already used for designing adaptive methods, in general it could serve as a catalyst for further research in this area.

6.2 Findings and contributions

It may be useful to separate the findings of the thesis according to the goals set in the Chapter 1.

6.2.1 Categorisation and formalisation of adaptive mechanisms

Relevant works dating from 90-s until recent years were surveyed with the results presented in Chapter 2. The focus was to analyse adaptive mechanism part separately from the prediction algorithm, and create a categorisation, where categories share common characteristics. The hierarchical categorization described in Chapter 2 provides a clear overview of adaptive mechanisms and serves as a basis for the experimentation in subsequent chapters. Chapter 4 presents a formalisation of the "adaptive mechanism" notion as a function which produces an adapted predictive model as an output, which allows a clear description of adaptive predictive models. These contributions are general and can be used in further research outside of this project.

6.2.2 Analysing the importance of adaptive mechanism selection

Adaptive strategies or similar concepts are scarcely mentioned in the relevant literature. As noted before, the strategies used in many works are based on intuitive reasoning, which may lead to the generation of poorly performing AM sequences. One of the contributions of this thesis is an empirical investigation on whether the choice of AM sequence significantly matters for the predictive accuracy of the algorithm (Chapter 4). As a separate contribution, this includes the development of the adaptive batch learning algorithm SABLE which allows the generation of flexible AM sequences, devising the experiments, conducting them on three real datasets from the process industry and analysing the results.

In the experiments, random adaptive sequences are compared to the greedy optimal sequence for all of the considered datasets. The experiments show that the optimal adaptation sequence provides significantly better results. Hence, it is possible to conclude that the AM selection and thus the choice of adaptation strategy is indeed an important factor for the predictive accuracy of the algorithm.

6.2.3 Investigation of adaptive mechanisms and adaptive strategies effects

Methods to analyse AMs' effects and the analysis of the AMs used in the thesis in detail are presented in Section 5.3. For this purpose, a *relative adaptation* ratio is introduced. Relative adaptation's properties provide a clear insight into the behaviour of an adaptive mechanism. Using the data generated by a 4-step limited exhaustive AM deployment as described in Section 5.2, it is observed that the AMs perform differently well for different datasets, although some common trends are noticeable, e.g. the generally good performance of certain AMs.

Chapter 5 further gives examples of commonly used adaptive strategies. It is noted that most of the popular adaptive strategies are fixed, in a sense that the same combination of AMs is deployed every time the adaptation of the model is performed. Experiments using common (fixed) adaptive strategies are conducted. Further contributions of the thesis are several novel flexible adaptive strategies – cross-validatory AM selection (*XVSelect*), retrospective model correction and meta-classifier for AM selection, which are described in that chapter. The former two strategies, often in combination, provide better results than the traditional adaptive strategies most of the times. The latter strategy didn't produce better results than other strategies, which could be caused by the way the meta-classifier was constructed. The empirical comparison of fixed and flexible strategies, together with the conclusion that flexible adaptive strategies generally perform better than the fixed ones is another contribution of the thesis. It was attempted to find the mapping between changes in data and the most appropriate AMs. While there was no straightforward answers to this question, the intuitive reasons why particular adaptive strategies and mechanisms perform better than others were outlined.

6.2.4 Research into new experts' addition for streaming classification ensembles

Training of new experts is an important part of many ensemble methods for streaming data. In the Appendix A, this issue is investigated on the example of DWM algorithm. Specifically, the conditions to add a new expert, as well as training data for the new experts were analysed. Empirical tests were conducted on 26 synthetic two-dimensional datasets with various types of changes, as well as two real world datasets from the electricity consumption domain. The results have demonstrated that some of the suggested modifications, particularly increasing the training data of the newly created experts at the cost of delaying their addition to the ensemble, show better or comparable accuracy as the original DWM and considerably decrease the number of created experts.

6.3 Future research

This thesis has highlighted issues regarding the usage of multiple adaptive mechanisms for predictive methods on streaming data and addressed some aspects of this area. It is hoped that this work can facilitate further research in this area, with the goal of finding adaptive strategies which minimize the predictive error on streaming data. As seen from the previous chapters, this is an important task, which can seriously affect the performance of the algorithm. Current project is so far one of the few in the area which means that the possibilities for future research are abundant.

An obvious direction of work is the improvement of the meta-classifier to select AMs. Selecting an optimal AM for the data not yet seen is not an easy task, however its accuracy may possibly be improved. Using different sets of input features, different classification algorithms and methods which deal with imbalanced training data may be useful for this purpose. On the other hand in this thesis, the meta-classifier attempted to find a greedy optimal adaptive sequence, which may not be necessarily the adaptive sequence which minimizes the predictive error over the range of incoming batches. A multi-step meta-classifier could be considered to optimize the performance for this purpose.

Another area of research may be the improvement of categorization of adaptive mechanisms which may enable rule-based approaches for their selection. This would create whitebox models which are easy to interpret, potentially illuminating hidden details of the process. Such rule-based models will make it easier for technicians and engineers to adapt the predictive models used in industry.

The *model trees* resulting after the deployment of different AMs on the same model is an interesting research direction as well. The leaves of these trees may be combined to produce a single output. Multiple leaves may be retained for propagation.

The proposed method has used the same algorithm as a base learner. It could be considered to use a different algorithms for different experts. This approachcan capture multiple types of input-output relationship as well as further increase the diversity of the ensemble.

One focus of the thesis was proposing heuristics for the AM sequences generation and empirical evaluation of results. This research could be further extended with a theoretical analysis of adaptive strategies. Error bounds of adaptive strategies, the speed of adaptation and the conditions that the data stream need to meet for satisfactory adaptation are examples of topics that could be investigated.

Appendices

Appendix A

Addition of new experts to adaptive classification ensembles

A.1 Elements of online expert ensemble creation

A.1.1 Condition for adding of an expert

Expert addition often brings the strongest possible change to the predictive model. To assess the effects of criteria to add experts and of the training sets of newly added experts, DWM (Section 2.5.4) algorithm is observed and modified. To help with this assessment, in the following, terms *reaction time* - the minimum number of observations, after which algorithm will react to observed change by creating an expert and *convergence time* - number of observations, after which algorithm will converge (total weight of the experts which are trained on the new concept is larger than the total weight of the experts which were trained on previous concepts) to new concept, are introduced.

Condition for adding an expert largely determines the *reaction time* of an algorithm, and thus plays a significant role in its *convergence time* as well. Reviewed model reacts to misclassifications by creating a new expert. Initially, it is suggested to add an expert every time when the prediction of current ensemble is false. This provides fast reaction time but may, in noisy conditions, result in adding many unnecessary and inaccurate experts. To deal with this problem, in (Kolter & Maloof, 2007) authors suggest that, in noisy domains or for large experiments, only every Ω-th example could be taken into consideration, which reduces the number of created experts in proportion to Ω. The drawback of having $\Omega > 1$ is a possibility of slower reaction to the change. This is best manifested during a sudden drift, where, in the worst case, the reaction time is Ω.

To reduce the effects of noise in a more deterministic way, the averaging window condition for expert creation as an alternative to having $\Omega > 1$ is proposed (In (Bach & Maloof, 2008) a similar condition is used to substitute learners). Here, the strategy is based on the decision of creating an expert from $\boldsymbol{x}_n$ (n-th datapoint) not only as a result of $\boldsymbol{x}_n$'s classification, but on the basis of accuracy in the window of the last l elements, with $\boldsymbol{x}_n$ being the last element of l. An expert trained from $\boldsymbol{x}_n$ is added if the average accuracy of the ensemble in the window is less than fixed threshold value z. If it is assumed that the change causes algorithm to always misclassify incoming data, then the reaction time to the change in this case can be calculated to be at most $l(1-\zeta)$ rounded up.

The choice of the threshold may be difficult for unknown data. Also, for the datasets where average accuracy may vary with the time, for example due to changing noise levels, using the above static threshold might result in creation of many unnecessary experts or not creation of experts when needed. For this work, a similar algorithm with dynamic threshold value, called"maximum accuracy threshold window" (MTW) is introduced. The dynamic threshold here is similar to the one used in DDM change detector (Gama et al., 2004). While classifying incoming data the maximum value of $\mu_{acc} + \sigma_{acc}$ is recorded, where μ_{acc} is mean accuracy and $\sigma_{acc} = \sqrt{\frac{\mu_{acc}(1-\mu_{acc})}{l}}$ is the standard deviation of the Bernoulli process. New expert is created a when the condition $\mu_{cur} - \sigma_{cur} < \mu_{max} - m\sigma_{max}$ is met. Here μ_{cur} and σ_{cur} are mean accuracy and standard deviation of current window and μ_{max} and σ_{max} are mean accuracy and standard deviation of the window where the maximum value of $\mu_{acc} + \sigma_{acc}$ was recorded. Parameter m is usually set to 3. After creation of new expert, the maximum values are reset. Using window based conditioning is illustrated in the Figure A.1b.

A.2 Training data for new experts

Assuming uniform class label distribution, training an expert from a single data instance means that this expert will assign the label it has been trained on to all other samples which makes its initial accuracy $1/J$ in the case of J-label classification problem. Low accuracy of experts trained on insufficient amount of data, also discussed in (Žliobaitė & Kuncheva, 2010), combined with the high weight of the new expert, may result in noticeable negative effect on the accuracy. To counter this it is possible to use a delay in reaction time to train the new expert on more examples before using it for predictions. The simplest option is to train an expert on l datapoints after its creation and only then add it to the ensemble, as in (Stanley, 2003). In this research this will be denoted "mature" experts (MATEX) approach. To prevent multiple reactions to one change, during the time that expert is being "matured", no new experts are introduced. When the expert is added to the ensemble, it is better trained and thus more accurate than the expert which is created from a single data instance. The reaction time for a change in this case is l. Another advantage of this approach is reducing the effect of noise on the created expert.

One possibility to reduce the number of unnecessary experts created in this way is assessing their performance. A sufficient condition for expert to benefit the ensemble independent of this expert's weight is predicting better than ensemble. So before adding it to the ensemble its performance can be compared with the that of the ensemble in the window of size l. Comparison

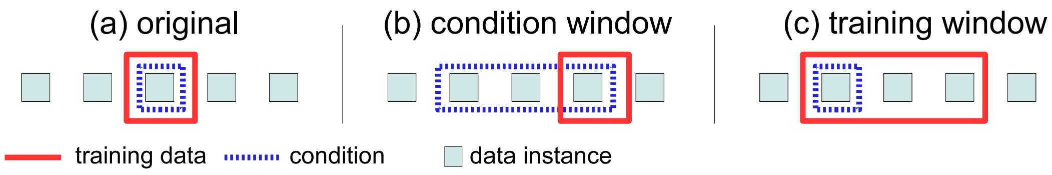


Figure A.1: *Using windows for expert adding condition and training data of a new expert.*

strategy is similarly used in (Bach & Maloof, 2008). The comparison can be done in various ways; comparing the *prequential* (Dawid & Vovk, 1999) accuracies of the expert and ensemble, or constructing certain test and training sets from the datapoints in the window and using cross-validation. Prequential accuracy[1] is calculated by making predictions on data instances *before* training on them. If the validation is successful, then the new expert which has been trained on the whole window is added to the ensemble. Here the reaction time is l. To prevent multiple reactions to one change, during the time that expert is being "validated", no new experts are introduced. Here, the effect of noise is further reduced - when the data suddenly becomes noisy, newly created experts will probably not predict better than existing ensemble and thus will be discarded. Validation approach can be combined with MATEX allowing the expert to train on l_{mature} datapoints (maturing window), before starting the comparison on l_{val} datapoints (validation window). This might help prevent the premature removal of experts but will accordingly increase reaction time to $l_{mature} + l_{val}$. It must be noted that this approach requires additional computational effort for the validation. Using window for the data basis of new expert is illustrated in the Figure A.1c.

A.3 Experimental results

A.3.1 Methods description

Experiments were performed with different variations of the methods described in the Section A.1. The implemented window based condition schemes from the section A.1.1 are presented in Table A.1. Several variations of the different data basis methods described in the section A.2 were tested. They are presented in the table A.2. It was also experimented with the periodical expert additions with periods Ω of 5, 7, 10 11, 25 and 50 (codenames DWM_P5, DWM_P7, DWM_P10, DWM_P11, DWM_25, DWM_50). In the implementation of original algorithms used in this theses, WIN and MTW a new expert was created from the single data instance. Static (Kolter & Maloof, 2007) and dynamic (Kolter & Maloof, 2005) weighting systems with various starting weights and weight update factors, β, for adding new weights and reducing the existing ones were tested. These are summarised in the Table A.3.

[1] Hereon also referred to as simply "accuracy".

Threshold type	z/m	l	**Codename**
Fixed	0.5	5	WIN_5_0.5
Fixed	0.5	10	WIN_10_0.5
Fixed	0.5	25	WIN_25_0.5
Fixed	0.5	50	WIN_50_0.5
Variable	3	5	MTW_5
Variable	3	10	MTW_10
Variable	3	25	MTW_25
Variable	3	50	MTW_50

Table A.1: *Experiments with window based conditions to add an expert. Here, z is fixed accuracy threshold to add an expert, m factor for variable threshold calculation and l is length of the window.*

l_{mature}	l_{val}	**Validation type**	**Codename**
5	0	N/A	MATEX_5
10	0	N/A	MATEX_10
25	0	N/A	MATEX_25
50	0	N/A	MATEX_50
0	5	Prequential	PVAL_5
0	10	Prequential	PVAL_10
0	25	Prequential	PVAL_25
0	50	Prequential	PVAL_50
5	5	Prequential	MATEX_ PVAL_5_5
0	10	10xCross-validation 1 train 9 test	XVAL_10_10
0	10	5xCross-validation 2 train 8 test	XVAL_10_5
0	10	3xCross-validation 3 train 7 test	XVAL_10_3
0	10	2xCross-validation 5 train 5 test	XVAL_10_2
0	10	Leave-one-out cross-validation 9 train 1 test	XVAL_10_1
0	25	Leave-one-out cross-validation 24 train 1 test	XVAL_10_1
0	50	Leave-one-out cross-validation 49 train 1 test	XVAL_10_1

Table A.2: *Experiments with the data basis for experts and their validation. Here l_{mature} is size of the maturing window and l_{val} is size of validation window.*

A.4 Results on synthetic data

26 two-dimensional data sets with various properties to examine the behaviour of the algorithms in different situations were generated, including rotating hyperplane (Hulten et al., 2001) data and Gaussians with different type of changes - switching between two data sources (Narasimhamurthy & Kuncheva, 2007), one Gaussian passing through the other one and returning, Gaussians moving together in one direction and returning (Sahel et al., 2007) which are visualised in Figure A.2. Experiments were performed with various magnitudes of changes and levels of artificial noise

Weighting system	New expert weight (static) / γ (dynamic)	β
Static	1	0.3
Static	1	0.5
Static	1	0.7
Dynamic	0.3	0.3
Dynamic	0.2	0.5
Dynamic	0.1	0.7

Table A.3: *Starting weights and weight update factors, β, used in experiments.*

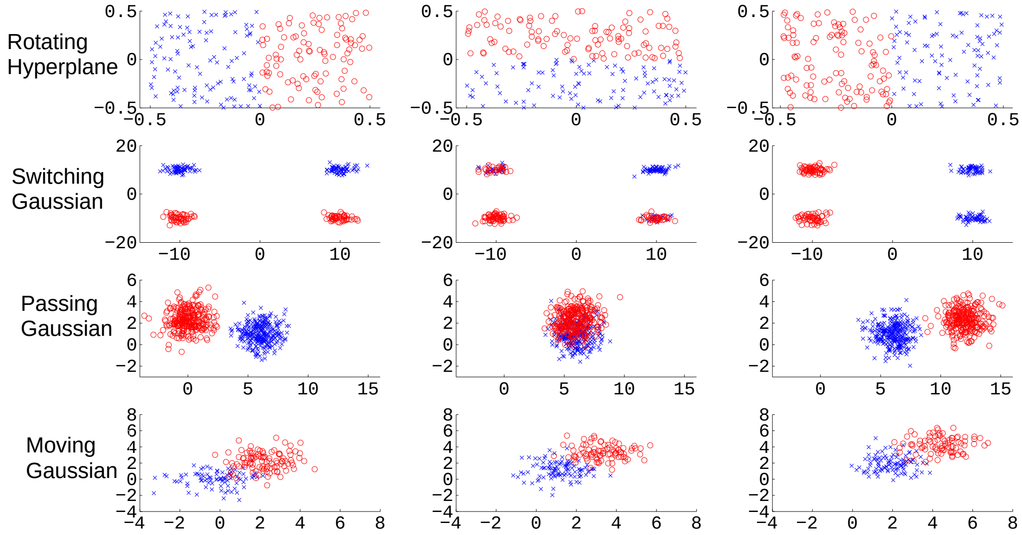


Figure A.2: *Changes in experimental datasets. From left to right: data in the start, in the middle, in the end (before possible return to starting position).*

and decision boundaries overlap (see Table A.4).

The accuracy of the tested algorithms, as well as in the number of experts they create during the classification process was assessed. After the testing on all 26 of described datasets with described methods, it was concluded that among them, the best ones in terms of average prequential accuracy are MATEX methods. Particularly, MATEX_25 with static weighting of 0.7 showed the best average test and prequential accuracy among all of the methods on 26 datasets, and the best average accuracy on 7 datasets. In fact, top 4 best performers in average accuracy were variations of MATEX with different window lengths and weighting systems. The leader in average accuracy among the family of methods which use validation was XVAL_25_1 (leave-one-out crossvalidation on a window of 25 data instances) with static weighting of 0.7 which had average accuracy 0.02% less than the leader. From the methods which involve changing of the condition of adding an expert, MTW_5 with dynamic weighting of 0.7 has shown the best performance, 0.04% less

Num.	Data type	Instances	Classes	Drift	Noise/overlap
1	Hyperplane	600	2	2x50% rotation	None
2	Hyperplane	600	2	2x50% rotation	10% uniform noise
3	Hyperplane	600	2	9x11.11% rotation	None
4	Hyperplane	600	2	9x11.11% rotation	10% uniform noise
5	Hyperplane	640	2	15x6.67% rotation	None
6	Hyperplane	640	2	15x6.67% rotation	10% uniform noise
7	Hyperplane	1500	4	2x50% rotation	None
8	Hyperplane	1500	4	2x50% rotation	10% uniform noise
9	Gaussian	1155	2	4x50% switching	0-50% overlap
10	Gaussian	1155	2	10x20% switching	0-50% overlap
11	Gaussian	1155	2	20x10% switching	0-50% overlap
12	Gaussian	2805	2	4x49.87% passing	0.21-49.97% overlap
13	Gaussian	2805	2	6x27.34% passing	0.21-49.97% overlap
14	Gaussian	2805	2	32x9.87% passing	0.21-49.97% overlap
15	Gaussian	945	2	4x52.05% move	0.04% overlap
16	Gaussian	945	2	4x52.05% move	10.39% overlap
17	Gaussian	945	2	8x27.63% move	0.04% overlap
18	Gaussian	945	2	8x27.63% move	10.39% overlap
19	Gaussian	945	2	20x11.25% move	0.04% overlap
20	Gaussian	945	2	20x11.25% move	10.39% overlap
21	Gaussian	1890	4	4x52.05% move	0.013% overlap
22	Gaussian	1890	4	4x52.05% move	10.24% overlap
23	Gaussian	1890	4	8x27.63% move	0.013% overlap
24	Gaussian	1890	4	8x27.63% move	10.24% overlap
25	Gaussian	1890	4	20x11.25% move	0.013% overlap
26	Gaussian	1890	4	20x11.25% move	10.24% overlap

Table A.4: *Synthetic datasets used in experiments. Column "Drift" specifies number of drifts, the percentage of change in the decision boundary and its type.*

than the best method. The best performing periodical adding algorithm was DWM_P7 with dynamic weighting of 0.5. Finally the 0.7 weighting showed the best results for original algorithm with period $\Omega = 0$. Table A.5 presents methods with top 20, bottom 5 (128-132), and original DWM with $\beta = 0.5$ (127-th place) average accuracy on 20 datasets. It is possible to see that the difference between values is relatively small, in tenths of percent, but the difference between the largest and the smallest accuracy value is 16.2%.

In the Table A.6 the results of different described variations of the original method with the results of original method were compared. Window length of 10 and $\beta = 0.5$ was used for all of the methods. XVAL is leave-one-out cross-validation. Threshold in the WIN is 0.5. Again, MATEX methods with dynamic weighting emerge as narrow leader in terms of average prequential accuracy. The accuracy rates are quite similar, but a noticeable decrease in the number of total created experts and the average ensemble size is observed. Validation methods PVAL and

Rank	**Method**	**Weighting**	β	**Average accuracy**	**Standard deviation**	**Average number of experts created**	**Average ensemble size**
1	MATEX_25	Static	0.7	0.901	0.062	29.308	7.009
2	MATEX_25	Static	0.5	0.900	0.063	29.346	5.015
3	MATEX_25	Dynamic	0.5	0.899	0.063	29.154	3.980
4	MATEX_50	Static	0.5	0.899	0.063	17.962	4.104
5	XVAL_25	Static	0.7	0.899	0.064	13.115	3.616
6	MATEX_50	Static	0.7	0.898	0.063	18.038	4.966
7	MATEX_10	Static	0.7	0.898	0.066	55.538	10.273
8	MATEX_50	Static	0.3	0.898	0.063	18.115	3.016
9	MATEX_50	Dynamic	0.3	0.898	0.063	17.962	2.658
10	XVAL_25	Static	0.5	0.898	0.064	13.269	2.592
11	MATEX_10	Dynamic	0.7	0.898	0.063	54.769	7.192
12	MATEX_10	Dynamic	0.5	0.897	0.064	54.500	4.753
13	MATEX_25	Static	0.3	0.897	0.064	31.038	3.380
14	MATEX_5	Dynamic	0.7	0.897	0.064	80.923	8.883
15	XVAL_10_1	Static	0.7	0.897	0.064	20.154	4.815
16	MATEX_25	Dynamic	0.3	0.897	0.064	29.538	2.756
17	MTW_5	Dynamic	0.7	0.897	0.064	58.846	6.964
18	MTW_10	Static	0.7	0.897	0.065	39.808	7.685
19	MTW_5	Static	0.7	0.897	0.064	57.962	10.524
20	MATEX_25	Dynamic	0.7	0.897	0.063	29.423	5.468
127	DWM_P1	Static	0.5	0.884	0.075	156.115	14.973
186	DWM_50	Dynamic	0.7	0.828	0.114	4.654	2.639
187	WIN_25	Dynamic	0.5	0.828	0.184	151.308	3.052
188	WIN_25	Dynamic	0.3	0.822	0.165	176.923	2.430
189	DWM_P1	Dynamic	0.3	0.761	0.189	339.577	5.064
190	WIN_5_0.5	Dynamic	0.3	0.739	0.221	483.423	5.267

Table A.5: *Methods with top 20, bottom 5 (128-132), and original DWM with* $\beta = 0.5$ *(93-th place) average accuracy on 20 datasets.*

XVAL further reduce the number of total created experts and average ensemble size, while having slightly lower accuracy rates than the leaders and requiring additional computation for validation purposes. To benchmark our results against non-ensemble methods tests with a simple online Naive Bayes classifier without any forgetting, state of the art change detectors DDM (Gama et al., 2004) and EDDM (Baena-García et al., 2006) and Paired Learners method with window size 10 and threshold 0.1 (Bach & Maloof, 2008) were conducted. As expected, online Naive Bayes performs noticeably worse than adaptive methods. Change detectors and paired learners show slightly lower but comparable accuracy to MATEX methods.

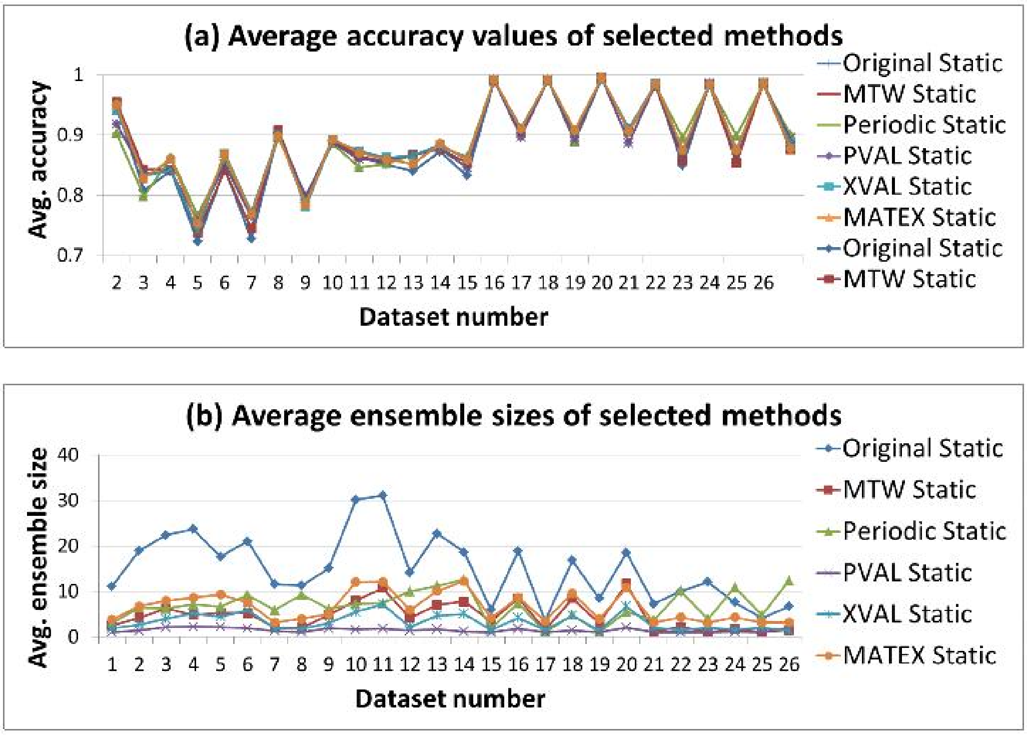
The top performers on some datasets can be different than the average leaders. For instance, validation methods perform better on the dataset with passing Gaussian with 4 drifts. Here, XVAL

Method	**Average accuracy**	**Std. deviation of avg. accuracy**	**Average total created experts**	**Average ensemble size**
MATEX dynamic weighting	0.897	0.064	54.50	4.75
MATEX static weighting	0.895	0.066	55.15	6.69
MTW dynamic weighting	0.894	0.067	40.38	3.31
MTW static weighting	0.889	0.071	40.54	4.63
DWM periodical dynamic weighting	0.894	0.064	15.58	5.42
DWM periodical static weighting	0.891	0.066	15.81	6.76
XVAL dynamic weighting	0.890	0.068	22.35	2.31
XVAL static weighting	0.893	0.068	21.12	3.31
PVAL dynamic weighting	0.888	0.066	5.23	1.53
PVAL static weighting	0.889	0.066	4.65	1.48
WIN dynamic weighting	0.880	0.073	51.04	3.39
WIN static weighting	0.881	0.072	24.85	4.71
Original dynamic weighting	0.867	0.091	181.15	7.94
Original static weighting	0.884	0.075	156.12	14.97
PAIRED LEARNER	0.891	0.069	4.5	2
DDM	0.88	0.077	2.27	1
EDDM	0.89	0.067	1.92	1
NAIVE_BAYES	0.807	0.137	1	1

Table A.6: *Results on 26 synthetic datasets, averaged. Window length $l = 10$ and $\beta = 0.5$ was used for all of the methods. XVAL is leave-one-out cross-validation. Threshold z in the WIN is 0.5.*

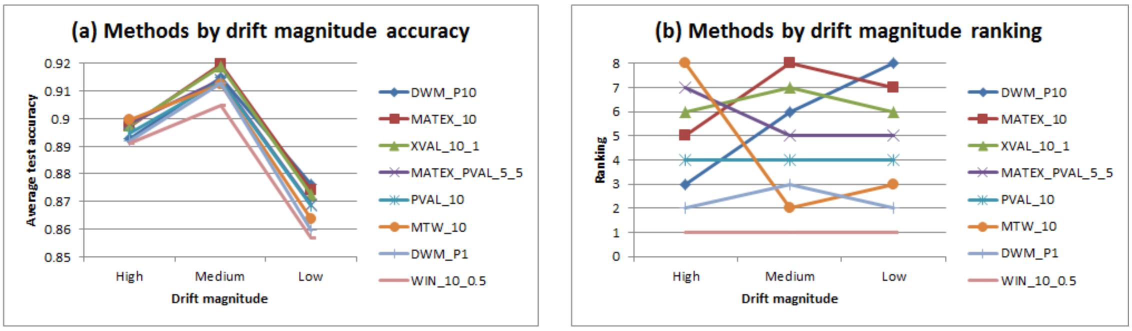
with dynamic static weighting shows the best accuracy among the methods compared above - 87.2% which is 0.8 % higher than the accuracy of the leader. Intuitively, this can be explained with a large proportion of class intersection area, where the expert creation is not beneficial, and two intersection-free areas where high accuracy experts can be created. In general, expert checking is beneficial for the datasets with variable noise or decision boundary intersection. Figure A.3 gives an insight on the performance for selected methods with static weighting from the Table A.6 on individual datasets.

Methods vs drift magnitude. To analyse performance of different methods and weighting strategies in different scenarios the datasets are divided to different groups based on the magnitude of concept drift ($d\%$) (slow if $d\% < 15$, medium if $15 \leq d\% < 35$, fast if $35 \leq d\%$), percentage of noise or classes' boundaries overlap (none/10%/varying) and number of classes (2/4). Several methods' performances are compared in terms of average accuracies, averaged by the magnitude of drift. DWM_P1, DWM_P10, WIN_10_0.5, MTW_10, MATEX_11, PVAL_10, MATEX_PVAL_5_5, XVAL_10_1 with static weighting and $\beta = 0.5$ (Figure A.4) are compared. Interestingly, MTW_10 which shows the best performance at high magnitude changes, but is one of the worst ones otherwise. Periodical DWM_P10 performance increases as the magnitude of


(a) Average accuracy values of selected methods

(b) Average ensemble sizes of selected methods

Figure A.3: *Average accuracy values and ensemble sizes for selected methods.*


(a) Methods by drift magnitude accuracy

(b) Methods by drift magnitude ranking

Figure A.4: *Results grouped by the drift magnitude.*

the drift decreases which explained by the intuition that as the magnitude of change decreases, the importance of the quick reaction decreases as well. XVAL_10_1 shows good results, and WIN_10_0.5 the worst results overall.

Methods vs noise level. Here the same methods as in above paragraph versus noise level (Figure A.5) are compared. It is ovserved that DWM_P10 shows the best performance at fixed low level noise, but worse performance otherwise. This could be explained by high percentages of overlap in "mixed" category, during which creation of experts can not be effective. This is also confirmed by the good performance of validatory strategies such as XVAL_10_1 in "mixed"

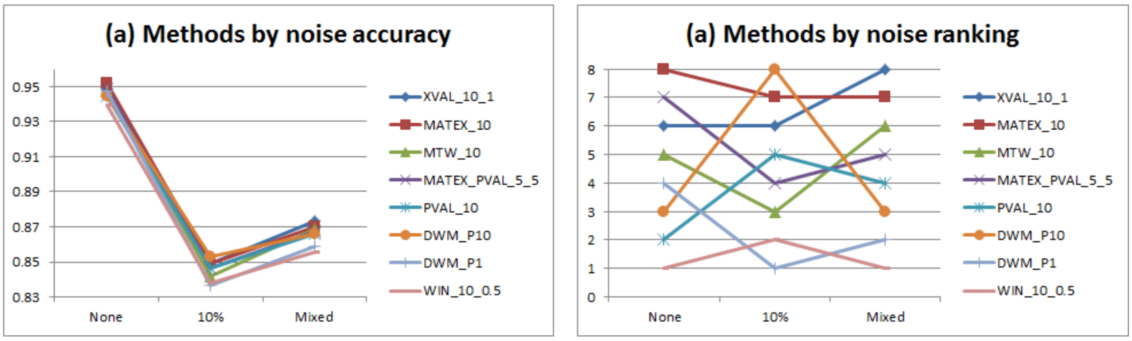


Figure A.5: *Results grouped by noise levels.*

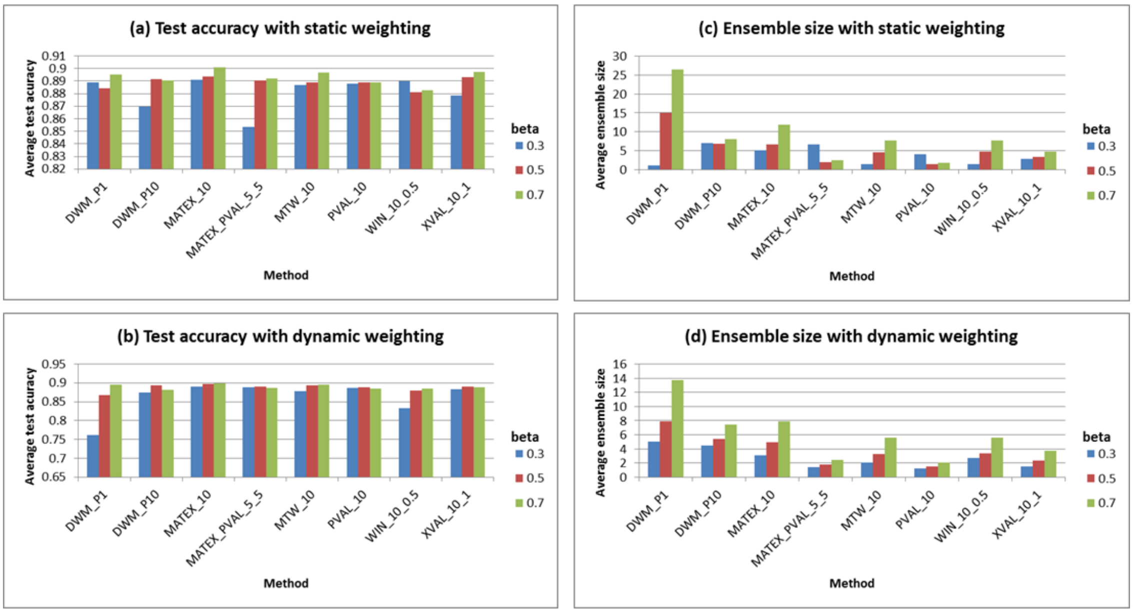


Figure A.6: *Results grouped by β value.*

category. Original DWM has the middle ranking in performance when there is no noise, but drops to lower rankings when noise exists. WIN_10_0.5 is again the worst performer.

Effects of different weighting styles and values of β. Effects of different weighting styles and β values are visualised in the Figure A.6. It can be seen that accuracy rises with the values of β, and sometimes drops noticeably when $\beta = 0.3$. Dynamic weighting shows less difference in accuracy values between different values of β. This could be related to the respective choice of γ (table A.3). Intuitively, average ensemble size should increase together with β. This appears to be not always the case in the case of static weighting. This can be related to the fact that low β could make the experts be deleted before they can be trained on enough samples to show reasonable performance, which decreases the performance of the algorithm and in turn increases the number of created experts.

Effects of different window/period sizes. Increasing window/period size increases the reaction time of algorithms. Accuracy of selected algorithms was compared using window sizes of 10,

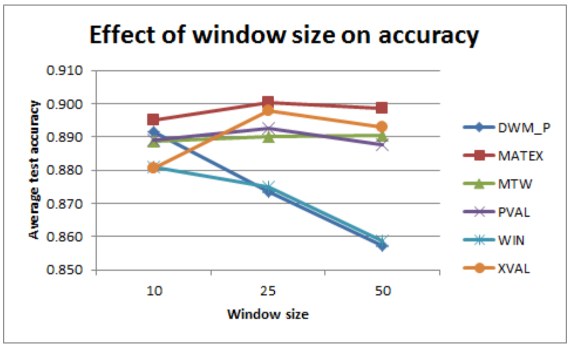


Figure A.7: *Results grouped by window size.*

25 and 50 (static weighting, $\beta = 0.5$). Effects of different choices of window size are visualised in the Figure A.7. It can be seen that for periodical DWM and window with fixed threshold, accuracy clearly decreases with the increase of window or period sizes. Accuracy of MTW is very slowly increasing with the window size increase. MATEX, PVAL and XVAL show increase in accuracy from the window size increase from 10 to 25 and a gradual drop, when the window size is increased to 50.

A.5 Results on real data

Electricity price rise/fall forecast. Elec2 dataset (Harries, 1999) is a widely used benchmark for online classification with concept drift. The task is to predict whether the price of the electricity in the Australian state of New South Wales will rise or fall. The dataset includes five numerical attributes; the day of the week, the 30-minute period of the day, the demand for electricity in New South Wales, the demand in Victoria, and the amount of electricity to be transferred between the two. It consists of 45,312 instances collected at 30-minute intervals from May 1996 to December 1998, however the first part of which contain unknown values. For the experiments in this theses, the last 27,887 rows which do not have any unknown values (May 1997 - December 1998) were used. It is assumed that the data contains some drifts, due to seasonality and other factors. Results with methods with static weighting and $\beta = 0.5$ can be seen in the Table A.7. It is interesting to note that for this dataset $\beta = 0.5$ provides the best prequential accuracy results, and also smaller window sizes seem to perform better. The dependence of performance of selected methods with the static weighting on β can be seen in the Figure A.9. Here, selecting $\beta = 0.5$ shows better results when compared with $\beta = 0.3$ and $\beta = 0.7$. the Accuracy of static and dynamic weighting systems are comparable.

Predicting hours of day using electricity demand. Here a dataset (PowerItaly) is analysed, containing 3741 instances which describes hourly power supply of an Italian electricity company from two sources: power supply from main grid and power transformed from other grids (Zhu, 2010; Keogh et al., 2011). This data contains records from 1995 to 1998. In the conducted

Method	**Prequential accuracy**	**Average ensemble size**
DWM_P1	0.899	11.5
MTW_5	0.898	6.64
MTW_10	0.897	5.47
MATEX_5	0.892	7.89
MATEX_10	0.888	6.45
WIN_5_0.5	0.883	8.24
PVAL_5	0.880	3.16
XVAL_10_1	0.876	3.21
PVAL_10	0.875	2.92
DWM_P5	0.871	10.66
MATEX_PVAL_5_5	0.865	2.13
WIN_10_0.5	0.862	6.89
DWM_P10	0.855	11.25
MTW_25	0.826	3.67
MTW_50	0.808	2.52
MATEX_25	0.800	4.27
PVAL_25	0.790	3.06
XVAL_25_1	0.790	3.32
MATEX_50	0.779	3.43
WIN_25	0.773	8.97
PVAL_50	0.771	2.54
XVAL_50_1	0.771	2.59
DWM_P25	0.757	14.08
WIN_50	0.747	7.43
DWM_P50	0.738	13.90
DDM	0.785	1
EDDM	0.84	1
PAIRED_10_01	0.886	2
NAIVE_BAYES	0.669	1

Table A.7: *Prequeuntial accuracy on Elec2 dataset. Static weighting,* $\beta = 0.5$

experiment, only hours 03:00, 10:00, 17:00 and 21:00 and try to predict the correct hour based on supply values. From the Figure A.8 (here only one feature is shown, other's behaviour is similar) it can be clearly seen that drift can be caused by the seasons, weather, and the differences between working days and weekend. There is also apparent class boundary intersection of various proportions. To increase the challenge a number of data instances from the dataset is randomly deleted. Results with methods with static weighting and $\beta = 0.5$ can be seen in the Table A.8. It is interesting to see the good performance of DWM_7. This might be related to the weekly cycle in the data. The dependence of the static weighting on β (except the methods with the window sizes of 25 and 50) can be seen in the Figure A.10. As with the synthetic data, higher β values provide better results in many cases.

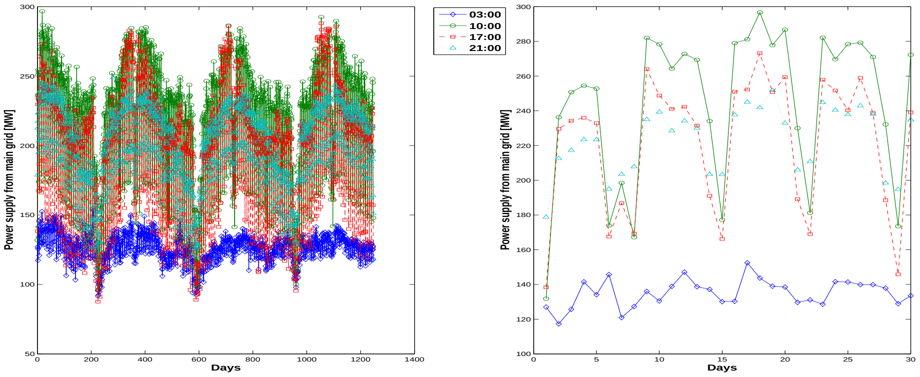
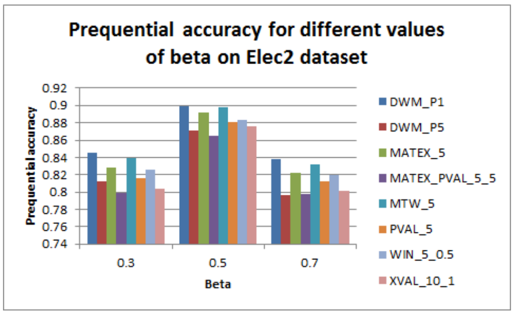


Figure A.8: *Power supply from main grid. Left: all data. Right: the first 30 days.*



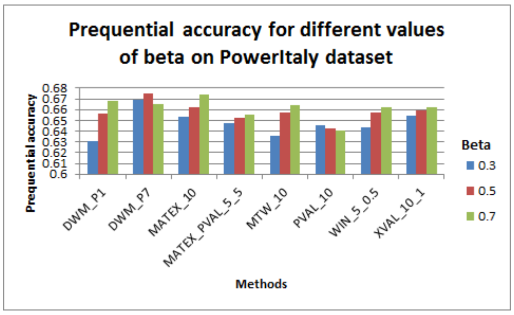


Figure A.9: *Results on Elec2 with different values of β.*

Figure A.10: *Results on PowerItaly with different values of β.*

A.6 Summary of experimental results

From the experimental results the following conclusions can be drawn.

- Accuracy-wise there is no single method which performs best in all of the conditions. MATEX often shows best performance, while XVAL combines good performance with low ensemble sizes. Performance of the original DWM is found to be inferior to that of its modifications in many cases. There are also no single best algorithm settings (choice of β and weighting strategy), although the choice of $\beta = 0.3$ often shows considerably inferior accuracy rates.
- Predictive accuracy rates of different methods are often very close to each other with differences from 0.1% to 2%. Thus the significance of difference in accuracy are debatable. However McNemar's significance test (Kuncheva, 2004b) suggests that many of the classification results are in fact significantly different.
- One of the goals of the above modifications of the original algorithm was reducing the ensemble sizes, while having comparable or better accuracy rates. This was clearly achieved. Methods which employ the validation of the newly created experts, such as PVAL or XVAL have particularly low ensemble sizes, even lower than periodical DWM. MATEX

Method	**Prequential accuracy**	**Average ensemble size**
DWM_P7	0.675	17.61
MATEX_25	0.671	5.47
DWM_P5	0.664	18.03
MATEX_10	0.663	9.09
XVAL_25	0.663	3.15
MATEX_50	0.661	3.62
XVAL_10_1	0.659	3.39
MTW_10	0.657	6.11
DWM_P10	0.657	15.71
WIN_5_0.5	0.657	26.03
DWM_P1	0.656	33.39
MTW_5	0.656	9.31
PVAL_25	0.653	1.27
MATEX_PVAL_5_5	0.652	2.90
MTW_25	0.651	3.23
MATEX_5	0.651	12.88
PVAL_50	0.647	1.47
MTW_50	0.646	2.39
WIN_10_0.5	0.645	16.78
XVAL_50	0.645	2.32
PVAL_10	0.642	1.45
WIN_25	0.637	18.12
DWM_P25	0.632	13.70
PVAL_5	0.632	1.57
WIN_50	0.623	23.66
DWM_P50	0.622	8.89
PAIRED_10_01	0.659	2
EDDM	0.616	1
NAIVE_BAYES	0.594	1
DDM	0.590	1

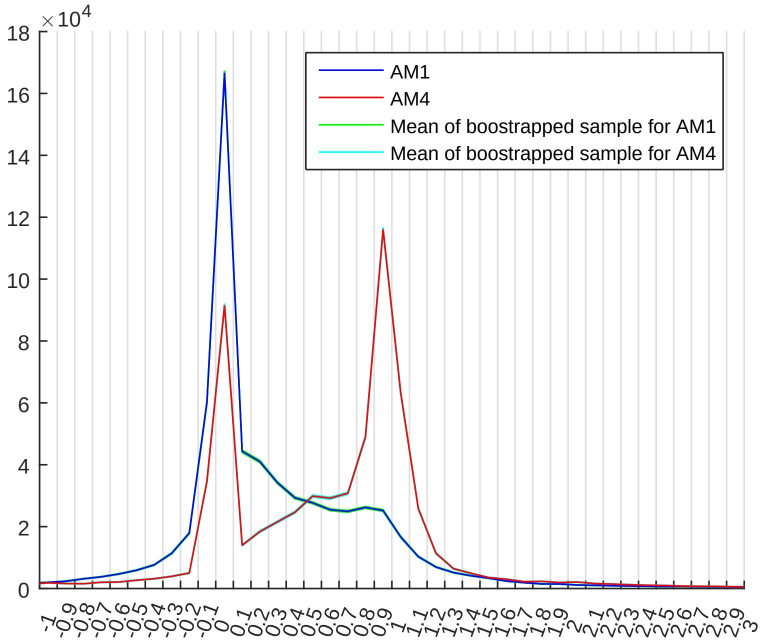
Table A.8: *Prequeuntial accuracy on PowerItaly dataset. Static weighting,* $\beta = 0.5$

and XVAL have often better accuracy rates than DWM.

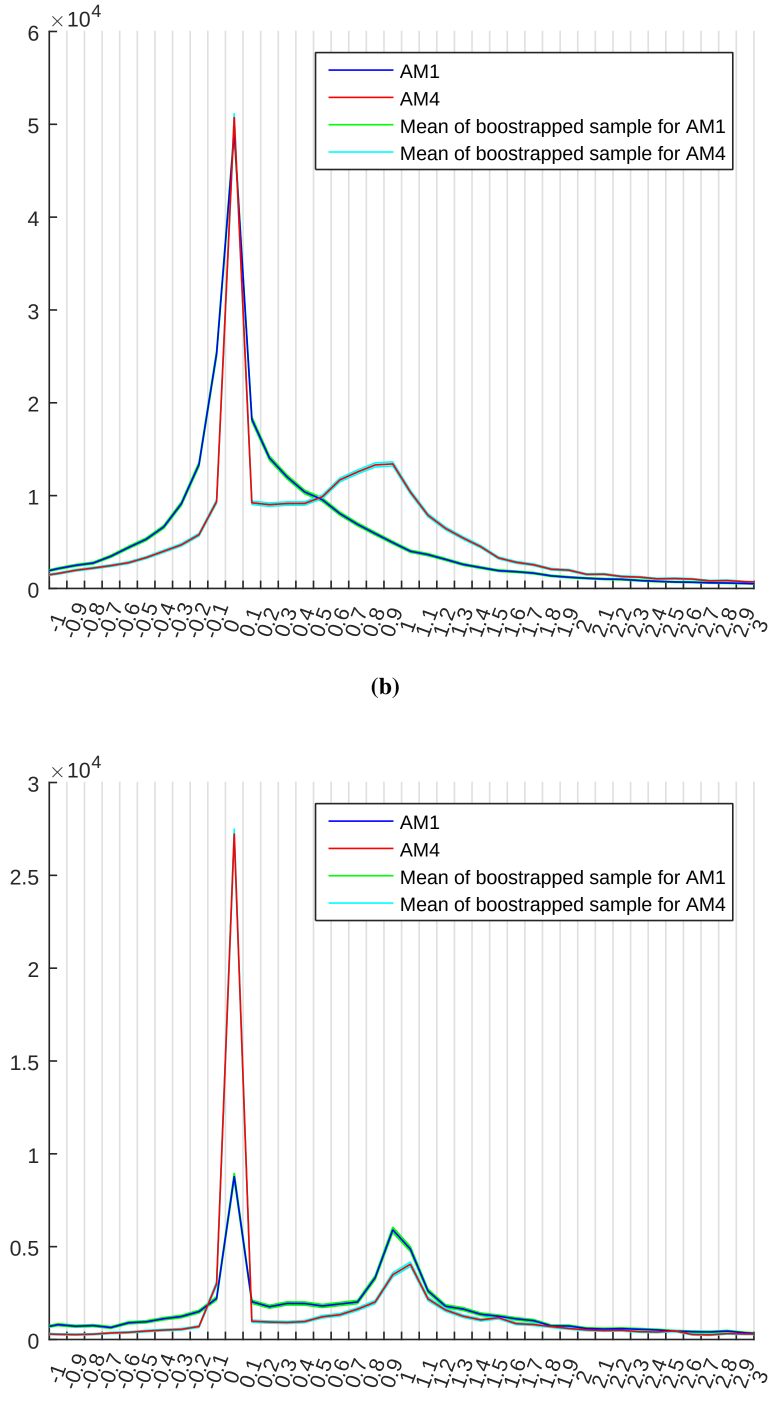
- As expected, accuracy values of the simple online learner are noticeably lower than those of original method and modifications. Also pure change detection mechanisms such as DDM and EDDM show lower accuracy rates. Paired learners algorithm shows comparable, but still somewhat lower accuracy than the leaders among tested methods.

Appendix B

Relative adaptation histograms with confidence levels



(a)



(b)

(c)

Figure B.1: *Histograms of exhaustive 4-step ahead adaptive mechanism deployment and means of histograms of 10,000 bootstrap samples with 0.95 confidence interval for AM1 and AM4 on (a) Catalyst100, (b) Oxidizer100, (c) Drier100.*

Appendix C

Cost matrices for considered datasets

True class \Cost	AM0	AM1	AM2	AM3	AM4	Average
AM0	0.000	0.015	0.018	0.015	0.020	0.014
AM1	0.016	0.000	0.007	0.019	0.013	0.011
AM2	0.018	0.003	0.000	0.021	0.013	0.011
AM3	0.046	0.045	0.047	0.000	0.013	0.030
AM4	0.043	0.036	0.036	0.026	0.000	0.028
Average	0.024	0.020	0.022	0.016	0.012	

Table C.1: *Adaptive mechanism classification cost matrix for Catalyst50.*

True class \Cost	AM0	AM1	AM2	AM3	AM4	Average
AM0	0.000	0.017	0.018	0.031	0.038	0.021
AM1	0.020	0.000	0.006	0.021	0.010	0.012
AM2	0.022	0.006	0.000	0.023	0.008	0.012
AM3	0.049	0.046	0.050	0.000	0.006	0.030
AM4	0.065	0.047	0.044	0.050	0.000	0.041
Average	0.031	0.023	0.024	0.025	0.013	

Table C.2: *Adaptive mechanism classification cost matrix for Catalyst100.*

True class \Cost	AM0	AM1	AM2	AM3	AM4	Average
AM0	0.000	0.004	0.006	0.011	0.012	0.007
AM1	0.015	0.000	0.002	0.032	0.015	0.013
AM2	0.023	0.005	0.000	0.026	0.012	0.013
AM3	0.028	0.029	0.031	0.000	0.010	0.020
AM4	0.088	0.064	0.061	0.064	0.000	0.055
Average	0.031	0.021	0.020	0.027	0.010	

Table C.3: *Adaptive mechanism classification cost matrix for Catalyst200.*

True class \Cost	**AM0**	**AM1**	**AM2**	**AM3**	**AM4**	**Average**
AM0	0.00	0.11	0.14	0.04	0.08	0.07
AM1	0.16	0.00	0.08	0.17	0.13	0.11
AM2	0.22	0.09	0.00	0.21	0.11	0.12
AM3	0.08	0.13	0.14	0.00	0.10	0.09
AM4	0.35	0.30	0.26	0.32	0.00	0.25
Average	0.16	0.13	0.12	0.15	0.08	

Table C.4: *Adaptive mechanism classification cost matrix for Oxidizer50.*

True class \Cost	**AM0**	**AM1**	**AM2**	**AM3**	**AM4**	**Average**
AM0	0.00	0.13	0.18	0.05	0.16	0.10
AM1	0.16	0.00	0.10	0.17	0.13	0.11
AM2	0.16	0.08	0.00	0.16	0.08	0.10
AM3	0.21	0.28	0.31	0.00	0.15	0.19
AM4	0.35	0.32	0.32	0.19	0.00	0.24
Average	0.18	0.16	0.18	0.11	0.10	

Table C.5: *Adaptive mechanism classification cost matrix for Oxidizer100.*

True class \Cost	**AM0**	**AM1**	**AM2**	**AM3**	**AM4**	**Average**
AM0	0.00	0.16	0.20	0.07	0.32	0.15
AM1	0.20	0.00	0.04	0.21	0.13	0.12
AM2	0.19	0.06	0.00	0.19	0.10	0.11
AM3	0.11	0.17	0.19	0.00	0.16	0.12
AM4	0.31	0.31	0.30	0.23	0.00	0.23
Average	0.16	0.14	0.14	0.14	0.14	

Table C.6: *Adaptive mechanism classification cost matrix for Oxidizer200.*

True class \Cost	**AM0**	**AM1**	**AM2**	**AM3**	**AM4**	**Average**
AM0	0.00E+00	7.67E-06	2.33E-05	2.57E-06	6.09E-05	1.89E-05
AM1	1.29E-04	0.00E+00	2.83E-05	1.34E-04	6.74E-05	7.17E-05
AM2	1.31E-04	3.46E-05	0.00E+00	1.38E-04	8.13E-05	7.69E-05
AM3	5.60E-05	6.17E-05	6.60E-05	0.00E+00	1.26E-05	3.92E-05
AM4	1.62E-04	8.56E-05	9.12E-05	1.60E-04	0.00E+00	9.97E-05
Average	9.55E-05	3.79E-05	4.17E-05	8.69E-05	4.44E-05	

Table C.7: *Adaptive mechanism classification cost matrix for Drier50.*

True class \Cost	**AM0**	**AM1**	**AM2**	**AM3**	**AM4**	**Average**
AM0	0.00E+00	6.89E-06	1.19E-05	1.72E-07	1.34E-05	6.48E-06
AM1	5.57E-05	0.00E+00	9.61E-06	5.21E-05	4.37E-05	3.22E-05
AM2	6.11E-05	5.78E-06	0.00E+00	6.00E-05	4.42E-05	3.42E-05
AM3	5.85E-06	7.67E-06	9.97E-06	0.00E+00	9.71E-06	6.64E-06
AM4	7.33E-06	1.00E-05	1.15E-05	4.87E-06	0.00E+00	6.75E-06
Average	2.60E-05	6.07E-06	8.61E-06	2.34E-05	2.22E-05	

Table C.8: *Adaptive mechanism classification cost matrix for Drier100.*

References

Abi-Haidar, A. & Rocha, L. (2008). 'Artificial Immune Systems'. In Bentley, P. J., Lee, D., & Jung, S. (eds.), *Artificial Immune Systems*, vol. 5132 of *Lecture Notes in Computer Science*, pp. 36–47. Springer Berlin Heidelberg, Berlin, Heidelberg.

Al-Jubouri, B. & Gabrys, B. (2016). 'Local Learning for Multi-layer, Multi-component Predictive System'. *Procedia Computer Science* **96**:723–732.

Alcobé, J. R. (2004). 'Incremental Hill-Climbing Search Applied to Bayesian Network Structure Learning'. In *Proceedings of the Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases*. Volume 3202 of Lecture Notes in Computer Science. Springer Verlag.

Alippi, C., Boracchi, G., Carrera, D., & Roveri, M. (2016). 'Change Detection in Multivariate Datastreams: Likelihood and Detectability Loss'. In *International Joint Conferences on Artificial Intelligence*, pp. 1–14.

Alippi, C., Boracchi, G., & Roveri, M. (2012). 'Just-in-time ensemble of classifiers'. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Altman, N. S. (1992). 'An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression'. *The American Statistician* **46**(3):175–185.

Anthony, M. & Biggs, N. (1997). *Computational learning theory*. Cambridge University Press.

Ba, J. & Frey, B. (2013). 'Adaptive dropout for training deep neural networks'. In *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, pp. 3084–3092.

Bach, S. H. & Maloof, M. a. (2008). 'Paired Learners for Concept Drift'. *2008 Eighth IEEE International Conference on Data Mining* pp. 23–32.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). 'Early drift detection method'. In *ECML PKDD 2006 Fourth International Workshop on Knowledge Discovery from Data Streams*, Berlin, Germany.

Bakirov, R. & Gabrys, B. (2013). 'Investigation of Expert Addition Criteria for Dynamically

Changing Online Ensemble Classifiers with Multiple Adaptive Mechanisms'. In Papadopoulos, H., Andreou, A., Iliadis, L., & Maglogiannis, I. (eds.), *Artificial Intelligence Applications and Innovations*, pp. 646–656.

Bakirov, R., Gabrys, B., & Fay, D. (2015). 'On sequences of different adaptive mechanisms in non-stationary regression problems'. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Bakirov, R., Gabrys, B., & Fay, D. (2016). 'Augmenting adaptation with retrospective model correction for non-stationary regression problems'. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 771–779. IEEE.

Bakirov, R., Gabrys, B., & Fay, D. (2017). 'Multiple adaptive mechanisms for data-driven soft sensors'. *Computers & Chemical Engineering* **96**:42–54.

Balaguer-Ballester, E., Tabas-Diaz, A., & Budka, M. (2014). 'Can We Identify Non-Stationary Dynamics of Trial-to-Trial Variability?'. *PLoS ONE* **9**(4).

Basak, J. (2006). 'Online adaptive decision trees: Pattern classification and function approximation'. *Neural computation* **18**(9):2062–2101.

Basseville, M. & Nikiforov, I. V. (1993). *Detection of Abrupt Changes: Theory and Appication (Prentice Hall information and system sciences series)*. Prentice Hall.

Bates, J. & Granger, C. (1969). 'The combination of forecasts'. *OR* **20**(4):451–468.

Bernadó-Mansilla, E. & Garrell-Guiu, J. M. (2003). 'Accuracy-based learning classifier systems: models, analysis and applications to classification tasks.'. *Evolutionary computation* **11**(3):209–38.

Bifet, A. & Gavaldà, R. (2007). 'Learning from time-changing data with adaptive windowing'. *SIAM International Conference on Data Mining* **7**:443–448.

Bifet, A., Holmes, G., Gavaldà, R., Pfahringer, B., & Kirkby, R. (2009). 'New Ensemble Methods For Evolving Data Streams'. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09* pp. 139–147.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). 'Learnability and the Vapnik-Chervonenkis dimension'. *Journal of the ACM* **36**(4):929–965.

Bouchachia, A. & Balaguer-Ballester, E. (2014). 'DELA: A dynamic online ensemble learning algorithm'. In *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014 - Proceedings*, pp. 491–496.

Bouchachia, A., Gabrys, B., & Sahel, Z. (2007). 'Overview of Some Incremental Learning Algorithms'. In *2007 IEEE International Fuzzy Systems Conference*, pp. 1–6. IEEE.

Breiman, L. (1996). ‘Bagging predictors’. *Machine Learning* **24**(2):123–140.

Breiman, L. (2001). ‘Random Forests’. *Machine Learning* **45**(1):5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A.

Budka, M., Eastwood, M., Gabrys, B., Kadlec, P., Martín Salvador, M., Schwan, S., Tsakonas, A., & Žliobaitė, I. (2014). ‘From Sensor Readings to Predictions: On the Process of Developing Practical Soft Sensors’. In *Advances in Intelligent Data Analysis XIII*, pp. 49–60. Springer, Cham.

Budka, M., Gabrys, B., & Ravagnan, E. (2010). ‘Robust predictive modelling of water pollution using biomarker data’. *Water Research* **44**(10):3294–3308.

Carpenter, G., Grossberg, S., & Reynolds, J. (1991). ‘ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network’. *Neural networks* **4**:565–588.

Carpenter, G. A. & Grossberg, S. (1987). ‘ART 2: self-organization of stable category recognition codes for analog input patterns’. *Applied Optics* **26**(23):4919.

Carpenter, G. a., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). ‘Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps.’. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* **3**(5):698–713.

Castillo, G. & Gama, J. (2006). ‘An Adaptive Prequential Learning Framework for Bayesian Network Classifiers’. In Fürnkranz, J., Scheffer, T., & Spiliopoulou, M. (eds.), *Knowledge Discovery in Databases: PKDD 2006*, vol. 4213 of *Lecture Notes in Computer Science*, pp. 67–78, Berlin, Heidelberg. Springer Berlin Heidelberg.

Castro, L. D. & Zuben, F. V. (1999). ‘Artificial immune systems: Part I basic theory and applications’. Tech. rep., Universidade Estadual de Campinas.

Cauwenberghs, G. & Poggio, T. (2001). ‘Incremental and decremental support vector machine learning’. In *Advances in neural information processing systems*, pp. 409–415.

Cinar, A., Parulekar, S. J., Undey, C., & Birol, G. (2003). *Batch Fermentation: Modeling: Monitoring, and Control*. CRC Press.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). ‘Credit card fraud detection and concept-drift adaptation with delayed supervised information’. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Dasgupta, D., Yu, S., & Nino, F. (2011). ‘Recent Advances in Artificial Immune Systems: Models and Applications’. *Applied Soft Computing* **11**(2):1574–1587.

Dasu, T., Krishnan, S., Venkatasubramanian, S., & Yi, K. (2006). ‘An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams’. In *Symposium on the Interface of Statistics, Computing Science, and Applications*, pp. 1–24.

Dawid, P. A. & Vovk, V. G. (1999). ‘Prequential probability: principles and properties’. *Bernoulli* **5**(1):125–162.

Dayal, B. S. & MacGregor, J. F. (1997). ‘Recursive exponentially weighted PLS and its applications to adaptive control and prediction’. *Journal of Process Control* **7**(3):169–179.

De Wolf, S., Cuypers, R., Zullo, L., Vos, B., & Bax, B. (1996). ‘Model predictive control of a slurry polymerisation reactor’. *Computers & Chemical Engineering* **20**:S955–S961.

Dietterich, T. G. (2000). ‘Ensemble Methods in Machine Learning’. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS ’00, pp. 1–15, London, UK, UK. Springer-Verlag.

Domingos, P. & Hulten, G. (2000). ‘Mining high-speed data streams’. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’00* pp. 71–80.

Doyle, F. J. (1998). ‘Nonlinear inferential control for process applications’. *Journal of Process Control* **8**(5):339–353.

Drucker, H., Burges, C., Kaufman, L., Smola, A., & Vapnik, V. (1996). ‘Support Vector Regression Machines’. *Neural Information Processing Systems* **1**:155–161.

Elwell, R. & Polikar, R. (2011). ‘Incremental learning of concept drift in nonstationary environments.’. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* **22**(10):1517–31.

Fern, A. & Givan, R. (2000). ‘Dynamic feature selection for hardware prediction’. Tech. rep., Purdue University.

Fortuna, L., Graziani, S., Rizzo, A., & Xibilia, M. G. (2007). *Soft Sensors for Monitoring and Control of Industrial Processes*. Advances in Industrial Control. Springer London, London.

Freund, Y. & Schapire, R. E. (1995). ‘A desicion-theoretic generalization of on-line learning and an application to boosting’. In Vitanyi, P. (ed.), *Computational Learning Theory*, vol. 904 of *Lecture Notes in Computer Science*, pp. 23–37. Springer Berlin / Heidelberg.

Friedman, N. & Goldszmidt, M. (1997). ‘Sequential update of Bayesian network structure’. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pp. 165–174.

Gabrys, B. (2004). ‘Learning hybrid neuro-fuzzy classifier models from data: to combine or not to combine?’. *Fuzzy Sets and Systems* **147**(1):39–56.

Gabrys, B. & Bargiela, A. (1999). ‘Neural Networks Based Decision Support in Presence of Uncertainties’. *Journal of Water Resources Planning and Management* **125**(5):272–280.

Gabrys, B. & Ruta, D. (2006). ‘Genetic algorithms in classifier fusion’. *Applied Soft Computing* **6**(4):337–347.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). ‘Learning with drift detection’. In Bazzan, A. & Labidi, S. (eds.), *Advances in Artificial Intelligence SBIA*, pp. 286–295. Springer, Berlin Heidelberg.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). ‘A survey on concept drift adaptation’. *Computing Surveys* **46**(4):1–37.

Gao, J., Fan, W., Han, J., & Yu, P. S. (2007). ‘A general framework for mining concept-drifting data streams with skewed distributions’. In *7th SIAM International Conference on Data Mining*, pp. 3–14.

Geladi, P. & Kowalski, B. R. (1986). ‘Partial least-squares regression: a tutorial’. *Analytica Chimica Acta* **185**:1–17.

Gomes Soares, S. & Araújo, R. (2015a). ‘A dynamic and on-line ensemble regression for changing environments’. *Expert Systems with Applications* **42**(6):2935–2948.

Gomes Soares, S. & Araújo, R. (2015b). ‘An on-line weighted ensemble of regressor models to handle concept drifts’. *Engineering Applications of Artificial Intelligence* **37**:392–406.

Grbić, R., Slišković, D., & Kadlec, P. (2013). ‘Adaptive soft sensor for online prediction and process monitoring based on a mixture of Gaussian process models’. *Computers & Chemical Engineering* **58**:84–97.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2016). ‘LSTM: A Search Space Odyssey’. *IEEE Transactions on Neural Networks and Learning Systems* **PP**(99):1–11.

Grisogono, A. (2006). ‘The implications of complex adaptive systems theory for C2’. Tech. rep., Land Operations Division, Defence Science and Technology Organisation.

Haag, C. R., Lamont, G. B., Williams, P. D., & Peterson, G. L. (2007). ‘An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions’. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation - GECCO ’07*, pp. 2717–2724, New York, New York, USA. ACM Press.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). 'The WEKA data mining software'. *ACM SIGKDD Explorations Newsletter* **11**(1):10.

Harries, M. (1999). 'Splice-2 comparative evaluation: Electricity pricing. Technical report. The University of South Wales'. Tech. rep., The University of South Wales.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer.

Haykin, S. S. (2009). *Neural networks and learning machines*. Prentice Hall/Pearson.

Hazan, E. & Seshadhri, C. (2009). 'Efficient learning algorithms for changing environments'. In *ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 393–400.

Herbster, M. & Warmuth, M. (1998). 'Tracking the best expert'. *Machine Learning* **29**:1–29.

Hochreiter, S. & Schmidhuber, J. (1997). 'Long Short-Term Memory'. *Neural Computation* **9**(8):1735–1780.

Holland, J. (1976). 'Adaptation'. In Rosen, R. & Snell, F. (eds.), *Progress in Theoretical Biology, 4. Plenum*.

Holland, J. (1992). 'Complex adaptive systems'. *Daedalus* **121**:17–30.

Hulten, G., Spencer, L., & Domingos, P. (2001). 'Mining time-changing data streams'. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, pp. 97–106, New York, New York, USA. ACM Press.

Ikonomovska, E., Gama, J., & Džeroski, S. (2010). 'Learning model trees from evolving data streams'. *Data Mining and Knowledge Discovery* **23**(1):128–168.

Jacobs, A., Shalizi, C. R., & Clauset, A. (2010). 'Adapting to Non-stationarity with Growing Expert Ensembles'. Tech. rep., Carnegie Mellon University.

Jacobs, R. a. (1995). 'Methods for combining experts' probability assessments.'. *Neural computation* **7**(5):867–88.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). 'Adaptive mixtures of local experts'. *Neural Comput.* **3**(1):79–87.

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall.

Jin, H., Chen, X., Yang, J., Wang, L., & Wu, L. (2015a). 'Online local learning based adaptive soft sensor and its application to an industrial fed-batch chlortetracycline fermentation process'. *Chemometrics and Intelligent Laboratory Systems* **143**:58–78.

Jin, H., Chen, X., Yang, J., & Wu, L. (2014). ‘Adaptive soft sensor modeling framework based on just-in-time learning and kernel partial least squares regression for nonlinear multiphase batch processes’. *Computers & Chemical Engineering* **71**:77–93.

Jin, H., Chen, X., Yang, J., Zhang, H., Wang, L., & Wu, L. (2015b). ‘Multi-model adaptive soft sensor modeling method using local learning and online support vector regression for nonlinear time-variant batch processes’. *Chemical Engineering Science* **131**:282–303.

Joe Qin, S. (1998). ‘Recursive PLS algorithms for adaptive data modeling’. *Computers & Chemical Engineering* **22**(4-5):503–514.

Kadlec, P. (2009). *On robust and adaptive soft sensors*. Ph.D. thesis, Bournemouth University.

Kadlec, P. & Gabrys, B. (2009a). ‘Architecture for development of adaptive on-line prediction models’. *Memetic Computing* **1**(4):241–269.

Kadlec, P. & Gabrys, B. (2009b). ‘Soft Sensor Based on Adaptive Local Learning’. In Köppen, M., Kasabov, N., & Coghill, G. (eds.), *Advances in Neuro-Information Processing: 15th International Conference, ICONIP 2008, Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part I*, vol. 5506, pp. 1172–1179. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kadlec, P. & Gabrys, B. (2010). ‘Adaptive on-line prediction soft sensing without historical data’. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Kadlec, P. & Gabrys, B. (2011). ‘Local learning-based adaptive soft sensor for catalyst activation prediction’. *AIChE Journal* **57**(5):1288–1301.

Kadlec, P., Gabrys, B., & Strandt, S. (2009). ‘Data-driven Soft Sensors in the process industry’. *Computers & Chemical Engineering* **33**(4):795–814.

Kadlec, P., Grbić, R., & Gabrys, B. (2011). ‘Review of adaptation mechanisms for data-driven soft sensors’. *Computers & Chemical Engineering* **35**(1):1–24.

Kaneko, H. & Funatsu, K. (2014). ‘Adaptive soft sensor based on online support vector regression and Bayesian ensemble learning for various states in chemical plants’. *Chemometrics and Intelligent Laboratory Systems* **137**:57–66.

Kaneko, H. & Funatsu, K. (2015). ‘Ensemble locally weighted partial least squares as a just-in-time modeling method’. *AIChE Journal* **62**:717–725.

Kaneko, H., Okada, T., & Funatsu, K. (2014). ‘Selective Use of Adaptive Soft Sensors Based on Process State’. *Industrial & Engineering Chemistry Research* **53**(41):15962–15968.

Kasabov, N. (2001). ‘Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning.'. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* **31**(6):902–18.

Keogh, E., Zhu, Q., Hu, B., Hao, Y., Xi, X., Wei, L., & Ratanamahatana, C. (2011). 'The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/~eamonn/time_series_data/'.

Kivinen, J., Smola, A. J., & Robert C. Williamson (2004). 'Online learning with kernels'. *Signal Processing, IEEE* **100**(10):1–12.

Klinkenberg, R. (2004). 'Learning drifting concepts : Example selection vs . example weighting'. *Intelligent Data Analysis* **8**(3):281–300.

Klinkenberg, R. & Joachims, T. (2000). 'Detecting concept drift with support vector machines'. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pp. 487–494.

Kolter, J. J. Z. & Maloof, M. A. (2005). 'Using additive expert ensembles to cope with concept drift'. In *ICML '05 Proceedings of the 22nd international conference on Machine learning*, ICML '05, pp. 449 – 456, New York, NY, USA. ACM.

Kolter, J. Z. & Maloof, M. A. (2007). 'Dynamic weighted majority: An ensemble method for drifting concepts'. *The Journal of Machine Learning Research* **Volume 8,**:2755–2790.

Kuh, A. & Petsche, T. (1990). 'Learning Time Varying Concepts with Applications to Pattern Recognition Problems'. In *1990 Conference Record Twenty-Fourth Asilomar Conference on Signals, Systems and Computers, 1990.*, vol. 2, pp. 971–975. IEEE.

Kullback, S. & Leibler, R. A. (1951). 'On Information and Sufficiency'. *The Annals of Mathematical Statistics* **22**(1):79–86.

Kuncheva, L. I. (2004a). 'Classifier Ensembles for Changing Environments'. In Roli, F., Kittler, J., & Windeatt, T. (eds.), *Multiple Classifier Systems: 5th International Workshop, MCS 2004, Cagliari, Italy, June 9-11, 2004. Proceedings*, pp. 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kuncheva, L. I. (2004b). *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Blackwell.

Lam, W. (1998). 'Bayesian network refinement via machine learning approach'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(3):240–251.

Lan, Y., Soh, Y. C., & Huang, G. B. (2009). 'Ensemble of online sequential extreme learning machine'. *Neurocomputing* **72**(13-15):3391–3395.

Lee, W., Stolfo, S. J., & Mok, K. W. (2000). 'Adaptive Intrusion Detection: A Data Mining Approach'. *Artificial Intelligence Review* **14**(6):533–567.

Lemke, C., Budka, M., & Gabrys, B. (2015). 'Metalearning: a survey of trends and technologies'. *Artificial Intelligence Review* **44**(1):117–130.

Lemke, C. & Gabrys, B. (2010). 'Meta-learning for time series forecasting and forecast combination'. *Neurocomputing* **73**(10):2006–2016.

Lemke, C., Riedel, S., & Gabrys, B. (2009). 'Dynamic combination of forecasts generated by diversification procedures applied to forecasting of airline cancellations'. In *2009 IEEE Symposium on Computational Intelligence for Financial Engineering*, pp. 85–91. IEEE.

Lemke, C., Riedel, S., & Gabrys, B. (2013). 'Evolving forecast combination structures for airline revenue management'. *Journal of Revenue and Pricing Management* **12**(3):221–234.

Li, W., Yue, H., Valle-Cervantes, S., & Qin, S. (2000). 'Recursive PCA for adaptive process monitoring'. *Journal of Process Control* **10**(5):471–486.

Lin, L.-J. (1992). 'Self-improving reactive agents based on reinforcement learning, planning and teaching'. *Machine Learning* **8**(3-4):293–321.

Littlestone, N., Long, P. M., & Warmuth, M. K. (1991). 'On-Line Learning of Linear Functions'. *Computational Complexity* **5**:465 – 475.

Littlestone, N. & Warmuth, M. (1994). 'The Weighted Majority Algorithm'. *Information and Computation* **108**(2):212–261.

Mann, H. B. & Whitney, D. R. (1947). 'On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other'. *The Annals of Mathematical Statistics* **18**(1):50–60.

Martín Salvador, M., Budka, M., & Gabrys, B. (2016a). 'Adapting Multicomponent Predictive Systems using Hybrid Adaptation Strategies with Auto-WEKA in Process Industry'. In *AutoML at ICML 2016*, no. 2011, pp. 1–8.

Martín Salvador, M., Budka, M., & Gabrys, B. (2016b). 'Automatic composition and optimisation of multicomponent predictive systems'. *arXiv preprint* .

Martín Salvador, M., Budka, M., & Gabrys, B. (2016c). 'Effects of Change Propagation Resulting from Adaptive Preprocessing in Multicomponent Predictive Systems'. *Procedia Computer Science* **96**:713–722.

Martín Salvador, M., Gabrys, B., & Žliobaitė, I. (2014). 'Online Detection of Shutdown Periods in Chemical Plants: A Case Study'. *Procedia Computer Science* **35**:580–588.

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C. C., Gao, J., Han, J., Srivastava, A., & Oza, N. C. (2013). 'Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams'. *IEEE Transactions on Knowledge and Data Engineering* **25**(7):1484–1497.

Mills, T. C. (1991). *Time Series Techniques for Economists*. Cambridge University Press.

Minku, L. L. & Yao, X. (2012). 'DDD: A New Ensemble Approach for Dealing with Concept Drift'. *IEEE Transactions on Knowledge and Data Engineering* **24**(4):619–633.

Mizrach, B. (1996). 'Forecast comparison in L2'. Tech. Rep. 908, Working Papers, No. 1995-24, Department of Economics, Rutgers, The State University of New Jersey.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). 'Human-level control through deep reinforcement learning'. *Nature* **518**(7540):529–533.

Moore, H. F., Bain, S. S., & Chaffin, M. A. (2011). 'Method and apparatus for controlling hydroprocessing on-line'.

Narasimhamurthy, A. & Kuncheva, L. (2007). 'A framework for generating data to simulate changing environments'. In *Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications*, pp. 384–389, Anaheim, CA, USA. ACTA Press.

Nason, G. (2013). 'A test for second-order stationarity and approximate confidence intervals for localized autocovariances for locally stationary time series'. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **75**(5):879–904.

Nath, S. V. (2007). 'Champion-challenger based predictive model selection'. In *Proceedings 2007 IEEE SoutheastCon*, pp. 254–254. IEEE.

Ni, W., Brown, S. D., & Man, R. (2014). 'A localized adaptive soft sensor for dynamic system modeling'. *Chemical Engineering Science* **111**:350–363.

Parzen, E. (1962). 'On Estimation of a Probability Density Function and Mode'. *The Annals of Mathematical Statistics* **33**(3):pp. 1065–1076.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Plackett, R. L. (1950). 'Some Theorems in Least Squares'. *Biometrika* **37**(1/2):149.

Polikar, R., Udpa, L., Udpa, S. S., & Honavar, V. (2001). 'Learn++: an incremental learning algorithm for supervised neural networks'. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* **31**(4):497–508.

Potts, D. & Sammut, C. (2005). 'Incremental Learning of Linear Model Trees'. *Machine Learning* **61**(1-3):5–48.

Prasad, V., Schley, M., Russo, L. P., & Wayne Bequette, B. (2002). 'Product property and production rate control of styrene polymerization'. *Journal of Process Control* **12**(3):353–372.

Priestley, M. & Subba Rao, T. (1969). 'A Test for Non-Stationarity of Time-Series'. *Journal of the Royal Statistical Society. Series B (Methodological)* **31, No. 1**:140–149.

Quinlan, J. R. (1992). 'LEARNING WITH CONTINUOUS CLASSES'. *Proceedings of AI'92 (Adams & Sterling, Eds) Singapore: World Scientific.* pp. 343–348.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann.

Raza, H., Prasad, G., & Li, Y. (2015). 'EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments'. *Pattern Recognition* **48**(3):659–669.

Riedel, S. & Gabrys, B. (2007). 'Dynamic Pooling for the Combination of Forecasts generated using Multi Level Learning'. In *2007 International Joint Conference on Neural Networks*, pp. 454–459. IEEE.

Riedmiller, M. & Braun, H. (1993). 'A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm'. In *IEEE International Conference on Neural Networks*.

Ruta, D. & Gabrys, B. (2000). 'An Overview of Classifier Fusion Methods'. *Computing and Information Systems* **7**(1):1–10.

Ruta, D. & Gabrys, B. (2002). 'A Theoretical Analysis of the Limits of Majority Voting Errors for Multiple Classifier Systems'. *Pattern Analysis and Applications* **5**(4):333–350.

Ruta, D. & Gabrys, B. (2005). 'Classifier selection for majority voting'. *Information Fusion* **6**(1):63–81.

Ruta, D. & Gabrys, B. (2007). 'Neural Network Ensembles for Time Series Prediction'. In *2007 International Joint Conference on Neural Networks*, pp. 1204–1209. IEEE.

Ruta, D., Gabrys, B., & Lemke, C. (2011). 'A Generic Multilevel Architecture for Time Series Prediction'. *IEEE Transactions on Knowledge and Data Engineering* **23**(3):350–359.

Sahel, Z., Bouchachia, A., Gabrys, B., & Rogers, P. (2007). 'Adaptive Mechanisms for Classification Problems with Drifting Data'. In *Proc. of the 11th International Conference on Knowledge-based Intelligent Engineering Systems (KES'2007)*, pp. 419–426. Springer, Berlin, Heidelberg.

Salganicoff, M. (1993a). 'Density-Adaptive Learning and Forgetting. Technical Report No. IRCS-93-50'. Tech. rep., University of Pennsylvania, Inistitute for Research in Cognitive Science.

Salganicoff, M. (1993b). 'Explicit Forgetting Algorithms for Memory Based Learning. Technical Report No. IRCS-93-49'. Tech. rep., University of Pennsylvania, Inistitute for Research in Cognitive Science.

Schlimmer, J. C. & Granger, R. H. (1986a). 'Beyond incremental processing: Tracking Concept Drift'. *AAAI-86 Proceedings* pp. 502–507.

Schlimmer, J. C. & Granger, R. H. (1986b). 'Incremental Learning from Noisy Data'. *Machine Learning* **1**(3):317–354.

Scholz, M. & Klinkenberg, R. (2007). 'Boosting Classifiers for Drifting Concepts'. *Intelligent Data Analysis* **11**(1):1–40.

Shao, W. & Tian, X. (2015). 'Adaptive soft sensor for quality prediction of chemical processes based on selective ensemble of local partial least squares models'. *Chemical Engineering Research and Design* **95**:113–132.

Shao, W., Tian, X., & Wang, P. (2014). 'Local Partial Least Squares Based Online Soft Sensing Method for Multi-output Processes with Adaptive Process States Division'. *Chinese Journal of Chemical Engineering* **22**(7):828–836.

Shao, W., Tian, X., & Wang, P. (2015a). 'Soft sensor development for nonlinear and time-varying processes based on supervised ensemble learning with improved process state partition'. *Asia-Pacific Journal of Chemical Engineering* **10**(2):282–296.

Shao, W., Tian, X., Wang, P., Deng, X., & Chen, S. (2015b). 'Online soft sensor design using local partial least squares models with adaptive process state partition'. *Chemometrics and Intelligent Laboratory Systems* **144**:108–121.

Stahl, F., Gabrys, B., Gaber, M. M., & Berendsen, M. (2013). 'An overview of interactive visual data mining techniques for knowledge discovery'. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **3**(4):239–256.

Stanley, K. O. (2002). 'Evolving neural networks through augmenting topologies'. *Evolutionary computation* **10**(2):99–127.

Stanley, K. O. (2003). 'Learning concept drift with a committee of decision trees'. Tech. rep., UT-AI-TR-03-302, Department of Computer Science, University of Texas in Austin.

Strackeljan, J. (2006). 'NiSIS Competition 2006- Soft Sensor for the adaptive Catalyst Monitoring of a MultiTube Reactor'. Tech. rep., Universität Magdeburg.

Street, W. N. & Kim, Y. S. (2001). 'A streaming ensemble algorithm (SEA) for large-scale classification'. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* pp. 377–382.

Student (1908). 'The Probable Error of a Mean'. *Biometrika* **6**(1):1–25.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., & Mahoney, P. (2006). 'Stanley: The robot that won the DARPA Grand Challenge'. *Journal of Field Robotics* **23**(9):661–692.

Tsakonas, A. & Gabrys, B. (2012). 'GRADIENT: Grammar-driven genetic programming framework for building multi-component, hierarchical predictive systems'. *Expert Systems with Applications* **39**(18):13253–13266.

Tsakonas, A. & Gabrys, B. (2013). 'A Fuzzy Evolutionary Framework for Combining Ensembles'. *Applied Soft Computing* **13**(4):1800–1812.

Tsymbal, A., Pechenizkiy, M., Cunningham, P., & Puuronen, S. (2008). 'Dynamic integration of classifiers for handling concept drift'. *Information Fusion* **9**(1):56–68.

Vakil-Baghmisheh, M.-T. & Pavešić, N. (2003). 'A Fast Simplified Fuzzy ARTMAP Network'. *Neural Processing Letters* **17**(3):273–316.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

Vapnik, V. N. & Chervonenkis, A. Y. (1971). 'On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities'. *Theory of Probability & Its Applications* **16**(2):264–280.

Vidyasagar, M. (1997). *A theory of learning and generalization : with applications to neural networks and control systems*. Springer.

von Sachs, R. & Neumann, M. H. (2000). 'A Wavelet-Based Test for Stationarity'. *Journal of Time Series Analysis* **21**(5):597–613.

Vovk, V. G. (1990). 'Aggregating strategies'. In *COLT '90 Proceedings of the third annual workshop on Computational learning theory*, pp. 371–386, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). 'Mining concept-drifting data streams using ensemble classifiers'. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, pp. 226–235, New York, New York, USA. ACM Press.

Wang, H., Yin, J., Pei, J., Yu, P. S., & Yu, J. X. (2006). 'Suppressing model overfitting in mining concept-drifting data streams'. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, pp. 736–741, New York, New York, USA. ACM Press.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Phd thesis, Harvard University.

Widmer, G. (1993). 'Effective learning in dynamic environments by explicit context tracking'. *Machine Learning: ECML-93* pp. 1–17.

Widmer, G. & Kubat, M. (1996). 'Learning in the presence of concept drift and hidden contexts'. *Machine Learning* **23**(1):69–101.

Wold, H. (1966). 'Estimation of Principal Components and Related Models by Iterative Least squares'. In P. R. Krishnaiah (ed.), *Multivariate Analysis*, pp. 391–420. Academic Press, New York.

Yang, J., Yan, R., & Hauptmann, A. G. (2007). 'Cross-domain video concept detection using adaptive SVMs'. In *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, p. 188, New York, New York, USA. ACM Press.

Zhu, X. (2010). 'Stream Data Mining Repository, http://www.cse.fau.edu/˜xqzhu/stream.html'.

Žliobaitė, I. (2009). 'Learning under concept drift: an overview'. Tech. rep., Vilnius University.

Žliobaitė, I. (2011). 'Combining Similarity in Time and Space for Training Set Formation under Concept Drift'. *Intelligent Data Analysis* **15**(4):589–611.

Žliobaitė, I., Bifet, A., Gaber, M., Gabrys, B., Gama, J., Minku, L., & Musial, K. (2012). 'Next challenges for adaptive learning systems'. *ACM SIGKDD Explorations Newsletter* **14**(1):48.

Žliobaitė, I. & Gabrys, B. (2014). 'Adaptive Preprocessing for Streaming Data'. *IEEE Transactions on Knowledge and Data Engineering* **26**(2):309–321.

Žliobaitė, I. & Kuncheva, L. I. (2010). 'Theoretical Window Size for Classification in the Presence of Sudden Concept Drift'. Tech. rep., CS-TR-001-2010, Bangor University, UK.