Generic adaptation strategies for automated machine learning

Rashid Bakirov[1], Bogdan Gabrys[2], and Damien Fay[3]

[1] Department of Computing and Informatics, Bournemouth University, Poole, UK.
rbakirov@bournemouth.ac.uk
[2] Advanced Analytics Institute, University of Technology Sydney, Australia. (see http://www.bogdan-gabrys.com)
[3] INFOR/Logicblox, Atlanta, Georgia, US.

Abstract. Automation of machine learning model development is increasingly becoming an established research area. While automated model selection and automated data pre-processing have been studied in depth, there is, however, a gap concerning automated model adaptation strategies when multiple strategies are available. Manually developing an adaptation strategy, including estimation of relevant parameters can be time consuming and costly. In this paper we address this issue by proposing generic adaptation strategies based on approaches from earlier works. Experimental results after using the proposed strategies on the modified version of Dynamic Weighted Majority algorithm on 31 datasets confirm their viability. These strategies often achieve better or comparable performance with custom adaptation strategies and naive methods such as repeatedly using only one adaptive mechanism.

Keywords: Adaptive machine learning · Streaming data · Non-stationary data · Concept drift · Automated machine learning

1 Introduction

Automated model selection has long been studied [47] with some notable recent advances [29, 37, 35]. In addition, automatic data pre-processing has also been a topic of recent interest [18, 38]. There is however a gap concerning automated development of models' adaptation strategy, which is addressed in this paper. Here we define *adaptation* as changes in model training set, parameters and structure all designed to track changes in the underlying data generating process over time. This contrasts with model selection which focuses on parameter estimation and the appropriate family to sample the model from. There is a dearth of research on adaptation strategies (Section 2) and the focus of this paper is on what strategy to apply at a given time given the history of adaptations.

With the current advances in data storage, database and data transmission technologies, mining streaming data has become a critical part of many processes. Many models which are used to make predictions on streaming data are static, in the sense that they do not learn on current data and hence remain unchanged. However, there exists a class of models, online learning models, which

are capable of adding observations from the stream to their training sets. In spite of the fact that these models utilise the data as it arrives, there can still arise situations where the underlying assumptions of the model no longer hold. We call such settings dynamic environments, where changes in data distribution [49], change in features' relevance [17], non-symmetrical noise levels [43] are common. These phenomena are sometimes called *concept drift*. It has been shown that many changes in the environment which are no longer being reflected in the model contribute to the deterioration of model's accuracy over time [42, 46]. This requires constant manual retraining and readjustment of the models which is often expensive, time consuming and in some cases impossible - for example when the historical data is not available any more. Various approaches have been proposed to tackle this issue by making the model adapt itself to the possible changes in environment while avoiding its complete retraining.

In this paper we aim to bridge the identified gap between automation and adaptation of machine learning algorithms. Typically there are several possible ways or *adaptive mechanisms* (AMs) to adapt a given model. In this scenario, the adaptation is achieved by deploying one of multiple AMs, which changes the state of the existing model. Note that as explained in Section 3.1, this formulation also includes algorithms with a single adaptive mechanism, if there is a possibility of *not deploying* it. This would apply to the most, if not all, adaptive machine learning methods. A sequential adaptation framework proposed in earlier works [7] separates adaptation from prediction, thus enabling flexible deployment orders of AMs. We call these orders *adaptation strategies*. Generic adaptation strategies, developed according to this framework can be applied to any set of adaptive mechanisms for various machine learning algorithm. This removes the need to design custom adaptive strategies which results in automation of adaptation process. In this work we empirically show the viability of the generic adaptive strategies based upon techniques shown in [6], specifically a cross-validatory adaptation strategy with the optional use of retrospective model correction.

We focus on the batch prediction scenario, where data arrives in large segments called batches. This is a common industrial scenario, especially in the chemical, microelectronics and pharmaceutical areas [11]. For the experiments we use the modified version of a well-known adaptive ensemble algorithm, Dynamic Weighted Majority (DWM) [33].

After a large-scale experimentation with 31 classification data sets, the main finding of this work is that in our settings, the proposed generic adaptive strategies usually show better or comparable accuracy rates with the repeated deployment of a single AM and the custom adaptive strategies. Thus, they are feasible to use for adaptation purposes, while saving time and effort spent on designing custom strategies.

Note that the full version of the paper, using more algorithms for experiments and deeper explanations, is available under https://arxiv.org/abs/1812.10793.

2 Related Work

Changes in data distribution, also called *dataset shift* [40], are common in the real world streaming data scenario. These can occur as a result of external factors in data generating process, such as seasonalities, changes of the model deployment conditions, sensor wear and tear etc. The changes often result in the decrease of the model's accuracy, a condition known as concept drift [21]. Adapting machine learning models is an essential strategy for automatically dealing with changes in an underlying data distribution to avoid training a new model manually. Modern machine learning methods typically contain a complex set of elements allowing many possible adaptation mechanisms. This can increase the flexibility of such methods and broaden their applicability to various settings. However, the existence of multiple AMs also increases the decision space with regards to the adaptation choices and parameters, ultimately increasing the complexity of adaptation strategy. Adaptation of the models, including the formulation in the next section can be also expressed as retraining, revision or reframing, as introduced in [28]. Dynamic ensemble algorithms are a good example of methods with multiple adaptive mechanisms. These may include update or retraining of existing experts, their reweighing and addition/removal.

One can distinguish global (based on global measures such as accuracy or error rate) and local (based on the input/output space of the data instances) weighting methods for experts, whereas global weighting is used more commonly [36]. Examples of methods that employ global weighting include [45, 34, 33] for classification and [22] for regression. Local ensemble learning has been applied to regression in [32, 23, 44, 30, 1].

Adding an expert is a more dramatic adaptation measure. In the simplest case an expert may be added after each misclassification [34, 33]. [22] 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 works consider error measures such as in [20] or based on input data as in [8]. [2] 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. Adding an expert at fixed intervals is done in [16] in the batch learning and in [45, 26] in incremental learning contests.

Removal of experts is often performed when their performance is unsatisfactory [33], when change is detected [8], when a new expert performs better and replaces the existing one [46], or based on their age [26].

3 Formulation

As adaptation mechanisms can affect several elements of a model and can depend on performance several time steps back, it is necessary to make concrete the meaning via a framework to avoid confusion. We assume that the data is generated by an unknown time varying data generating process which can be formulated as:

$$y_\tau = \psi(\boldsymbol{x}_\tau, \tau) + \epsilon_\tau, \tag{1}$$

where ψ is the unknown function, ϵ_τ a noise term, $\boldsymbol{x}_\tau \in \mathcal{R}^M$ is an input data instance, and y_τ is the observed output at time τ. Then we consider the predictive method at a time τ as a function:

$$\hat{y}_\tau = f_\tau(\boldsymbol{x}_\tau, \Theta_f), \tag{2}$$

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. Our estimate, f_τ, evolves via adaptation as each batch of data arrives as is now explained.

3.1 Adaptation

In the batch streaming scenario considered in this paper, 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 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 $\boldsymbol{y}_k = y_{\tau_k}, \cdots, y_{\tau_{k+1}-1}$. We examine the case where the prediction function f_k is static within a k-th batch.[4]

We denote the *a priori* predictive function at batch k 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 the batch $\boldsymbol{\mathcal{V}}_k = \{\boldsymbol{X}_k, \boldsymbol{y}_k\}$ and other optional inputs. This can be written as:

$$g_k(\boldsymbol{X}_k, \boldsymbol{y}_k, \Theta_g, f_k^-, \hat{\boldsymbol{y}}_k) : f_k^- \rightarrow f_k^+. \tag{3}$$

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

We examine a situation when a choice of multiple, different AMs, $\{\emptyset, g_1, ..., g_H\} = G$, is available. Any AM $g_{h_k} \subset G$ can be deployed on each batch, where h_k denotes the AM deployed at batch k. As the history of all adaptations up to the current batch, k, have in essence created f_k^-, we call that sequence $g_{h_1}, ..., g_{h_k}$ an *adaptation sequence*. Note that we also include the option of applying no adaptation denoted by $\emptyset$, thus any adaptive algorithm fits our framework, as long as there is an option of *not adapting*. In this formulation, only one element of G is applied for each batch of data. Deploying multiple adaptation mechanisms on the same batch are accounted for with their own symbol in G.

[4] A batch typically represents a real-world segmentation of the data which is meaningful, for example a plant run and so our adaptation attempts to track run to run changes in the process. We also found in our experiments that adapting within a batch can be detrimental as it leads to drift in the models.

3.2 Generic adaptation strategies

In this section we present different strategies we examined to understand better the issues surrounding flexible deployment of AMs and assist in the choice of adaptation sequence.

At every batch k, an AM g_{h_k} must be chosen to deploy on the current batch of data. To obtain a benchmark performance, an adaptation strategy which minimizes the error over the incoming data batch $\boldsymbol{X}_{k+1}, \boldsymbol{y}_{k+1}$:

$$f_{k+1}^{-} = f_k^{-} \circ g_{h_k}, \quad h_k = \underset{h_k \in 1 \cdots H}{\operatorname{argmin}} \langle (f_k^{-} \circ g_{h_k})(\boldsymbol{X}_{k+1}), \boldsymbol{y}_{k+1} \rangle \tag{4}$$

where $\langle\ \rangle$ denotes the chosen error measure, can be used. Since $\boldsymbol{X}_{k+1}, \boldsymbol{y}_{k+1}$ are not yet obtained, this strategy is not applicable in the real life situations. Also note that this may not be the overall optimal strategy which minimizes the error over the whole dataset. While discussing the results in the Section 5 we refer to this strategy as *Oracle*.

Given the inability to conduct the *Oracle* strategy, below we list the alternatives. The simplest adaptation strategy is applying the same AM to every batch (these are denoted *Sequence1*, *Sequence2* etc. in Section 5). A more common practice is applying multiple or all available adaptive mechanisms, denoted as *Joint* in Section 5.

As introduced in [5], it is also possible to use $\mathcal{V}_k$ for the choice of g_{h_k}. Given observations, the *a posteriori* prediction error $\mathcal{V}_k$ is $\langle (f_k^{-} \circ g_{h_k})(\boldsymbol{X}_k), \boldsymbol{y}_k \rangle$. However, this is effectively an in-sample error as g_{h_k} is a function of $\{\boldsymbol{X}_k, \boldsymbol{y}_k\}$.[5] To obtain a generalised estimate of the prediction error we apply q-fold[6] cross validation. 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. $f_k^{+} = f_k^{-} \circ g_{h_k}(\{\boldsymbol{X}_k, \boldsymbol{y}_k\}_{\in \mathcal{S}})$ 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 q 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:

$$f_{k+1}^{-} = f_k^{-} \circ g_{h_k}, \quad h_k = \underset{h_k \in 1 \cdots H}{\operatorname{argmin}} \langle (f_k^{-} \circ g_{h_k})(\boldsymbol{X}_k), \boldsymbol{y}_k \rangle^{\times} \tag{5}$$

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

The next strategy can be used in combination with any of the above strategies as it focuses on the history of the adaptation sequence and retrospectively adapts two steps back. This is called the *retrospective model correction* (RC) [6]. Specifically, we estimate which adaptation at batch $k-1$ would have produced the best estimate in block k:

$$f_{k+1}^{-} = f_{k-1}^{-} \circ g_{h_{k-1}} \circ g_{h_k}, \quad h_{k-1} = \underset{h_{k-1} \in 1 \cdots H}{\operatorname{argmin}} \langle (f_{k-1}^{-} \circ g_{h_{k-1}})(\boldsymbol{X}_k), \boldsymbol{y}_k \rangle \tag{6}$$

[5] 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 we risk overfitting the model if we also evaluate the goodness of fit on $\boldsymbol{y}_k$.

[6] In subsequent experiments, $q = 10$

Using the cross-validated error measure in Equation 6 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}). This strategy can be extended to consider the sequence of r AMs while choosing the optimal state for the current batch, which we call r-step retrospective correction:

$$f^-_{k+1} = f^-_{k-r} \circ g_{h_{k-r}} \circ \cdots \circ g_{h_{k-1}} \circ g_{h_k}, \{h_{k-r} \cdots h_{k-1}\} = \\ = \underset{h_{k-r} \cdots h_{k-1} \in 1 \cdots H}{\operatorname{argmin}} \langle (f^-_{k-r} \circ g_{h_{k-r}} \circ \cdots \circ g_{h_{k-1}})(\boldsymbol{X}_k), \boldsymbol{y}_k \rangle \quad (7)$$

Notice that when using this approach, the prediction function $f_k(x)$, which is used to generate predictions, can be different from the *propagation* function $f'_k(x)$ which is used as input for adaptation.

4 Batch Dynamic Weighted Majority

For our experiments we have modified the Dynamic Weighted Majority (DWM) [33]. Batch Dynamic Weighted Majority (bDWM) is an extension of DWM designed to operate on batches of data instead of on single instances as in the original algorithm. bDWM is a global experts ensemble. Assume 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 $\boldsymbol{x}$ and 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:

$$a_{i,j} = \begin{cases} 1 & \text{if } s_i(\boldsymbol{x}) = c_j \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

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 $z_j = \sum_{i=1}^{I} w_i a_{i,j}$. The final prediction is[7]:

$$\hat{y} = \underset{c_j}{\operatorname{argmax}}(z_j). \quad (9)$$

bDWM starts with a single expert and can be adapted using an arbitrary sequence of 8 possible AMs (including no adaptation) given below.

DAM0 (No adaptation). No changes are applied to the predictive model, corresponding to $\emptyset$. This AM will be denoted as DAM0.

DAM1 (Batch Learning). After the arrival of the batch $\boldsymbol{\mathcal{V}}_t$ at time t each expert is updated with it. This AM will be denoted as DAM1.

DAM2 (Weights Update and Experts Pruning). Weights of experts are updated using following rule:

$$w_i^{t+1} = w_i^t * e^{u_i^t}. \quad (10)$$

[7] This definition is adapted from [36].

where w_i^t is the weight of the i-th expert at time t, and u_i^t is its accuracy on the batch $\boldsymbol{\mathcal{V}}_t$. The weights of all experts in ensemble are then normalized and the experts with a weight less than a defined threshold η are removed. It should be noted that the choice of factor $e^{u_i^t}$ is inspired by [27], although due to different algorithm settings, the theory developed there is not readily applicable to our scenario. Weights update is different to the original DWM, which uses an arbitrary factor $\beta < 1$ to decrease the weights of misclassifying experts. This AM will be denoted as DAM2.

DAM3 (Creation of a New Expert). New expert is created from the batch $\boldsymbol{\mathcal{V}}_t$ and is given a weight of 1. This AM will be denoted as DAM3.

The following joint AMs are combinations of the DAM1-3:

DAM4. DAM2 (Weights Update and Experts Pruning) followed by DAM1 (Batch Learning).

DAM5. DAM1 (Batch Learning) followed by DAM3 (Creation of a New Expert).

DAM6. DAM2 (Weights Update and Experts Pruning) followed by DAM3 (Creation of a New Expert).

DAM7. DAM2 (Weights Update and Experts Pruning) followed by DAM1 (Batch Learning) followed by DAM3 (Creation of a New Expert).

bDWM Custom Adaptive Strategy. Having described the separate adaptive mechanisms, we now give the custom adaptive strategy for bDWM, adapted for batch setting from the original DWM. It starts with a single expert with a weight of one. At time t, after an arrival of new batch $\boldsymbol{\mathcal{V}}_t$, experts makes predictions and overall prediction is calculated as shown earlier in this section. After the arrival of true labels all experts learn on the batch $\boldsymbol{\mathcal{V}}_t$ (invoking DAM1), update their weights (DAM2) and ensemble's accuracy u_t is calculated. If u_t accuracy is less than the accuracy of the naive majority classifier (based on all the batches of data seen up to this point) on the last batch, a new expert with the weight of one is created (DAM3).

5 Experimental results

The goal of experiments given in this section is the empirical comparison of generic adaptation strategies proposed in 3.2 with strategies involving repeated deployment of one or all available AMs and custom adaptive strategies. The experimentation was performed using 5 real world (Table 2) and 26 synthetic (Table 1) datasets. Synthetic data is visualised in Figure 1 [8].

5.1 Methodology

We use the adaptive strategies presented in Table 3 for our experimental comparison. These strategies consist of fixed deployment sequences of AMs given

[8] Code for bDWM as well as all the datasets https://github.com/RashidBakirov/multiple-adaptive-mechanisms.

Table 1: Synthetic classification datasets used in experiments, from [4]. Column "Drift" specifies number of drifts/changes in data, the percentage of change in the decision boundary and its type. ***N*** stands for number of instances and ***C*** for number of classes. All datasets have 2 input features.

#	Data type	***N***	***C***	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

in Section 4, the custom bDWM adaptation strategy and the proposed generic strategies. The tests were run on classification datasets shown in Tables 1, as well as 5 real world datasets from various areas (Table 2). The datasets have different rate of change and noise. Three different batch sizes - 10, 20 and 50 for each dataset were examined. Prtools [15] implementation of Naive Bayes (NB) and Weka [24] implementation of Hoeffding Trees[9] (HT) [13] were used as base learners.

[9] Using incremental learning across the batch.

Table 2: Real world classification datasets.***N*** stands for number of instances, ***M*** for number of features and ***C*** for number of classes.

#	Name	*N*	*M*	*C*	Brief description
27	Australian electricity prices (Elec2)	27887	6	2	Widely used concept drift benchmark dataset thought to have seasonal and other changes as well as noise. Task is the prediction of whether electricity price rises or falls while inputs are days of the week, times of the day and electricity demands [25].
28	Power Italy	4489	2	4	The task is prediction of hour of the day (03:00, 10:00, 17:00 and 21:00) based on supplied and transferred power measured in Italy. [48, 10].
29	Contraceptive	4419	9	3	Contraceptive dataset from UCI repository [41] with artificially added drift [39].
30	Iris	450	4	4	Iris dataset [3, 19] with artificially added drift [39].
31	Yeast	5928	8	10	Contraceptive dataset from UCI repository [41] with artificially added drift [39].

5.2 Results

We compare the accuracy values[10] of the proposed strategies *XVSelect* and *XVSelectRC* to the bDWM custom strategy with and without retrospective correction (*bDWM* and *BDWM+RC*) and the most accurate single AM deployment strategy with and without retrospective correction across the datasets (*BestAM* and *BestAM+RC*) on all of the 31 classification datasets. The statistical significance is assessed using the Friedman test with post-hoc Nemenyi test, which are widely used to compare multiple classifiers [12]. The Friedman test checks for statistical difference between the compared classifiers; if so, the Nemenyi test is used to identify which classifiers are significantly better than others. The results of the Nemenyi test are shown in Figure 2 as Nemenyi plots[11]. They plot the average rank of all methods and the critical difference per batch/base learner. Classifiers that are statistically equivalent are connected by a line.

For all of the cases except batch size of 10 with NB as base learner, *XVSelect* or *XVSelectRC* are ranked higher than *bDWM*, in some cases significantly so. It is interesting to note the differences in relative accuracy values between real and synthetic datasets. For example, *for all of the real datasets with all batch sizes with base learner HT, XVSelect and XVSelectRC perform better than any other strategies* (there is a similar but less pronounced effect for NB base learner). This may be related to the more complicated nature of these datasets, which leads to the fact that repeatedly deploying a single AM or using a fixed custom adaptive strategy results in lower accuracy rates. In fact, this leads to the obesrvation that

[10] For synthetic datasets in this work we generate an additional 100 test data instances for each single instance in training data using the same distribution. The predictive accuracy on the batch is then measured on test data relevant to that batch. This test data is not used for training or adapting models.

[11] Freely available code from [14] and [9] were used to make these plots

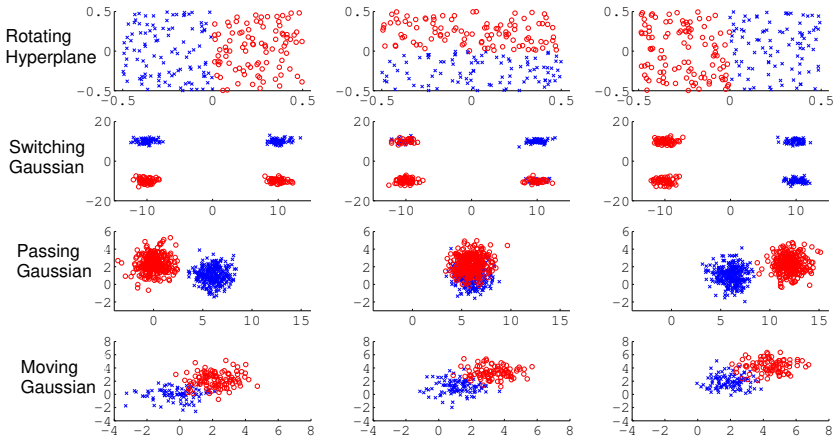


Fig. 1: Synthetic datasets visualisation [4].

Nemenyi plots in Figure 2 may even understate the performance of *XVSelect* and *XVSelectRC*, because in them, we group the results from much higher number (26) of synthetic than real datasets (5).

The benefit of *retrospective correction* seems to be dependent on the base learner. For Hoeffding Trees, this technique proved to be more useful than for Naive Bayes. This can be seen from the figure 2, where for HT, RC often improves the accuracy of the adaptive strategy, as opposed to NB. This may relate to the fact that adapting Hoefffding Tree can be influenced by its initial state much stronger than adapting Naive Bayes model. In terms of batch sizes, increasing n improves the performance of *XVSelect* and *XVSelectRC*. This effect is to be expected, as with the larger sizes batches tend to become more representative of the true distribution, and hence using them to choose adaptive mechanism is more warranted.

6 Discussion and Conclusions

The core aim of this paper was to explore the issue of automating the adaptation of predictive algorithms, which was found to be a rather overlooked direction in otherwise popular area of automated machine learning. In our research, we have addressed this by utilising a simple, yet powerful adaptation framework, which separates adaptation from prediction, defines adaptive mechanisms and adaptive strategies, as well as allows the use of retrospective model correction. This adaptation framework enables the development of generic adaptation strategies, which can be deployed on any set of adaptive mechanisms, thus facilitating the automation of predictive algorithms' adaptation.

Table 3: bDWM Adaptive strategies

Strategy	Description
Sequence0(+RC)	Deploy DAM0 (alternatively with RC) on every batch. This means that only the first batch of data is used to create a model.
Sequence1(+RC)	Deploy DAM1 (alternatively with RC) on every batch.
Sequence2(+RC)	Deploy DAM2 (alternatively with RC) on every batch.
Sequence3(+RC)	Deploy DAM3 (alternatively with RC) on every batch.
Sequence4(+RC)	Deploy DAM4 (alternatively with RC) on every batch.
Sequence5(+RC)	Deploy DAM5 (alternatively with RC) on every batch.
Sequence6(+RC)	Deploy DAM6 (alternatively with RC) on every batch.
Sequence7(+RC)	Deploy DAM7 (alternatively with RC) on every batch.
bDWM(+RC)	Deploy bDWM custom adaptation strategy (optionally with RC).
XVSelect	Select AM based on the current data batch using the cross-validatory approach described in the Section 3.2.
XVSelectRC	Select AM based on the current data batch using the cross-validatory approach using RC as described in the Section 3.2.

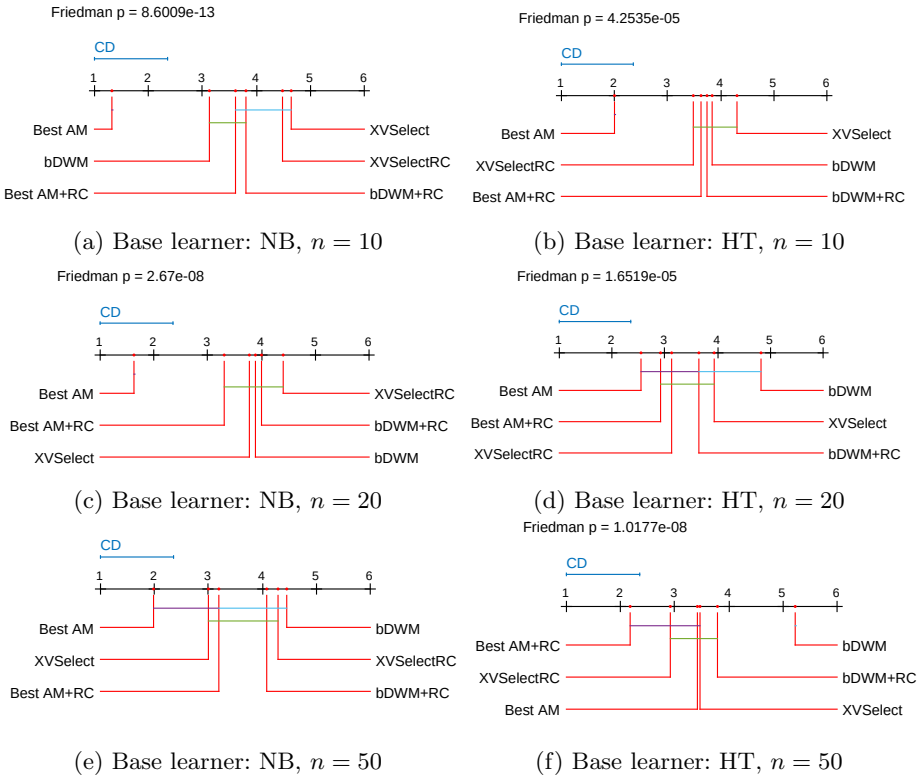
We have used several generic adaptation strategies, based on cross-validation on the current batch and retrospectively reverting the model to the oracle state after obtaining the most recent batch of data. We postulate that the recently seen data is likely to be more related to the incoming data, therefore these strategies tend to steer the adaptation of the predictive model to achieve better results on the most recent available data.

To confirm our assumptions, we have empirically investigated the merit of generic adaptation strategies *XVSelect* and *XVSelectRC*. For this purpose we have conducted experiments on 5 real and 26 synthetic datasets, exhibiting various levels of adaptation need.

The results are promising, as for the majority of these datasets, the proposed generic approaches were able to demonstrate comparable or better performance to those of specifically designed custom algorithms and the repeated deployment of any single adaptive mechanism. We have analysed the cases where proposed strategies performed relatively poor. It is postulated that the reasons for these cases were a) lack of change/need for adaptation, b) insufficient data in a batch and c) relatively simple datasets, all of which have trivial solutions. We have also identified the choice of base learner can affect the performance of proposed strategies.

A benefit of proposed generic adaptation strategies is that they can help designers of machine learning solutions save time by not having to devise a custom adaptive strategy. *XVSelect* and *XVSelectRC* are generally parameter-free, except for number of cross validation folds, choosing which is relatively trivial.

This research has focused on batch scenario, which is natural for many use cases. Adopting the introduced generic adaptive strategies for incremental learn-



(a) Base learner: NB, $n = 10$

(b) Base learner: HT, $n = 10$

(c) Base learner: NB, $n = 20$

(d) Base learner: HT, $n = 20$

(e) Base learner: NB, $n = 50$

(f) Base learner: HT, $n = 50$

Fig. 2: Nemenyi plots of bDWM *XVSelect*, *XVSelectRC* and the most accurate single AM strategy with/without RC for different batch sizes n with NB and HT as base learners.

ing scenario remains a future research question. In that case a lack of batches would for example pose a question of which data should be used for cross validation? This could be addressed using data windows of static or dynamically changing size. Another useful scope of research is focusing on a semi-supervised scenario, where true values or labels are not always available. This is relevant for many applications, amongst them in the process industry. An additional future research direction is theoretical analysis of this direction of research where relevant expert/bandit strategies may be useful.

In general, there is a rising tendency of modular systems for construction of machine learning solutions, where adaptive mechanisms are considered as separate entities, along with pre-processing and predictive techniques. One of the features of such systems is easy, and often automated plug-and-play machine learning [31]. Generic adaptive strategies introduced in this paper further contribute towards this automation.

Bibliography

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

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

[3] Anderson, E.: The Species Problem in Iris. Annals of the Missouri Botanical Garden **23**(3), 457 (sep 1936)

[4] Bakirov, R., Gabrys, B.: 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. vol. 412, pp. 646–656 (2013)

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

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

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

[8] Bifet, A., Holmes, G., Gavaldà, R., Pfahringer, B., Kirkby, R.: 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 (jun 2009)

[9] Cardillo, G.: MYFRIEDMAN: Friedman test for non parametric two way ANalysis Of VAriance (2009), https://www.mathworks.com/matlabcentral/fileexchange/25882-myfriedman

[10] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The UCR Time Series Classification Archive (jul 2015)

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

[12] Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research **7**(Jan), 1–30 (2006)

[13] Domingos, P., Hulten, G.: Mining high-speed data streams. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00 pp. 71–80 (2000)

[14] DrawNemenyi: drawNemenyi (2019), https://github.com/sepehrband/drawNemenyi

[15] Duin, R.P.W., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D., Tax, D.M.J., Verzakov, S.: PRTools4.1, A Matlab Toolbox for Pattern Recognition (2007)
[16] Elwell, R., Polikar, R.: 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 (oct 2011)
[17] Fern, A., Givan, R.: Dynamic feature selection for hardware prediction. Tech. rep., Purdue University (2000)
[18] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and Robust Automated Machine Learning. In: Advances in Neural Information Processing Systems 28 (NIPS 2015). pp. 2962–2970 (2015)
[19] Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics **7**(2), 179–188 (sep 1936)
[20] Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A., Labidi, S. (eds.) Advances in Artificial Intelligence SBIA, pp. 286–295. Springer, Berlin Heidelberg (2004)
[21] Gama, J., Žliobait, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. Computing Surveys **46**(4), 1–37 (2014)
[22] Gomes Soares, S., Araújo, R.: An on-line weighted ensemble of regressor models to handle concept drifts. Engineering Applications of Artificial Intelligence **37**, 392–406 (jan 2015)
[23] Grbić, R., Slišković, D., Kadlec, P.: Adaptive soft sensor for online prediction and process monitoring based on a mixture of Gaussian process models. Computers & Chemical Engineering **58**, 84–97 (nov 2013)
[24] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter **11**(1), 10 (nov 2009)
[25] Harries, M.: Splice-2 comparative evaluation: Electricity pricing. Technical report. The University of South Wales. Tech. rep., The University of South Wales (1999)
[26] Hazan, E., Seshadhri, C.: Efficient learning algorithms for changing environments. In: ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning. pp. 393–400 (2009)
[27] Herbster, M., Warmuth, M.: Tracking the best expert. Machine Learning **29**, 1–29 (1998)
[28] Hernández-Orallo, J., Martínez-Usó, A., Prudêncio, R.B., Kull, M., Flach, P., Farhan Ahmed, C., Lachiche, N.: Reframing in context: A systematic approach for model reuse in machine learning. AI Communications **29**(5), 551–566 (nov 2016)
[29] Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential Model-Based Optimization for General Algorithm Configuration. In: LION'05 Proceedings of the 5th international conference on Learning and Intelligent Optimization. pp. 507–523. Springer, Berlin, Heidelberg (2011)
[30] Jin, H., Chen, X., Yang, J., Zhang, H., Wang, L., Wu, L.: Multi-model adaptive soft sensor modeling method using local learning and online sup-

port vector regression for nonlinear time-variant batch processes. Chemical Engineering Science **131**, 282–303 (jul 2015)
[31] Kadlec, P., Gabrys, B.: Architecture for development of adaptive on-line prediction models. Memetic Computing **1**(4), 241–269 (sep 2009)
[32] Kadlec, P., Gabrys, B.: Local learning-based adaptive soft sensor for catalyst activation prediction. AIChE Journal **57**(5), 1288–1301 (may 2011)
[33] Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. The Journal of Machine Learning Research **Volume 8,**, 2755–2790 (2007)
[34] Kolter, J.J.Z., Maloof, M.A.: Using additive expert ensembles to cope with concept drift. In: ICML '05 Proceedings of the 22nd international conference on Machine learning. pp. 449 – 456. ICML '05, ACM, New York, NY, USA (2005)
[35] Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.: Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. Journal of Machine Learning Research **18**(25), 1–5 (2017)
[36] Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Blackwell (2004)
[37] Lloyd, J.R., Duvenaud, D., Grosse, R., Tenenbaum, J.B., Ghahramani, Z.: Automatic construction and natural-language description of nonparametric regression models. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. pp. 1242–1250. AAAI Press (2014)
[38] Martin Salvador, M., Budka, M., Gabrys, B.: Automatic Composition and Optimization of Multicomponent Predictive Systems With an Extended Auto-WEKA. IEEE Transactions on Automation Science and Engineering **16**(2), 946–959 (apr 2019)
[39] Minku, L., White, A., Xin Yao: The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift. IEEE Transactions on Knowledge and Data Engineering **22**(5), 730–742 (may 2010)
[40] Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (jan 2012)
[41] Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
[42] Schlimmer, J.C., Granger, R.H.: Beyond incremental processing: Tracking Concept Drift. AAAI-86 Proceedings pp. 502–507 (1986)
[43] Schmidt, M., Lipson, H.: Learning noise. Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07 pp. 1680–1685 (2007)
[44] Shao, W., Tian, X.: 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 (mar 2015)
[45] Stanley, K.O.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
[46] Street, W.N., Kim, Y.S.: 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 (2001)

[47] Wasserman, L.: Bayesian Model Selection and Model Averaging. Journal of Mathematical Psychology **44**(1), 92–107 (mar 2000)
[48] Zhu, X.: Stream Data Mining Repository, http://www.cse.fau.edu/~xqzhu/stream.html (2010)
[49] Zliobaite, I.: Combining Similarity in Time and Space for Training Set Formation under Concept Drift. Intelligent Data Analysis **15**(4), 589–611 (2011)