# Towards cost-sensitive adaptation: when is it worth updating your predictive model?

Indrė Žliobaitė[a], Marcin Budka[b], Frederic Stahl[c]

[a]*Aalto University and Helsinki Institute for Information Technology, Espoo, Finland*
[b]*Bournemouth University, Poole, UK*
[c]*University of Reading, Reading, UK*

**Abstract**

Our digital universe is rapidly expanding, more and more daily activities are digitally recorded, data arrives in streams, it needs to be analyzed in real time and may evolve over time. In the last decade many adaptive learning algorithms and prediction systems, which can automatically update themselves with the new incoming data, have been developed. The majority of those algorithms focus on improving the predictive performance and assume that model update is always desired as soon as possible and as frequently as possible. In this study we consider potential model update as an investment decision, which, as in the financial markets, should be taken only if a certain return on investment is expected. We introduce and motivate a new research problem for data streams – cost-sensitive adaptation. We propose a reference framework for analyzing adaptation strategies in terms of costs and benefits. Our framework allows to characterize and decompose the costs of model updates, and to asses and interpret the gains in performance due to model adaptation for a given learning algorithm on a given prediction task. Our proof-of-concept experiment demonstrates how the framework can aid in analyzing and managing adaptation decisions in the chemical industry.

*Keywords:* evolving data streams, concept drift, evaluation, cost-sensitive adaptation, utility of data mining

## 1. Introduction

Learning from evolving data has become a popular research topic in the last decade. As distributions of real world data often evolve over time [1], predictive models need to have mechanisms to update themselves by regularly taking into account new data, otherwise their predictive performance will degrade. Such adaptive predictive algorithms have been developed in different research fields, such as data mining and machine learning in general [2, 3, 4], recommender systems [5], user modeling and personalization [6], information retrieval [7], intrusion detection [8], robotics [9], time series analysis [10], chemical engineering [11] and more. The majority of those algorithms focus on optimizing the prediction accuracy over time. Several studies (e.g. [12, 13]), consider time and memory consumed for this operation as additional performance criteria. Different predictive analytics applications may operate in different environments, generate very different volumes of data, have different complexities of predictive decisions, and different sensitivity to errors. Naturally, there is no single best adaptation strategy or algorithm for all situations ("no free lunch" [14]).

From a practical perspective excessive adaptation (e.g. too often) may be a waste of resources and provide only incremental insignificant benefits towards the predictive performance. Consider as an example a chemical production process where an adaptive predictive model estimates the quality of a product given sensor readings as inputs. Sensor readings may arrive every second. However, changes in the process that require a model update, such as new suppliers or replacement of sensors, are not likely to happen every second, but rather on a yearly basis or so. During that year an incremental learning algorithm would make updates to itself every second, resulting in 30 million incremental updates. The question is, weather it would be a desirable investment of resources.

In response to this question we introduce a research problem of *cost-sensitive adaptation*, where model adaptation is considered as an investment decision. Computational resources are invested for updating the models, and labour resources to obtain feedback (the true labels) expecting to improve the predictive performance. In the financial markets investment decisions are made on the basis of the expected return on investment (ROI) [15]. In predictive systems to estimate ROI of adaptation we need to assess costs and benefits of running an adaptive algorithm. This assessment needs to be performed in a standardized way such that different algorithms and their implementations could be compared before putting an adaptive system into operation. Moreover, in order to fully utilize the opportunities presented by the modern market of computing resources, such as cloud computing, we need to be able to decompose adaptive algorithms into critical and optional components and to assess the costs and contributions of each component independently.

In this study we propose a reference framework for assessing the utility of an adaptive algorithm for a given prediction problem. The framework includes a methodology for identify-

*Email addresses:* indre.zliobaite@aalto.fi (Indrė Žliobaitė), mbudka@bournemouth.ac.uk (Marcin Budka), f.t.stahl@reading.ac.uk (Frederic Stahl)

ing cost components, a model for analytically combining those components, and a setting for experimental assessment of ROI before putting the algorithm to online operation. We propose using our analytical framework and the ROI – the gain in predictive performance per resources invested, for comparing and justifying algorithmic decisions in designing adaptive prediction systems.

Our study makes the following contributions. Firstly, we introduce and motivate a new research problem for data streams – cost-sensitive adaptation. Secondly, we systematically characterize costs of adaptive learning, which are typically ignored in theoretical work, but are critical for real-world applications. Thirdly, the proposed reference framework makes it possible to compare adaptive algorithms *within* a given application context in terms of costs and benefits of adaptation. Different businesses naturally have different costs and benefits. Even if the same measure is used for assessing the predictive performance, the implications of 1% improvement may be very different in different applications. For instance, in the airline industry a 10% improvement in the demand prediction accuracy can bring 2-4% additional monetary revenue [16]. Our proof-of-concept experiments demonstrate how the proposed framework can help in deciding upon an optimal adaptation strategy in a chemical production application.

The paper is organized as follows. Section 2 discusses the requirements for adaptation, and overviews adaptation possibilities offered by available computing resources. In Section 3 we propose a framework and accompanying methodology for quantifying utility of adaptation retrospectively and online in real time. Section 4 characterizes existing adaptive learning algorithms following our framework. Section 5 presents a proof-of-concept experimental analysis that demonstrates how the framework can be used for analyzing learning algorithms. Section 6 discusses related work, and Section 7 concludes the study.

## 2. Overview of requirements and resources

This section provides a context for our study. We first discuss the requirements for adaptation quoted in research literature and illustrate the need for adaptation with a few application examples. Then we present an overview of currently available computing resources, and discuss what are the possibilities for adaptivity from the technical point of view. The goal of this section is to analyze how adaptivity can be organized and to what extent adaptivity needs to and can be flexible (on demand).

### *2.1. Requirements and need for adaptation*

In the data streams literature the following requirements are often quoted [17]: (1) process one example at a time, and inspect it only once; (2) use a limited amount of memory; (3) work in a limited amount of time; (4) be ready to predict at any time.

The requirements (2) and (3) are critical in the online setting, since otherwise a predictive system fails to operate. These requirements originate from the *incremental learning* domain. Incremental learning[18] refers to situations where the learning data arrives and model is updated over time. Adaptive learning is different from incremental learning setting in a way that incremental learning only learns from new data, while adaptive learning in addition needs to forget the old data. Both incremental and adaptive learning algorithms need to scale linearly with the incoming data.

The requirement (1) originates from *one-pass algorithms* that read their inputs exactly once, and generally require $O(n)$ time and less than $O(n)$ memory, where $n$ is the size of the input. Such algorithms traditionally address exploratory analysis of large data, where the output of an algorithm is an answer to a query (e.g. what is the mean of the data). Reading of data is typically a major operation, followed by a basic mathematical operation (e.g. addition). In adaptive learning the output of a training algorithm is a trained predictive model. Reading of data consumes a small share of resources, the majority is consumed by model training or update (e.g. gradient descent or matrix inversion). Thus, in our opinion, multiple passes can be considered if that improves overall utility of the system.

The requirement (4) originates from *any-time* algorithms in computing, which can provide an answer after performing a fixed amount of computation. The answer may be not globally optimal. This requirement is, as well, more critical in exploratory data analysis, where the answer is a summary of data. In predictive modeling the answer is a trained predictive model. We argue, that in adaptive learning the any-time requirement is relevant only in situations where occasionally next observation arrives so fast that the previous model update is not yet finished. In such cases we may consider keeping our previously trained model instead of replacing it by a half-trained new model, i.e. we may chose to adapt the predictive model less frequently. This cannot happen continuously, since in such a case an earlier requirement (3) would be violated.

With respect to the above discussion we conclude that in terms of resources, online adaptive algorithms should: (1) scale linearly with the incoming data in terms of processing time; (2) use limited memory; (3) execute adaptation only if the expected utility is sufficient.

Let us consider two application examples that both require adaptivity, but the constraints for adaptation are very different. TransUnion provides credit scoring services, aggregating information from $\sim$ 83 000 data sources, with the data coming in 4 000 different formats [19]. TransUnion has 8.5 PB of data and receives 100 million updates a day and all this data is stored in their data warehouse. TransUnion uses `Ab Initio`[1], which is close to a brute-force parallelization scheme. The Chief Information Officer of the company states that the major costs associated with data are costs of moving and storing it, while computing is not a problem, as long as it can be parallelized.

Consider an industrial plant producing chemical raw materials. A plant typically runs from one up to twenty processes, with around five of those running in parallel. During the production process sensor readings (e.g. temperature, pressure,

[1] `http://www.abinitio.com/`

flow) are available in real time, the frequency of data arrival can range from less than a second to one hour with a typical frequency of around five seconds. Sensor readings are mostly real-valued. Large processes may have a few thousands of sensors, while a typically process has up to 500. Such data is neither too large nor too speedy, commodity workstations can be used for processing. Typically, neither computing resources, nor storage is a limiting resource, but that, of course bears costs. The major limiting resource is often a feedback in the form of the true target value, as these values are obtained manually, for instance, in a chemical laboratory. Adaptivity is required due to changes in the operation styles or variations in raw material.

The utility of adaptation depends on a particular source of data; in assessing the utility of adaptation it is important to take into account the source of data and realistically assess whether, for instance, adaptation needs to be fully incremental or a buffer of historical data can be easily stored in memory.

### 2.2. Available computing resources and technologies

Online processing of smaller data can be handled by conventional workstations and data warehousing technologies. Big online data may require specific processing and storage solutions, such as *in-memory* data analytics, *in-database* data analytics or *cloud computing*.

**In-memory analytics** is an approach to querying data from RAM, as opposed to querying data that is stored on physical disks. It allows faster query response times. In-memory online analytics on a desktop PC is becoming feasible for many applications, as the 64-bit operating systems can already address 1 TB of memory.

**In-database analytics** refers to processing data within a Database Management System (DBMS) using analytic database platforms that provide parallel processing, partitioning and scalability. That saves efforts of moving data between the database and analytics applications. Two trends can be distinguished: relational DBMS optimized for analytical workloads (ADBMS) and nonrelational systems (NoSQL) for processing multistructured and unstructured data [20]. ADMBS focus on parallel processing, enhanced data structures, data compression and pushing analytical processing into the DBMS. NoSQL systems are primarily targeted for processing multistructured big data. `Hadoop`[2] is a leading nonrelational processing system, which uses the `MapReduce` programming model [21] to divide application processing into small fragments that can be executed concurrently on multiple nodes in a network of commodity workstations.

Recent developments aim to bring real time analytics and parallel and distributed processing together. For example `Spark`[3], aims to bypass Hadoop's batch processing nature by providing distributed event processing capabilities. Also the `StreamCentral`[4] system aims to facilitate real time data analytics in cloud computing environments.

**Cloud computing** delivers computing as a service rather than tools. `Amazon Elastic Compute Cloud`[5] (Amazon EC2) provides resizable computing capacity in the cloud as a web service. Pricing is per memory-time, data transfer *in* and *out* carry additional charges. Moreover, Amazon facilitates a spot market for biding for unused EC2 computing capacity in real time. Public cloud computing web services are also provided by `GoGrid`[6] or `Ninefold`[7].

Growing cloud computing possibilities, and particularly, computing as a service provides the main motivation for studying resource-aware cost-sensitive adaptation. In cloud computing the resources may be released when not in use, which is particularly attractive in the context of adaptive systems, in which the demand for processing and storage for a given application may vary over time depending on the need to adapt. In cloud computing every update of the current model comes at a cost. Thus, in this setting it is particularly important to be able to assess the utility of adaptation and select the most beneficial adaptation regime. Therefore, our further analysis focuses on the cloud computing settings.

## 3. Assessing costs and benefits of adaptation

In this section we consider how to measure the utility of adaptation. First we formally define the setting of adaptive supervised learning and then we present the framework for assessing utility of adaptation and discuss its components.

### 3.1. Problem setting for online prediction

Data continuously arrives over time and we need to predict in real time. Let $X_t$ be the input data at time $t$, and $y_t$ be the target variable which we aim to predict (label). A predictive model is a function $\mathcal{L} : X_t \rightarrow y_t$, which operates online making a prediction for every incoming data observation ($t$ denotes the time of arrival). The model $\mathcal{L}$ can be fixed, or it may be *adaptive*, Table 1 illustrates the procedure in each case.

Table 1: Operation of online predictors at time $t$

| Fixed predictor | Adaptive predictor |
|---|---|
| at time $t$: | |
| (1) receive $X_t$ | (1) receive $X_t$ |
| (2) predict $\mathcal{L} : X_t \rightarrow \hat{y}_t$ | (2) predict $\mathcal{L} : X_t \rightarrow \hat{y}_t$ |
| | (3) receive true $y_t$ |
| | (4) update $\mathcal{L}$ with $(X_t, y_t)$ |

An adaptive predictor has two additional operation steps, it needs to receive feedback (in the form of the true label) and update the model. Both actions bear additional cost. The model update itself includes the following steps:

1. *Adaptation condition check*, e.g. change detection, request by a human operator, or arrival of the true label $y_t$ in case of periodic adaptation.

[2] `http://hadoop.apache.org/`
[3] `http://spark.incubator.apache.org/`
[4] `http://www.virtus-it.com/stream-central-real-time-analytics-solutions/`
[5] `http://aws.amazon.com/ec2/`
[6] `http://www.gogrid.com/`
[7] `http://ninefold.com/`

2. *Model adaptation action*, such as updating model parameters, switching to an alternative model, or deploying a new model part. If the adaptation condition is *not* flagged, the old model will be retained.
3. *Data management action* updates the information in memory that is not used for making predictions, such as a past data buffer or data summary, an alternative predictive model for potential future use or performance statistics for change detection, if required.

Model adaptation is always triggered by the adaptation condition. Data management may be triggered or independent from the adaptation condition.

### 3.2. When to adapt?

This study considers steps (3) and (4) in Table 1 as optional; we propose how to analyze the utility of executing those steps. In the vast majority of cases if step (4) is intended, step (3) must be executed, since supervised learning models need labels for training. The standard *adaptive* predictors aim to adapt whenever opportunity presents itself regardless of resources needed for adaptation (with a few exceptions of active learning restricting labeling costs, e.g. [22]). We argue that adaptation rate can and often should be varied or even suspended depending on expected gains in predictive performance relative to spending on computational resources and labeling of training examples that enable adaptivity. At each time step expected costs and benefits of that adaptation need to be estimated. Adaptation should only happen if the expected gain in performance, such as the prediction accuracy, exceeds the cost of resources required for adaptation.

Our framework focuses on measuring the utility of adaptation. Let $\mathcal{L}_T$ be the current predictive model. Suppose at time $t_1$ adaptation is triggered and we produce a model $\mathcal{L}_{T+1}$. If we *do not update* the model at time $t_1$, we would save some computational and labeling costs, but the model would potentially lose some accuracy. To measure that effect we retain the old model $\mathcal{L}_T$ to operate in parallel until the next update at time $t_2$, which allows assessing the gains of adaptation within the time horizon $[t_1, t_2]$, based on the difference in performance of $\mathcal{L}_T$ and $\mathcal{L}_{T+1}$, as well as assessing the costs of adaptation associated with maintaining $\mathcal{L}_{T+1}$ adaptable (i.e. change detectors, data buffers, alternative models). At $t_2$ we build $\mathcal{L}_{T+2}$, and replace the reference model $\mathcal{L}_T$ with $\mathcal{L}_{T+1}$. This assessment requires small additional computational costs, however, we expect the costs of operation of existing model to be negligible as compared to the costs of training new models and obtaining the labels.

Decisions for the future adaptation can be based on observing the trends of utility, i.e. if utility is decreasing or remains at unsatisfactory level, then adaptation needs to become less frequent or stop.

### 3.3. The gain of adaptation

The gain of adaptation can be measured retrospectively in terms of a change in *loss* $\lambda(\mathcal{L}_T(X_t), y_t)$, where $\mathcal{L}_T$ denotes the predictive model after the $T^{th}$ adaptation, and $L$ is the loss function. The gain of the $T^{th}$ adaptation is given by

$$\gamma_T = \sum_{t=T_1}^{T_{N_T}} \Big[\lambda\Big(\mathcal{L}_{T-1}(X_t), y_t\Big) - \lambda\Big(\mathcal{L}_T(X_t), y_t\Big)\Big]. \quad (1)$$

where $N_T$ is the number of samples after the $T^{th}$ adaptation but before the $T + 1^{st}$ adaptation, and $T_i$ is the time index of the $i^{th}$ sample after the $T^{th}$ adaptation. Note that $\mathcal{L}_{T-1}$ is just a copy of the previous predictive model, it does not require any extra change detection or adaptation itself.

The simplest loss function is the absolute prediction error, defined as $\lambda : |\mathcal{L}_T(X_t) - y_t|$. In this case gain is a decrease in prediction error, which can also be called an increase in the prediction accuracy. Alternative loss functions can be defined depending on an application at hand, for instance, by assigning different relative costs to various types of incorrect predictions [23]. For example, in a binary classification problem of medical diagnosis, we can assign higher cost of classifying a person with a serious disease as healthy than the other way around.

In the usual circumstances the loss function should be fixed throughout the online operation. For comparing alternative predictors under the proposed framework, the same loss function should be used or losses from different functions should be converted to monetary units.

### 3.4. The cost of adaptation

The costs of adaptation are not limited to computational costs of model updates. We pay for keeping the option to adapt even if the actual model update does not happen. Such costs may include: keeping a data buffer in memory, periodically running change detectors or training alternative models for potential future use. Adaptation can also generate indirect (*opportunity*) costs for delayed or missed predictions due to resources being engaged elsewhere or the predictive model's ongoing updates. Moreover, for updating the model the true labels are required, which may impose costs. Last but not least, communication costs may occur to supply the predictor with new data and to retrieve results. Thus, adaptation costs consist of four components: (1) computational costs, (2) costs of delayed predictions, (3) labeling costs, and (4) communication costs, which we discuss in the following subsections.

#### 3.4.1. Computational cost

In order to perform model adaptation computational resources need to be engaged. The costs of computational resources include the processing power, memory consumption and disk storage. When computation is outsourced (e.g. cloud computing discussed in Section 2.2) disk storage is usually priced in slots of a reasonably large quantity (e.g. each 500GB). Thus, we assume them to constitute the fixed costs of running the predictive system independently whether it is adaptive or not. Hence in assessing the utility of adaptation we only take into account the processing power and memory consumption.

Recently Bifet et al proposed [13] to use RAM-hours for quantifying the consumption of resources in online learning. A

RAM-hour (Rh) refers to 1 GB of RAM engaged for one hour, and we use it as a basis for quantifying computational costs. RAM-hour is a standard unit of pricing in cloud computing services. That is, many providers of computing services charge per RAM-hour, hence computing costs in RAM-hours can straightforwardly be translated into monetary costs. Growing availability of computing as a service is one of the main motivations for this work on assessing the utility of adaptation. If we can asses the benefit of extra (computing) resources deployed, we can deploy exactly as much resources as is beneficial.

The computational costs of adaptation $\psi^{com}$ relate to:

- PROCESSING TIME of
  - $C_t$ data observation interarrival time,
  - $c^P$ making prediction (not adaptation costs),
  - $c^D$ verifying if model update is needed,
  - $c^L$ updating predictive model,
  - $c^B$ obtaining and updating data summaries, including alternative decision models that are not currently in use, and
- REQUIRED MEMORY for
  - $m^D$ verifying if model update is needed,
  - $M^D$ storing detectors and statistics for detection,
  - $m^L$ updating predictive model,
  - $M^L$ storing predictive model (not adaptation costs),
  - $m^P$ making prediction (not adaptation costs),
  - $m^B$ updating data summaries or alternative models,
  - $M^B$ storing raw data, data summaries and/or alternative models in the memory buffer.

In the above, $M$ denotes resources that are constantly employed, while $m$ denotes resources that are deployed only during the corresponding processing actions.

Consider the time period between two model updates. We define the computational costs for the $T^{th}$ update as:

$$\psi_T^{com} = \sum_{t=T_1}^{T_{N_T}} \Big[ C_t(M_t^D + M_t^B) + (c_t^B m_t^B + c_t^D m_t^D + c_t^L m_t^L) \Big]. \quad (2)$$

$N_T$ is the number of samples after the $T^{th}$ adaptation but before the $T+1^{st}$ adaptation, $T_i$ is the $i^{th}$ sample after $T^{th}$ adaptation. The first component occupies memory constantly, since we need to store the detector ($M^D$) and data buffer ($M^B$) continuously if we opt for an adaptive system. The second component is the memory occupied temporarily during checks and updates. Note, that $M^L$ and $c^P$ are not accounted as these costs are present in an online prediction system regardless of it being adaptive or not. We assume the system operation in the setting of on-demand computational resources (e.g. cloud computing), where unused resources can be released at most once per time step of operation (a time step $t$ has been defined in Table 1), therefore, the maximum of the operational resources for detection, model update or data buffering is considered in Eq. (2).

#### 3.4.2. Opportunity cost

The opportunity costs (loss) occur if the system is unable to deliver predictions at the time of adaptation because model update is being performed. This cost highly depends on the application and in general may be difficult to quantify. In safety-critical applications a prediction may be needed, for instance, at every second, and any delay may have catastrophic consequences. In other applications, e.g. recommender systems, skipping predictions may not be that critical, if overall the system is improved.

The delayed prediction cost can be mitigated at the expense of extra RAM-hours. One possible scenario is to use as many RAM-hours as needed in order to perform adaptation within a fixed time. However, in practice any adaptation procedure can only be parallelized up to some point, after which employing additional computational resources will have no effect. That is known as the *maximum speed up*. Moreover, usage of parallelization frameworks imposes computational administration overheads. Hence, it is important that the data mining task is large enough (in computational terms) to be able to gain enough speed up from parallelization in order to be beneficial. For example setting up a task in `Hadoop` will take some time. In the Hadoop manual it is recommended that each subtask takes at least one minute to execute[8].

Alternatively we can deploy a backup copy of the predictive model, which would be used for providing predictions, while the original model is adapting, at an expense of additional RAM-hours, storage and data transfer. The cost of deploying and using this copy is defined as

$$\psi_T^{opp} = \sum_{t=T_1}^{T_{N_T}} c_t^L m_t^L. \quad (3)$$

If for a given application model adaptation is expected to be fast and a delay in prediction would not make a major harm, we may choose not to deploy a copy of the model and assume $\psi_T^{opp} = 0$. We also would not need to deploy a copy in cases where adaptation affects only restricted parts of the model, meanwhile other parts can deliver meaningful predictions, e.g. retraining one expert in an ensemble of classifiers.

#### 3.4.3. Labeling cost

Timely feedback in the form of the true labels is required for the adaptation of predictive models. Obtaining the labels may be a manual resource-intensive, and hence costly procedure. The costs of labeling $\psi_T^{lab}$ need to be expressed in comparable units to $\psi_T^{com}$ and $\psi_T^{opp}$, thus converting $\psi_T^{lab}$, $\psi_T^{com}$, $\psi_T^{opp}$ into monetary costs may be required. In forecasting tasks, such as weather or sales prediction, labels become available the next time period at almost no cost. Therefore in this study we assume $\psi_T^{lab} = 0$, but the proposed framework is flexible enough to accommodate nonzero or even variable labeling cost.

[8] `https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html`

#### 3.4.4. Communication cost

Using computational services, such as cloud services for adaptive or nonadaptive predictive systems requires to transmit data from the source to the service, and from the service to the user in the form of predictions. The cost of communicating predictions may be neglected as it is usually very small in size. However, communicating training data or unlabeled data for the prediction may impose a bottleneck and cause higher cost. Cost in this context is caused for consuming a certain bit rate (measured in bits per second – $bps$) over a period of time.

We distinguish two kinds of communication costs with the cloud service: the *general communication cost* of prediction, and the *extra* costs associated with having the predictive system adaptive. The general communication cost is dependent on the number of data items that need to be predicted by the system, these costs are the same for nonadaptive and adaptive systems and hence are not part of the cost of adaptation. The extra communication cost $\psi_T^{ntw}$ associated with having the system adaptive are caused by communicating new training observations to the cloud service. This consumption may vary depending on the required speed of adaptation, as the training observations may have to be made available in time to accommodate for the required speed of adaptation. The cost of communication $\psi_T^{ntw}$ needs to be expressed in units comparable to $\psi_T^{com}$, $\psi_T^{opp}$ and $\psi_T^{lab}$ and may be converted into monetary costs if required. In some cases $\psi_T^{ntw}$ may be zero or small enough to be neglected. This could be the case if the bandwidth is already available and does not need to be purchased, or if the cloud services are provided in-house.

#### 3.4.5. The total cost of adaptation

The overall cost of the $T^{th}$ adaptation is given by:

$$\psi_T = \psi_T^{com} + \psi_T^{opp} + \psi_T^{lab} + \psi_T^{ntw}, \tag{4}$$

where $\psi_T^{com}$ and $\psi_T^{opp}$ are defined in Eq.(2) and (3), $\psi_T^{lab}$ is defined in Section 3.4.3, $\psi_T^{ntw}$ is defined in Section 3.4.4. With respect to the arguments discussed in this study we assume that the major cost is the computational cost, hence $\psi_T \approx \psi_T^{com}$ while $\psi_T^{opp}, \psi_T^{lab}, \psi_T^{ntw} \approx 0$. For instance, in the credit scoring and chemical plant examples in Section 2 data storage and processing capacities could be varied depending on the deployed data analysis methods, while other costs would stay more or less fixed. Thus, we focus on accurately assessing $\psi_T^{com}$.

### 3.5. Return on resources invested (ROI)

Return on Investment (*ROI*) is a standard performance measure used to asses the attractiveness of an investment [15]. This metric is used for measuring, per period, rates of return on money invested in an economic entity in order to decide whether or not to undertake an investment. It is also used as indicator to compare different project investments within a project portfolio. The one which gives the best return (ROI) is prioritized. As a decision tool it is simple to compute and understand (return per unit of resources invested), and gives the means for standardized comparison across different applications.

One adaptive (or nonadaptive) predictive modeling algorithm can be seen as a single investment project. Our goal is not to choose the best performing predictor regardless of the cost, but the best performing predictor considering the resources invested. For example, 1% improvement in prediction accuracy in spam filtering, may be less beneficial than 1% improvement in bankruptcy prediction. Hence, frequent updates may potentially be more beneficial in the second case.

We propose using *ROI* for quantifying the gain in performance per resources (*Rh*) invested. The return on RAM-hours over the time period between adaptations $T$ and $T+1$ is

$$ROI_T = \frac{\gamma_T}{\psi_T}, \tag{5}$$

gains $\gamma_T$ are defined in Eq.(1), costs $\psi_T$ in Eq.(4).

We can make use of *ROI* measurement in three cases. Firstly, we can use streaming $ROI_T$ as any other performance statistic, such as streaming error, for monitoring the performance online. For that after each adaptation action we need to compute the gain in Eq.(5) since the previous adaptation action.

Secondly, we can measure the weighted average $\overline{ROI}$ of all the adaptations over a given period of time retrospectively to judge about the overall effectiveness of the employed adaptation strategy. This overall $\overline{ROI}$ is given by

$$\overline{ROI} = \frac{\sum_{i=1}^{T'} (N_i \times ROI_i)}{\sum_{i=1}^{T'} N_i} = \frac{1}{\sum_{i=1}^{T'} N_i} \sum_{i=1}^{T'} N_i \frac{\gamma_i}{\psi_i}, \tag{6}$$

where $T'$ is the total number of adaptation actions, gains $\gamma_i$ are defined in Eq.(1), costs $\psi_i$ in Eq.(4) and (2).

Finally, a baseline $\underline{ROI}$ can quantify the return on employing an adaptive predictor as compared to keeping a fixed nonadaptive model

$$\underline{ROI} = \frac{\gamma}{\psi}. \tag{7}$$

In this setting we assume that all data corresponds to one long adaptation period ($T = 1$). $\mathcal{L}_0$ is the fixed predictor and $\mathcal{L}_i$ is the adaptive predictor at time $i$. The gains in Eq.(7), as defined in Eq.(1), become

$$\gamma = \sum_{t=1}^{N_{T'}} \Big[ \lambda(\mathcal{L}_0(X_t), y_t) - \lambda(\mathcal{L}_t(X_t), y_t) \Big], \tag{8}$$

Following the assumption, that all the costs except computational are zero (opportunity, labeling and network costs), the costs in the denominator of Eq.(7) become

$$\psi = \sum_{t=1}^{N_{T'}} \Big[ C_t(M_t^D + M_t^B) + (c_t^B m_t^B + c_t^D m_t^D + c_t^L m_t^L) \Big]. \tag{9}$$

### 3.6. The price of evaluation

The evaluation of *ROI* carries some computational costs itself, that is a price of evaluation. The costs are related to storing and executing the previous model $\mathcal{L}_{T-1}$ for assessing the gain of adaptation given in Eq.(1).

A data analyst needs to be aware of those costs; however, we do not recommend including those costs into *ROI* computation.

The reason is that *ROI* is supposed to quantify, how beneficial it is to deploy an adaptive model. The benefit is there independently of whether we are running the evaluation of it or not.

The computational price of *ROI* evaluation over the period between adaptations $T$ and $T+1$ is given by the cost of storing an additional predictive model $M^L$ , the cost of making an additional prediction $c^P$, the cost of storing all the cost variables $m^{ROI}$ (about 10 variables) and the cost of computing *ROI* after each adaptation $c^{ROI}$.

$$\pi_T = \sum_{t=1}^{N_{T'}} \left[ C_t M_t^L + c_t^P m_t^P \right] + c^{ROI} m^{ROI}. \quad (10)$$

In practice we expect $\pi_T$ to be much smaller than the cost of adaptation $\psi^T$, since making an extra prediction using an existing model is typically less computationally intensive than updating the model. The latter belongs to the costs of adaptation. As an example, consider linear regression, where prediction is just a weighted sum of inputs, while model update requires computationally intensive matrix inversion or gradient descent.

*ROI* evaluation would not incur extra opportunity costs, would not require any additional labeling costs, and the extra communication costs in case of online evaluation would be negligible. The input data sent to the server would still be the same. The data coming back from the server would have one extra value (*ROI*) in addition to predictions, which need to be broadcasted periodically anyway.

The proposed framework defines costs and benefits of adaptation and allows assessing the utility of a given adaptive learning algorithm in a given application context. Next we will characterize existing adaptation algorithms within our framework.

## 4. Adaptation strategies

Many adaptive learning algorithms have been developed, each offering different benefits, but also requiring different resources for adaptation. In terms of resources, five types of adaptation strategies can be distinguished.

**Fully Incremental (FI).** The predictive model is updated using only the previous model and the latest data observation: $\mathcal{L}_T = f(\mathcal{L}_{T-1}, \{X_t, y_t\})$, here $f$ is the model update function, $t$ is the current time and $T = t$ is the counter of model adaptations. Adaptation is performed on a sample-by-sample basis and *only* the information necessary for producing predictions is stored; no models in progress, data summaries or extra parameters are stored.

**Summary Incremental (SI).** The predictive model is updated using the previous model, the latest data observation and a data summary: $\mathcal{L}_T = f(\mathcal{L}_{T-1}, \{X_t, y_t\}, \mathcal{L}')$. This data summary can be stored as an alternative predictor $\mathcal{L}'$ that is not used for making predictions yet, or sufficient statistics of the incoming data, for instance, the covariance matrix. Summary Incremental strategies can be further split into:

- *fixed memory* methods, where the space for storing a data summary does not depend on the data covered (e.g. storing the covariance matrix for updating linear regression);
- *variable memory* methods, where data summaries can grow in line with the arriving data (e.g. data summary in a form of an alternative decision tree).

**Batch Incremental (BI).** The previous model is updated with a batch of past observations that are stored in a memory buffer: $\mathcal{L}_T = f(\mathcal{L}_{T-1}, \{X_{t-b}, y_{t-b}\}, \ldots, \{X_t, y_t\})$, where $b$ is the size of the buffer. The update does not require rebuilding the model from scratch. Some versions of this strategy may require a summary of the past data, leading to a **Summary Batch Incremental (SBI)** strategy.

**Nonincremental (NI).** The model is rebuilt from scratch on a batch of past observations every time adaptation is required: $\mathcal{L}_T = f(\{X_{t-b}, y_{t-b}\}, \ldots, \{X_t, y_t\})$. Typically the batch to be stored in memory is much larger than in BI to make up for the loss of information caused by discarding the old model.

Table 2 summarizes the strategies. $(X_t, y_t)$ is the current data observation, $\mathcal{L}_{T-1}$ is the latest available predictive model and $\mathcal{L}'_{T-1}$ is a data summary, $b$ is the batch size.

Table 2: Adaptation strategies

| | |
|---|---|
| Fully Incremental | $\mathcal{L}_T = f(\mathcal{L}_{T-1}, \{X_t, y_t\})$ |
| Summary Incremental | $\mathcal{L}_T = f\left(\mathcal{L}_{T-1}, \mathcal{L}'_{T-1}, \{X_t, y_t\}\right)$ |
| Batch Incremental | $\mathcal{L}_T = f(\mathcal{L}_{T-1}, \{X_{t-b}, y_{t-b}\}, \ldots, \{X_t, y_t\})$ |
| Summary Batch Incremental | $\mathcal{L}_T = f\left(\mathcal{L}_{T-1}, \mathcal{L}'_{t-1}, \{X_{t-b}, y_{t-b}\}, \ldots, \{X_t, y_t\}\right)$ |
| Nonincremental | $\mathcal{L}_T = f(\{X_{t-b}, y_{t-b}\}, \ldots, \{X_t, y_t\})$ |

Above adaptation strategies apply to individual prediction models as well as ensembles of models. In the Fully and Summary Incremental strategies every labeled data observation has an immediate impact on the following predictions. The Fully Incremental adaptation does not require a change detector, it updates periodically (P), while Summary Incremental typically requires a change detector (D). Batch incremental and Nonincremental adaptation may work with or without a change detector.

Adaptation is not always a yes-no decision. Incremental learning algorithms can be modified to smoothly forget the past information using observation weighting [3]. Such algorithms fit well into our framework as FI or BSI strategies updating periodically.

Note, that strategies that update periodically consume memory and processing power evenly over time. The strategies with change detection, on the other hand, have peaks in processing time when changes are detected. At those times predictive models are updated, and at the same time the memory buffer is released. Strategies, which are using change detection, may have larger peaks in memory consumption when changes are not detected for a longer time. During no change periods they accumulate a long buffer of observations in memory, but at the same time have lower computation costs, which are only used for change detection and making predictions, no model updates are happening. Variation of time and memory consumption is an important aspect to take into account when considering periodical adaptation against detection based adaptation.

Examples of Fully Incremental algorithms with forgetting mechanisms can be found in [24, 25], where DWM [25] is

a Fully Incremental ensemble. Summary Incremental algorithms often employ a change detector that signals the (re)start of storing a data summary in a form of an alternative predictive model. This model is maintained in parallel before a change is confirmed, and it becomes accurate enough. In that respect CVFDT [26], and DDM [27] can be considered as Summary Incremental strategies. Adaptive ensembles that retrain and replace individual members on the latest batch (such as [28]) can be considered Batch Incremental.

Developing an incremental algorithm for model update is a challenge and not all of the base learners have those versions, therefore an alternative is to retrain a model from scratch every time adaptation is required (e.g. [3]). Retraining from scratch has another important advantage, which is the ability to optimize additional parameters, which are not optimized by the learning algorithm itself (e.g. the number of principal components in PCA).

In Table 3 we characterize each adaptation strategy in terms of typical memory consumption ($m$ and $M$) and typical processing time ($c$ and $C$) defined in Section 3. For the strategies with change detection we provide an indication of maximum, minimum and expected mean time and memory; however, practically the required resources may vary a lot depending on the frequency of changes.

**FI** requires minimum amount of memory, since it does not require to store data or detectors ($M^D = 0$ and $M^B = 0$, as well as update them $m^D = 0$, $m^D = 0$). Therefore, FI is expected to generate very little overhead in terms of RAM-hours due to adaptation, only to cover for $c^L m^L$. Thus, $\psi^{com} = c^L m^L$. FI however adapts even if changes are not happening. Although in the limit its performance (loss) should converge to that of the batch counterpart, in practice it can take many thousands of samples. Thus FI is likely to exhibit the highest loss.

**SI** may store data summaries in a form of an alternative predictive model, thus $M^B > 0$, but it is expected to be small. Updating the summary model is comparable to updating the original model $c^B m^B \leq c^L m^L$. If we need to store and operate a detector, it is an overhead, which is likely to be $c^D m^D > c^L m^L$. Thus, $\psi^{com} \leq C(M_D + M_B) + c_D m_D + 2c^L m^L$.

At the other end of the spectrum there is the loaded with features NI strategy, which is likely to produce low loss but is resource-hungry. **BI** and **NI** are expected to have high memory consumption due to data storage (typically higher for NI than for BI), while updating the data storage should not be resource-intensive $c^B m^B \leq c^L m^L$. If there is no change detection ($M^D = 0$ and $m^D = 0$), then $\psi^{com} \leq CM^B + 2c^L m^L$. If there is a change detector, typically $c_D m_D < c_L m_L$, particularly for NI, then $\psi^{com} < C(M^D + M^B) + 3c^L m^L$.

This leads to a trade-off situation and in practice one may need to find a compromise. Hence, BI may often appear as the most viable choice, given that the incremental version of the chosen learning algorithm is available. We emphasize that our analysis should be seen as an indication of expectations, one can always find algorithms that may constitute exceptions.

Table 3: Performance: ↓ low, ⇝ medium, ↑ high, ↑↑ very high; (D) with detection, (P) periodic

| Adaptation strategy | FI (P) | SI (D) | BI (P) | BI (D) | NI (P) | NI (D) |
|---|---|---|---|---|---|---|
| Processing time [$c$,$C$] | ↓ | ↓ | ⇝ | ⇝ | ↑ | ↑ |
| *max time* | | | | ↑ | | ↑↑ |
| *min time* | | | | ↓ | | ↓ |
| Memory [$m$,$M$] | ↓ | ⇝ | ↑ | ↑ | ↑↑ | ↑↑ |
| *max memory* | | | | ↑ | | ↑↑ |
| *min memory* | | | | ⇝ | | ⇝ |
| Frequency of adaptation | ↑ | ↑ | ⇝ | ↓ | ⇝ | ↓ |
| Expected costs [$R$] | • | • | •• | •• | ••• | ••• |
| Expected gains [$B$] | ○ | ○○ | ○ | ○○ | ○○ | ○○○ |

## 5. Experimental Analysis

The goal of this experimental analysis is to demonstrate how a basic instantiation of the proposed evaluation framework could help in the evaluation of adaptive learning algorithms. The intention is to present a proof-of-concept case rather than examining a wide spectrum of methods and problems.

### 5.1. Data

The experiments are performed on the Catalyst Activation Data, which has been made publicly available within the NiSIS 2006 competition[9]. The dataset originates from a chemical reactor, which consists of some 1000 tubes filled with catalyst, used to oxidize a gaseous feed in a exothermal reaction counteracted by the cooling, which 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. The process to be modeled takes input from other, larger processes, so that the feed will vary over several days. The operating personnel reacts to this by choosing appropriate operating conditions. The catalyst decay is, however, much slower than these effects.

The data contains 13 input variables (originally there were 14, we discarded one with a constant value), and one continuous target variable – the activity of the catalyst, corresponding to an operating period of 8 months (242 days × 24 hours). Data is standardized prior to the experiments.

### 5.2. Predictive model

We use the Recursive Partial Least Squares (RPLS) [29] as the inner predictive model to be integrated with the adaptation strategies. Following the classification presented in Section 4, RPLS uses a Summary Incremental adaptation strategy, since it stores data summaries required for updating the model with new data, and updates the model recursively. RPLS also can easily be used as a batch nonincremental learner. The choice of RPLS has been dictated by the popularity of PLS in the process industry [30].

[9] `http://www.nisis.risk-technologies.com`

### 5.3. Experimental protocol

We test and compare two strategies: incremental (BSI) and nonincremental (NI) with periodic adaptation at different adaptation rates $k$ (time intervals at which adaptation is executed). The size of the data buffer (fixed window) for NI is set to be the same as the adaptation rate. Note, that if $k = 1$ BSI becomes SI, because adaptation is no longer executed in batches, but with every new observation. These strategies have been selected for comparison, since they offer an interesting opportunity to analyze the performance across a spectrum of possible parameter settings ($k$), while keeping the rest of the settings nearly identical.

The goals of the experiments are to:

- Demonstrate how to select an optimal adaptation strategy and adaptation rate for a given dataset using $\underline{ROI}$ offline.
- Illustrate how the proposed $ROI_T$ can be used for monitoring and analyzing the benefits of adaptation online.

In each experiment we first train an initial model $\mathcal{L}_0$ on 120 data points and then process the dataset from observation #121 with a fixed adaptation rate $k$ from 1 to 240. The initial training includes simple optimization of the number of latent variables ($N_{LV}$) in terms of the norm of the input data residual matrix. The chosen $N_{LV}$ is then kept constant during model updates, unless a full model retraining is performed.

The performance of the models is assessed using the test-then-train evaluation protocol [31]. Following this protocol each incoming observation is first used for testing the current model, and then for updating the model. We use the Mean Absolute Error (MAE) as the loss function.

All the experiments are coded in MATLAB, which provides built in functions for measuring processing time and memory used by data structures at a fine granularity. All the costs are calculated as defined in Eq.(2). The costs of adaptation are measured in RAM-seconds.

### 5.4. Analyzing the utility of adaptation offline

Consider an online prediction scenario where data is expected to evolve over time. A snapshot of historical data is available for designing a prediction system and testing it before putting it to online operation. Suppose we have a choice between two adaptive algorithms using BSI or NI adaptation strategies. We also need to fix the adaptation parameter $k$.

With the current evaluation practice a data scientist would run the two algorithms with different parameter settings on the historical snapshot and select the one which gives the lowest testing error. Figure 1 (top) presents an example of such evaluation, where the best algorithm is considered to be the one giving the lowest testing MAE. In our example, the data scientist would deploy BSI with $k = 1$, since it gives the best predictive performance. However, this evaluation does not take into account computational costs. If new data arrives every second, updates would happen every second.

The proposed *ROI* evaluation would suggest a different strategy, as illustrated in Figure 1 (bottom). Here the higher the $\underline{ROI}$, the better. We can see from the plot that the best tradeoff

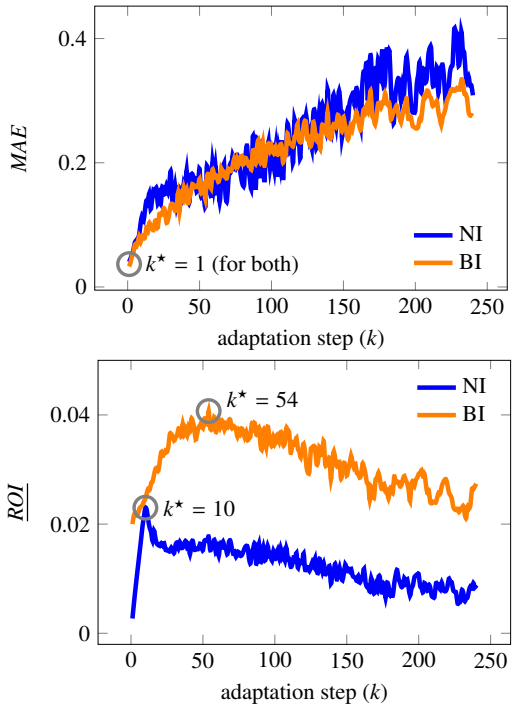


Figure 1: Evaluation of adaptive algorithms using *MAE* vs. *ROI*. Circles denote best performance in each criteria. $k^\star$ denote optimal adaptation rates.

between the computational costs and achieved accuracy is at $k = 54$. It suggests, that for this prediction problem it is worth having an adaptive model ($\underline{ROI}$ is positive), but it is not worth adapting at every time step.

We can see that the proposed evaluation allows to capture three aspects of the performance in one measure: accuracy, benefits over a nonadaptive model and computational costs. In addition, the proposed evaluation can be used for selecting the optimal adaptation parameters. If we assessed only prediction accuracy in a conventional way, we would be pushed to spend unnecessary large amount of computational resources. The proposed framework, on the other hand, indicates that the optimum trade-off between the gains in accuracy and the resources invested can be achieved at a less frequent adaptation rate. The proposed $\underline{ROI}$ evaluation appears to be more informative from the application perspective, and could particularly be useful in the environments where the processing power can be consumed in a flexible way, for instance, in a cloud.

One more interesting observation follows from this experimental analysis. We compare two adaptive systems that use very similar settings and the same base predictor (PLS regression) based only on accuracy. As it can be seen, the performance does not differ considerably (Figure 1, top) and in general the loss increases with $k$. However, the proposed integrated *ROI* measure clearly shows that the incremental version (BSI) is more beneficial or is a better investment of resources, which is the result of lower computational costs at similar gain. At the same time, as it can be expected, NI has lower accuracy at small adaptation rates, since it does not accumulate sufficient amount of data to rebuild a good predictive model.

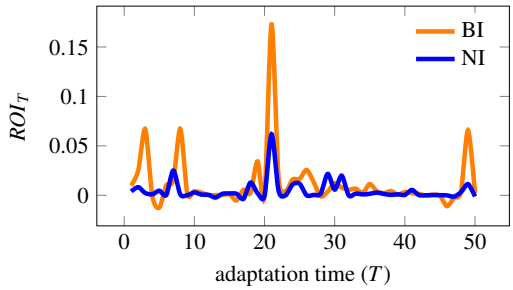


Figure 2: Monitoring *ROI* online.

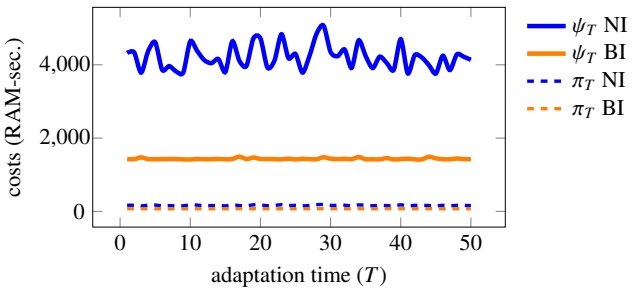


Figure 3: Analysis of the costs of adaptation.

### 5.5. Analyzing the utility of adaptation online

Consider another online prediction scenario, where an adaptive predictor is deployed, and the performance of the system needs to be monitored. The proposed *ROI* evaluation may be used for monitoring and diagnostics of performance in the same way as any other streaming statistic, such as streaming error. After each adaptation action $T$ we can calculate $ROI_T$, and, for instance, plot the statistic over time. This way the progress of *ROI* can be monitored, and if the return is unsatisfactory, the adaptation rate can be decreased or the adaptation can be suspended all together, while the fixed current model continues to provide predictions.

Figure 2 illustrates such monitoring of $ROI_T$ for a fixed adaptation rate $k = 110$ for illustration purposes. We can see that there are two periods of low return, one starts around adaptation action #10 and the other around #30. If we were using cloud computing services with dynamic allocation of resources, at about time $T = 15$, after observing a continuous period of low return, we could consider making adaptation less frequent, since adaptation at those times does not give a reasonable return on the resources invested. Reducing the adaptation frequency would make the computational costs go down, and in turn $ROI_T$ can go up.

Monitoring *ROI* and online evaluation carries extra computational costs of storing an alternative predictive model and computing the performance statistics. However, as discussed in Section 3.6, these costs are expected to be negligible as compared to the costs of model updates. Figure 3 plots the costs of adaptation $psi_T$ and the costs of evaluation $\pi_T$. We can, indeed, see that the costs for evaluation is by orders of magnitude lower, than the costs for adaptation, and can be considered negligible in the overall system design.

The experimental analysis demonstrates that important information about the cost-benefit tradeoffs becomes available when using the proposed cost-sensitive evaluation strategy, that enables more precise evaluation of the performance offline for design choices, and online for monitoring and fine-tuning the system in operation.

## 6. Related work

From the algorithmic perspective our approach to monitoring the performance over time may resemble optimization with adaptive learning rates [32, 33]. Adaptive learning rates in optimization refer to dynamically adjusting the search step in search for the optimum. The goal is typically to arrive at the optimum faster. While the adaptive learning rate procedure is concerned with finding the final optimum, which is fixed, our monitoring of *ROI* aims to remain at the optimum position while the optimum itself is moving over time, which falls into the class of dynamic optimization problems [34]. In our future research we plan to explore the theory of dynamic optimization in search for theoretically optimal adaptation rates for individual adaptive learning strategies.

Our setting is also related to cost-sensitive learning [35], which is the problem of optimal learning and decision making when different misclassification errors incur different penalties. The settings are similar as they aim to guide the learning process by incorporating costs into decision making. There are two major differences though. First, cost-sensitive learning traditionally concerns learning at the model level, while we consider learning at a system level. Moreover, traditionally cost-sensitive learning concentrates on the costs of different types of errors [36], while in our framework cost of errors are considered to be the same and the costs are incurred due to the need to update the model due to nonstationarity of data.

Finally, as we have already discussed, the trade-off between accuracy and computational costs has been analyzed in a recent study on data streams [13], where RAM-hours were proposed as a measure for quantifying the consumption of resources in online learning, which is a standard basis for pay-as-you-go pricing in cloud computing.

## 7. Conclusions and future work

We introduced a new research problem of cost-sensitive adaptation and proposed a reference framework for assessing the utility of adaptation in online predictive modeling tasks. The proposed framework defines the components of gains and costs in adaptive online learning, and proposes a way to measure the return resulting from adaptation of the model on the resources invested. As we saw in the airline example, business can estimate concrete monetary gains that can result from improved predictive accuracy. The proposed framework enables a standardized assessment and comparison of different adaptive algorithms within a given application context. An analytical assessment of costs and benefits is essential before setting up a predictive system for a long term online operation.

Our proof-of-concept experiments in chemical industry domain comparing two distinct adaptation strategies were intended as a demonstration of the potential of our framework. The experiments also demonstrated that the proposed approach can help in finding an optimal adaptation rate for a given application. The analysis showed that the proposed measure can be effectively applied for retrospective analysis of adaptive learning algorithms performance and online monitoring of the utility of adaptation.

Our work opens a range of new interesting research avenues for data streams. Firstly, a thorough investigation of individual costs components, their interactions and tradeoffs would enable much more precise assessment of the utility, and more importantly, possibly theoretical estimation of utility given an adaptive learning algorithm. Secondly, we analyzed several simple adaptation strategies. A comparative analysis of a wide range of existing adaptive learning algorithms in terms of adaptation costs would provide a good systematization of data streams research and useful guidelines to practitioners.

From the resource perspective, our study used a simplified representation of a public cloud model as a motivation and the cost of communication has been described in basic terms. A focused more fine-grained study on communication costs presents an interesting future research direction. Different network types as well as exploiting network latency times need to be addressed. For example, if the available network bit rate is fluctuating over time, then it may be desirable to wait for the available bit rate to increase before new data is supplied to the cloud (independently of the required speed of adaptation).

Finally, another exciting and promising follow up research direction would be to develop a methodology for taking into account possible variations in spot prices of computational resources that could help to make more informed decisions whether to adapt in real time.

**Acknowledgements**. This research has been supported by the EC within the Marie Curie IAPP programme (grant agreement no. 251617), and the Academy of Finland CoE ALGODAN (grant 118653).

## References

[1] D. Hand, Classifier technology and the illusion of progress, Statistical Science 21 (2006) 1–34.

[2] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.

[3] R. Klinkenberg, Learning drifting concepts: Example selection vs. example weighting, Intell. Data Anal. 8 (3) (2004) 281–300.

[4] E. Ikonomovska, J. Gama, S. Dzeroski, Learning model trees from evolving data streams, Data Min. Knowl. Discov. 23 (1) (2011) 128–168.

[5] Y. Koren, Collaborative filtering with temporal dynamics, Commun. ACM 53 (4) (2010) 89–97.

[6] D. Billsus, M. Pazzani, User modeling for adaptive news access, User Modeling and User-Adapted Interact. 10 (2000) 147–180.

[7] F. Menczer, G. Pant, P. Srinivasan, Topical web crawlers: Evaluating adaptive algorithms, ACM Trans. Internet Technol. 4 (2004) 378–419.

[8] W. Lee, S. J. Stolfo, K. W. Mok, Adaptive intrusion detection: A data mining approach, Artif. Intell. Rev. 14 (2000) 533–567.

[9] D. Stavens, G. Hoffmann, S. Thrun, Online speed adaptation using supervised learning for high-speed, off-road autonomous driving, in: Proc. of the 20th int. joint conf. on Artif. intell., IJCAI, 2007, pp. 2218–2224.

[10] Y. Le Borgne, S. Santini, G. Bontempi, Adaptive model selection for time series prediction in wireless sensor networks, Signal Proc. 87 (2007) 3010–3020.

[11] P. Kadlec, R. Grbic, B. Gabrys, Review of adaptation mechanisms for data-driven soft sensors, Comp. & Chem. Eng. 35 (1) (2011) 1–24.

[12] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavalda, New ensemble methods for evolving data streams, in: Proc. of the 15th ACM SIGKDD int. conf. on Knowl. discovery and data mining, KDD, 2009, pp. 139–148.

[13] A. Bifet, G. Holmes, B. Pfahringer, E. Frank, Fast perceptron decision tree learning from evolving data streams, in: Proc. of the 14th Pacific-Asia conf. on Advances in Knowledge Discovery and Data Mining, PAKDD, 2010, pp. 299–310.

[14] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. on Evol. Comp. 1 (1) (1997) 67–82.

[15] D. Brealey, S. C. Myers, Principles Of Corporate Finance, 6th Edition, McGraw-Hill, 1999.

[16] S. Riedel, B. Gabrys, Combination of multi level forecasts, The Journal of VLSI Signal Processing 49 (2007) 265–280.

[17] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online analysis, J. of Mach. Learn. Res. 11 (2010) 1601–1604.

[18] C. Giraud-Carrier, A note on the utility of incremental learning, AI Commun. 13 (2000) 215–223.

[19] PWC, Making sense of big data, Technology Forecast 3.

[20] C. White, Using big data for smarter decision making, Report, BI Research (2011).

[21] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (2008) 107–113.

[22] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes, Active learning with evolving streaming data, in: Proc. of the 2011 Eur. conf. on Mach. learn. knowl. discov. in databases, ECMLPKDD, 2011, pp. 597–612.

[23] R. Duda, P. Hart, D. Stork, Pattern Classification 2nd ed., John Wiley & Sons, New York, USA, 2001.

[24] J. Gama, G. Castillo, Adaptive bayes, in: Proc. of the 8th Ibero-American Conference on Advances in Artificial Intelligence, 2002, pp. 765–774.

[25] J. Kolter, M. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, J. of Mach. Learn. Res. 8 (2007) 2755–2790.

[26] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proc. of the 7th ACM SIGKDD int. conf. on Knowledge discovery and data mining, KDD, 2001, pp. 97–106.

[27] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symp. on Artif. Intell., SBIA, 2004, pp. 286–295.

[28] H. Wang, W. Fan, P. S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proc. of the 9th ACM SIGKDD int. conf. on Knowledge discovery and data mining, 2003, pp. 226–235.

[29] S. Qin, Recursive PLS algorithms for adaptive data modeling, Computers & Chemical Eng. 22 (4-5) (1998) 503–514.

[30] L. Fortuna, Soft sensors for monitoring and control of industrial processes, Springer Verlag, 2007.

[31] J. Gama, R. Sebastião, P. P. Rodrigues, On evaluating stream learning algorithms, Machine Learning 90 (3) (2013) 317–346.

[32] Z. Luo, On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks, Neural Comp. 3 (2) (1991) 226–245.

[33] C. Chinrungrueng, C. Sequin, Optimal adaptive k-means algorithm with dynamic adjustment of learning rate, IEEE Trans. on Neural Networks 6 (1) (1995) 157–169.

[34] M. Carter, Foundations of Mathematical Economics, MIT Press, 2001.

[35] C. Elkan, The foundations of cost-sensitive learning, in: Proc. of the 17th int. joint conf. on Artif. intell., IJCAI, 2001, pp. 973–978.

[36] P. Turney, Types of cost in inductive concept learning, in: Workshop on Cost-Sensitive Learning at ICML, 2000.