Engineering Applications of Artificial Intelligence 34 (2014) 178–192

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Enhancing data stream predictions with reliability estimators and explanation

Zoran Bosnić [a,*], Jaka Demšar [a], Grega Kešpret [a], Pedro Pereira Rodrigues [b], João Gama [c], Igor Kononenko [a]

[a] University of Ljubljana, Faculty of Computer and Information Science, Slovenia
[b] CINTESIS – Faculty of Medicine, University of Porto, Portugal
[c] LIAAD – INESC Porto, L.A. & Faculty of Economics, University of Porto, Portugal

ARTICLE INFO

Article history:
Received 23 February 2014
Received in revised form
28 April 2014
Accepted 2 June 2014
Available online 28 June 2014

Keywords:
Data stream
Incremental learning
Prediction accuracy
Prediction correction
Prediction explanation

ABSTRACT

Incremental learning from data streams is increasingly attracting research focus due to many real streaming problems (such as learning from transactions, sensors or other sequential observations) that require processing and forecasting in the real time. In this paper we deal with two issues related to incremental learning – prediction accuracy and prediction explanation – and demonstrate their applicability on several streaming problems for predicting electricity load in the future. For improving prediction accuracy we propose and evaluate the use of two reliability estimators that allow us to estimate prediction error and correct predictions. For improving interpretability of the incremental model and its predictions we propose an adaptation of the existing prediction explanation methodology, which was originally developed for batch learning from stationary data. The explanation methodology is combined with a state-of-the-art concept drift detector and a visualization technique to enhance the explanation in dynamic streaming settings. The results show that the proposed approaches can improve prediction accuracy and allow transparent insight into the modeled concept.

1. Introduction

In supervised learning we aim to induce a model on training data so we can afterwards make predictions for the examples that were not included in the learning process. In the field of online learning from data streams it is particularly challenging to achieve good accuracy and quick predictions because the set of examples is being continuously incremented. As available computer memory puts an upper bound to the storage space and users often require predictions in real-time, this calls for an adaptation of standard supervised learning techniques. These adaptations include windowing of data (i.e. holding only a sample of examples) and learning by incrementally modifying a model with the most recent examples instead of re-building it from all previous examples. The ability to increment the learned knowledge allows the model to adapt to changing data distribution, i.e. to situations where a *concept drift* occurs.

In this paper we propose an adaptation of two existing methodologies to incremental learning: prediction correction methodology and prediction explanation methodology. Application of both incremental learning add-ons can increase the trust of users into predictions and their understanding of modeled knowledge, which is also a main goal of our work. So far, both methodologies have been developed and evaluated only for learning from stationary data, while in this paper we focus on their adaptation to streaming scenarios.

The first adapted methodology is built on top of the *reliability estimation for individual predictions in regression*. The reliability estimation methodology augments the bare prediction value K_u for example x_u with its reliability estimate r_u and thus predicts a tuple (K_u, r_u). Note that this methodology does not evaluate accuracy of the model as the whole but provides prediction performance estimates for each individual example $x_u, u \in 1 \ldots n$. In this paper we use two methods for computing such reliability estimates to assess the prediction error and correct it accordingly: they are based on the sensitivity analysis (Bosnić and Kononenko, 2008b) and local modeling of prediction error (Bosnić and Kononenko, 2008a).

The second approach, the *explanation methodology for individual predictions and models*, provides an insight into not only the importance of attributes but also the strength of particular attribute values with which they either positively or negatively contribute to the final prediction. Formally, the methodology extends a prediction K_u for example x_u into a tuple $(K_u, (\varphi_1,$

* Corresponding author.
E-mail addresses: zoran.bosnic@fri.uni-lj.si (Z. Bosnić), jaka.demsar0@gmail.com (J. Demšar), grega.kespret@gmail.com (G. Kešpret), pprodrigues@med.up.pt (P. Pereira Rodrigues), jgama@fep.up.pt (J. Gama), igor.kononenko@fri.uni-lj.si (I. Kononenko).

http://dx.doi.org/10.1016/j.engappai.2014.06.001
0952-1976/© 2014 Elsevier Ltd. All rights reserved.

$\varphi_2, \ldots, \varphi_m$)), where φ_j denotes a contribution of the j-th attribute, $j \in 1 \ldots m$. The contributions are computed using a model-independent wrapper algorithm (Štrumbelj and Kononenko, 2010). We adapt this approach to storage limitations in incremental learning and propose a novel way to visualize the occurring concept drift.

The paper is structured as follows. In Section 2 we overview relevant achievements in the field of incremental learning and describe reliability estimation and prediction explanation in greater detail. Section 3 presents our approach to correcting streaming predictions and Section 4 presents our adaptation of explanation methodology to incremental learning. Our final conclusions are summarized in the last Section 5.

2. Related work

The topic of our research strongly relates to the field of incremental learning from data streams. A data stream is an ordered sequence of instances that can be read only once or a small number of times using limited computing and storage capabilities. The data elements in the stream arrive online, being potentially unbounded in size. Once an element from a data stream has been processed it is discarded or archived. These sources of data are characterized by being open-ended, flowing at high-speed, and generated by nonstationary distributions (Gama and Rodrigues, 2007; Gama et al., 2013). Learning techniques which operate through fixed training sets and generate static models are obsolete in these contexts. Faster answers are usually required, keeping an anytime data model and enabling better decisions, possibly forgetting older information (Hulten et al., 2001).

In the following we review the reliability estimation for individual predictions, related prediction correction, and the explanation methodology for individual predictions and models, which we later adapt to an incremental environment.

2.1. Reliability estimation for individual predictions

When computing a prediction for an unseen example, we cannot estimate its accuracy if the example's true value is unknown. Reliability estimators (Bosnić and Kononenko, 2009) try to quantitatively reason about individual predictions (their accuracy, error residuals etc.) using various data and model characteristics. In contrast to evaluating the averaged accuracy of the whole model, evaluating reliability of individual predictions provides significant additional information for establishing trust into predictions, which is a necessity in decision-critical environments such as medicine, economics, finance and industry.

Prior to reviewing the related work, note that in this paper we use the term “reliability estimator” as a hypernym for various measures that estimate trust in individual predictions: non-exhaustive list of them includes *confidence*, *credibility*, *stability*, *prediction interval* and *confidence interval*. Whereas *reliability estimator* denotes a method (an algorithm), we denote the computed values for each individual example as *reliability estimates*. In general, we can separate reliability estimators into two groups: *model-dependent*, which exploit particular properties of predictive models (such as number of support vectors and splitting nodes in a regression tree) and *model-independent*, which only exploit properties of learning data. In the following, let us review some of related work for each of the groups separately.

Within the group of model-dependent reliability estimators, Gammerman et al. (1998) and Saunders et al. (1999) proposed measuring *confidence* as the probability of the second most probable class and used it for extending the support vector machine (SVM) algorithm for classification. In their work they showed that their estimate correlates well with predictive accuracy of individual example. Next, Li and Wechsler (2005) introduced the measures of *confidence* and *credibility* and showed them to be informative estimates of misclassification probability within the application of face recognition problem. Nouretdinov et al. (2001) redefined the measure of *confidence* for the ridge regression where they improved the initial accuracy using residuals of learning examples and probabilistic modelling. Weigend and Nix (1994), Heskes (1997) and Carney and Cunningham (1999) focused on proposing a local variance estimator for the multilayer perceptron model by extending the output layer with additional nodes and adapting the learning algorithm.

Model-independent reliability estimators are based on *sensitivity analysis* of predictive models, local modeling of prediction error or density estimation of the example space. The first of these approaches, the sensitivity analysis, observes the magnitudes of changes in model output as a result of inducing changes in its input data set (Bousquet and Elisseeff, 2002). By combining the measured changes appropriately we were able to successfully estimate local bias and variance in our previous work (Bosnić and Kononenko, 2008b). The second approach, the local error modeling, implies the reliability of an example from reliability of other local examples. The third approach, density estimation, estimates reliability under the premise that the accuracy is greater in example-richer areas of input space. In our previous work (Bosnić and Kononenko, 2008b, 2008a) we presented nine reliability estimates that are based on the former approaches. Their evaluation revealed promising results for their usage in critical application domains of machine learning. The proposed methodology was later also implemented on an oncological prognostic problem (Štrumbelj et al., 2010) and arterial stenosis problem (Bosnić et al., 2012).

The proposed reliability estimates have been also preliminarily evaluated in data streaming environments (Rodrigues et al., 2008), where the results revealed significant correlations of locality based and sensitivity analysis approaches with the prediction error. This motivates us to evaluate these estimates for correcting predictions, on which we focus in this paper.

2.2. Explanation of predictive models and individual predictions

Researchers in machine learning focus on explaining logic in predictive models to improve transparency of decision support systems and seek arguments for their predictions. This is a challenging task because many predictive models lack an intrinsic mechanism for interpreting their outputs. For example, neural networks and random forests do not provide a human-interpretable reasoning on the logic between the inputs and outputs. On the other hand, in decision trees, following branches of a tree provides an interpretation of predictions, but the approach is model-specific only.

Other model-specific approaches include explaining a naive Bayes model using information gains of attributes (Becker et al., 1997; Kononenko, 1993; Možina et al., 2004) and explaining Bayesian networks (deSantana et al., 2007), tools for explaining the importance of individual attributes for Random Forests (Breiman, 2001), extracting rules from neural networks (Andrews et al., 1995; Craven and Shavlik, 1994; Towell and Shavlik, 1993), and several methods for Support Vector machines (Hamel, 2006; Jakulin et al., 2005; Poulet, 2004).

The model-specific methods have a disadvantage of lacking a uniform representation of explanation across different models. Whenever we change the predictive model, the user has to adapt to a new explanation method. This requires extra time and effort which is what practitioners dislike. In contrast, the advantage of model-independent methods is that they can be applied to an arbitrary predictive model. Several such approaches, which observe the sensitivity of the model by changing the values of attributes, exist (Lemaire et al., 2008; Robnik-Šikonja and Kononenko, 2008); however, they cannot detect interactions between attributes, for example their disjunctive dependencies, which results in less informative explanations. The method of Štrumbelj and Kononenko (2010) has overcome this drawback. To explain the whole model, this method assigns a contribution to each attribute value describing how a particular value affects the prediction for a given instance. The influence of a certain attribute value is defined as the expected change in model's prediction when the attribute's value is omitted from the instance. The method is generalized so that instead of observing the influence of single attributes, the influences of all subsets of attributes are considered. In order to obtain a global contribution for the attribute's value the contributions across different instances are averaged.

3. Correcting online predictions

In this section we present our approach for correcting online predictions using reliability estimators *CNK* and *SAbias* (Bosnić and Kononenko, 2008a). We start by describing these two estimators and their repurposing for correcting predictions. Since the correction methodology is based on the reliability estimation for regression predictions, we evaluate our adapted approach on a real regression domain – electricity load prediction data for the state of New York, USA.

In our further notation, we use $L=\{(x_1,C_1),\dots,(x_n,C_n)\}$ to denote a learning set of examples, where $x_i, i=1\dots n$ denote the attribute vectors and $C_i, i=1\dots n$ denote the true target values of the examples. The goal of the regression task is to compute a prediction K_u for a given unseen example $(x_u,\_)$ as closely as possible to its true value C_u that is unknown to the learning algorithm. We also compute a reliability estimate r_u for $(x_u,\_)$ using a reliability estimator R. In our case, we will use *CNK* or *SAbias* as a reliability estimator R; using these two estimators (algorithms) we will compute reliability estimates CNK_u and $SAbias_u$ for each particular unseen example x_u.

3.1. Reliability estimators

Reliability estimator *CNK* models the prediction error in the local neighborhood of the query example and reliability estimator *SAbias* uses sensitivity analysis to analyse prediction stability (Bosnić and Kononenko, 2008a). Both of these reliability estimators estimate *reliability* in terms of estimating *prediction error* for a particular example. Depending on the algorithm of reliability estimators, their values do not necessarily belong to the same scale as the prediction error itself, but only correlate with it.

In contrast to other model-independent reliability estimators that we evaluated in our previous work and which correlate only to error magnitudes (their values are always non-negative), the estimators *CNK* and *SAbias* produce estimate values that correlate to magnitude and direction of the prediction error. This means that their higher absolute values denote the magnitude of the estimated error and the sign denotes its direction. In this work we explore the latter fact firstly to transform the estimator's scale to the scale of prediction errors and secondly to correct the predictions with the estimated error.

3.1.1. Reliability estimator CNK

The reliability estimator *CNK*[1] estimates the prediction error locally in the neighborhood of the query example. In batch (i.e. non-incremental) learning, we compute the reliability estimate CNK_u for a particular unseen query example $q_u=(x_u,\_)$ as follows. First, we induce a regression model on learning set L and compute the prediction K_u for that particular unseen query example. In the next step we localize the set of the k nearest neighbors of q_u, $N(q_u)=\{(x_{u1},C_{u1}),\dots,(x_{uk},C_{uk})\}$, where $N(q_u)\subseteq L$. Reliability estimate CNK_u is then defined as the difference between the average true target value of the nearest neighbors and the prediction K_u:

$$\mathrm{CNK}_u=\frac{\sum_{i=1}^{k}C_{ui}}{k}-K_u \qquad (1)$$

In the above equation, k denotes the number of neighbors, C_{ui} denotes the true target values of neighbors and K_u denotes the prediction for the query example q_u. The construction of the estimate CNK is illustrated in Fig. 1. In our experiments, five neighbors were used for the computation of CNK, as proposed in Bosnić and Kononenko (2008a).

3.1.2. Reliability estimator SAbias

The reliability estimator *SAbias*[2] (Bosnić and Kononenko, 2008b) estimates prediction error based on the sensitivity analysis of the prediction model. The sensitivity analysis is performed for each query example $q_u=(x_u,\_)$ to determine how stable is the model when making that particular prediction. The approach presumes that the *stable* models will not substantially change their predictions for a given example if the learning data change slightly.

We observe the influence between changes in input and output as follows. First, we build a predictive model in traditional way and compute prediction K_u, which we call the *initial prediction*, for example q_u. Next, we expand the learning set with an additional example q_u, which is the same example of interest from the first step. As this example lacks its target value, we assign it a value $K_u+\varepsilon\cdot value\_range$, where K_u is the example's initial prediction (i.e., the best guess for the true label), ε is a sensitivity parameter that will be modified in the further steps, and *value_range* represents the interval width of true values (i.e. the difference between maximum and minimum true value in the learning set).

In the following steps the modified data set (with $n+1$ examples) is used to build a set of models with varying values of parameter ε. We use each of these models to compute prediction for the q_u again. Because the models slightly differ, their predictions (which we call *sensitivity predictions*) differ as well. Therefore, by using a set of parameters $\varepsilon_1,\varepsilon_2,\dots,\varepsilon_l$ and by using each parameter with its positive and its negated value, we obtain a set of sensitivity predictions:

$$\{K_{u,\varepsilon_1},K_{u,-\varepsilon_1},K_{u,\varepsilon_2},K_{u,-\varepsilon_2},\dots,K_{u,\varepsilon_l},K_{u,-\varepsilon_l}\}$$

where K_{u,ε_i} is the sensitivity prediction for q_u computed using parameter $\varepsilon_i, i=1,\dots,l$.

Different parameters $\varepsilon_i, i=1,\dots,l$ allow us to explore the sensitivity of the model depending on varying magnitudes of incurred change in the dataset. To estimate reliability for q_u we combine the computed sensitivity predictions into a single metric – reliability estimator *SAbias* – as follows:

$$SAbias_u=\frac{\sum_{i=1,\dots,l}(K_{u,\varepsilon_i}-K_u)+(K_{u,-\varepsilon_i}-K_u)}{2l}. \qquad (2)$$

The computation of *SAbias* is illustrated in Fig. 2. In our experiments we used $\varepsilon_i\in\{0.01,0.1,0.5,1.0,2.0\}$, as proposed in Bosnić and Kononenko (2008b).

Note that the estimator *SAbias* returns value 0 when the sensitivity predictions match the initial prediction. In this case

[1] In the original work, acronym CNK stands for "$\underline{C}$ of $\underline{N}$ eighbors minus $\underline{K}$", where C denotes the true value and K the prediction for a query example.

[2] In the original work, acronym SAbias stands for *Sensitivity Analysis bias*.

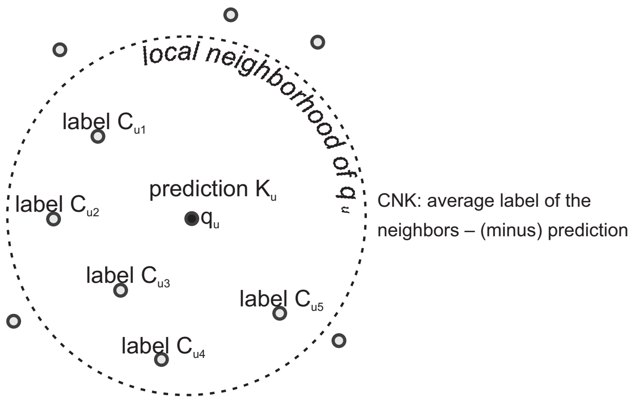
local neighborhood of q_u
label C_u1
label C_u2
prediction K_u
q_u
label C_u3
label C_u5
label C_u4
CNK: average label of the neighbors – (minus) prediction

Fig. 1. Construction of the reliability estimate *CNK*.

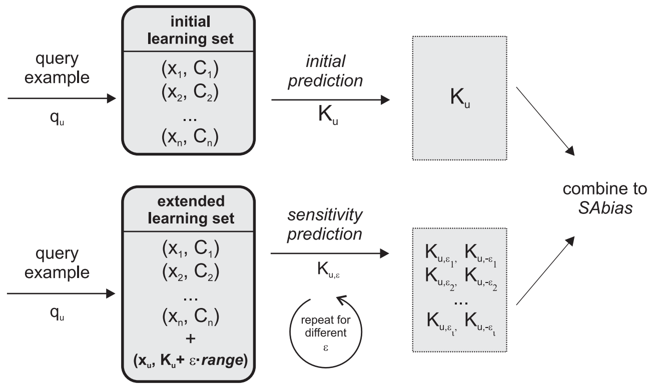

Fig. 2. Construction of the reliability estimate *SAbias*.

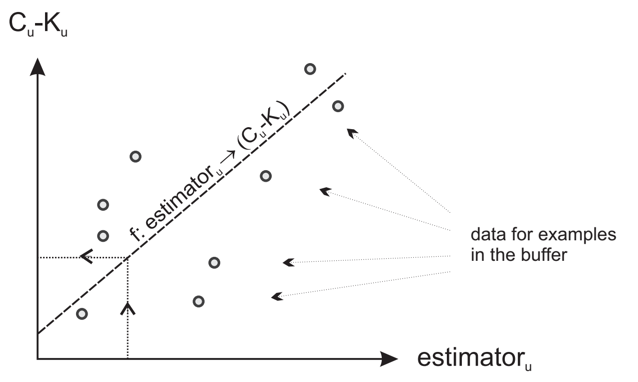

Fig. 3. Modeling dependency between estimators' outputs ($estimator \in \{CNK, SAbias\}$) and prediction error for examples in the buffer.

we assume that the initial prediction is the most reliable. In other cases the metric estimates whether the model more likely under- or overestimates the initial prediction in the sensitivity steps. In this paper we exploit this fact to use *SAbias* for correcting predictions in incremental learning.

3.2. Correction of predictions in incremental learning

To repurpose reliability estimators from Section 3.1 for correcting streaming predictions, we need to make two adaptations: (i) adapt reliability estimates to online learning scenario and (ii) transform estimators' output values to the scale of prediction error magnitudes. Let us focus on each of these two tasks in the following.

Adaptation of reliability estimators to online learning (task (i) above) has drawbacks that stem from the limited storage space and limited computational time. Firstly, as the stream evolves, the examples in the buffer change; consequently, the nearest neighbors of the same query example q_u are different at different times. This impacts the locality-based methods such as *CNK* which therefore exploit less information from the data and return more varying outputs. Secondly, the limited number of examples in the buffer prevents correct estimation of data parameters such as maximum/minimum target value (interval width) that we require for computation of *SAbias*. We could indeed incrementally compute these parameters through the stream history, but if meanwhile concept drifts occur, the results would be misleading. We therefore decided to use both estimators with examples in the buffer only, intending to observe the impact of this limitation on results.

In the second adaptation task we need to transform the estimators' outputs to the scale of prediction error magnitudes. This step is required because the noise in data causes noise in estimator outputs; hence, it makes sense to model a trend between reliability estimates of examples in the buffer and their prediction errors (when their true target values become known). We unavoidably require such model for the *SAbias* estimator – its outputs *only correlate* to prediction errors (Bosnić and Kononenko, 2008a) and do not represent prediction errors themselves (they are computed as the average difference between initial and multiple sensitivity predictions) in contrast to *CNK*, which already estimates the local prediction error. In our work we used linear regression to model the former dependencies $f_C : CNK_u \rightarrow (C_u - K_u)$ and $f_S : SAbias_u \rightarrow (C_u - K_u)$ for a given example q_u. Models f_C and f_S are built from the examples in the buffer and are used to predict the prediction error for new examples of interest. The purpose of this model is illustrated in Fig. 3.

Within the adaptation of reliability estimators to correct predictions, we have to consider time complexity of both reliability estimators, which is high, as well as the time complexity of the linear regression transformation. For *CNK*, we have to compute distances to all examples in the buffer and for *SAbias*, we have to build multiple models in each iteration. These high computational costs conflict with limitations in data stream processing; however, both reliability estimators can be useful if the prediction problem allows it (i.e., it demands low prediction frequency and therefore allows sufficiently wide time gap to perform all computations). The linear regression, on the other hand, is computed only for examples in the buffer and needs to be modeled in two dimensions only; its slope and intersection can therefore be computed efficiently analytically.

To correct the value of the initial prediction, we sum the value of the initial prediction and the predicted error of reliability estimators. By this we obtain the following three predictions:

- K_u: the initial prediction for the example q_u,
- $K_u + f_C(CNK_u)$: corrected initial prediction using estimator *CNK*, and
- $K_u + f_S(SAbias_u)$: corrected initial prediction using estimator *SAbias*.

We compare these three predictions for accuracy in the sections that follow.

3.3. Experimental evaluation

We evaluated the method from Section 3.2 with electricity load consumption prediction data, which is an important and relevant problem for electrical distributors (Bunn and Farmer, 1985). These companies plan their business operations based on predictions for various times ahead, for example for 1 h, 1 day or 1 week ahead.

In the following subsections we present the used data, compare initial and corrected predictions and present the results of our evaluation.

3.3.1. *Experimental data*

Our experimental dataset contains electricity consumption data for the state of New York, USA (NYISO, 2012). These data have been collected by the New York Independent System Operator (NYISO) since 2001 for 11 different areas of the state of New York abbreviated as WEST, GENESE, CENTRL, NORTH, MHK VL, CAPITL, HUD VL, MILLWD, DUNWOD, N.Y.C., and LONGIL. Up to year 2012 the data contain 13.346.802 examples, which represent electricity load measurements taken approximately in 5-min intervals. Because the data for 11 different areas are independent of each other, each having own patterns in data distribution, we considered them as 11 independent data streams.

An example segment of a NYISO data is shown in Fig. 4. In the figure we can note daily cycles of the electricity consumption and an anomaly that occurs approximately in the middle of the stream segment. Since the data contain plenty of anomalies (sensor malfunctions, missing values and noise) and was measured in uneven intervals, the NYISO data had to be preprocessed; we also had to transform the time series into attributional data that is suitable for supervised learning. We used the following preprocessing steps:

- To remove invalid sensor values we applied a mean-shift outlier detector that observes the change in data mean by computing the Studentized residual (Fox, 2002). If anomaly is detected, the erroneous data are replaced with data median for the same hour of day of the last 3 days.
- The inter-arrival time between two examples may fluctuate from less than 4 min up to multiples of 10 min. We transformed the data into regular time series by adjusting the period to 1 h.
- The original data series contain only two attributes: timestamp and data load value. To represent these data as a supervised learning problem we transformed the series into an attributional representation where each example represents 1 h in a day and contains the following attributes (suggested in Rodrigues and Gama, 2009):
  - *Load*: target load value (true value) for that hour,
  - *Sin.24*, *Cos.24*, *Sin.168* and *Cos.168*: periodical attributes that specify a point in a daily (24-h) and weekly (168-h) cycle, correspondingly (for example, the same time of each day has the same values of Sin.24 and Cos.24 and the same in

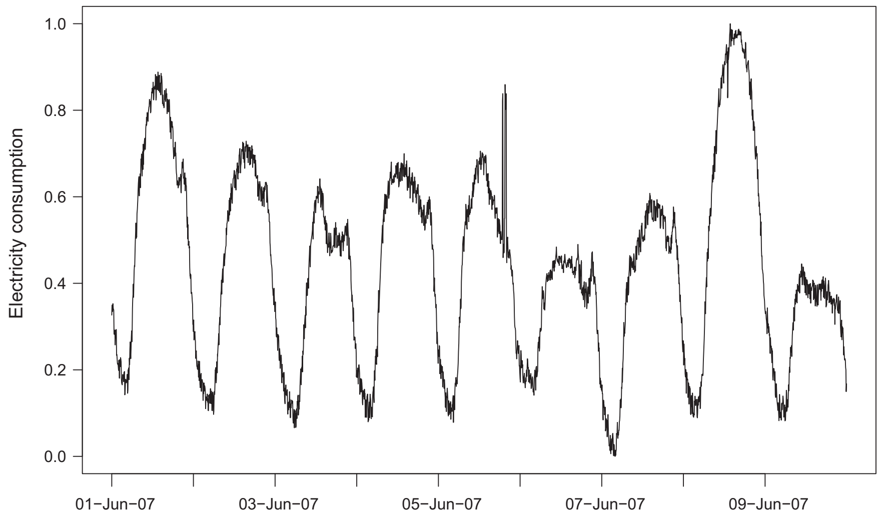

Fig. 4. An example segment of a NYISO data for the region WEST. In the figure we can note daily cycles of the electricity load and an anomaly in sensor readings that occurs approximately in the middle of the data.

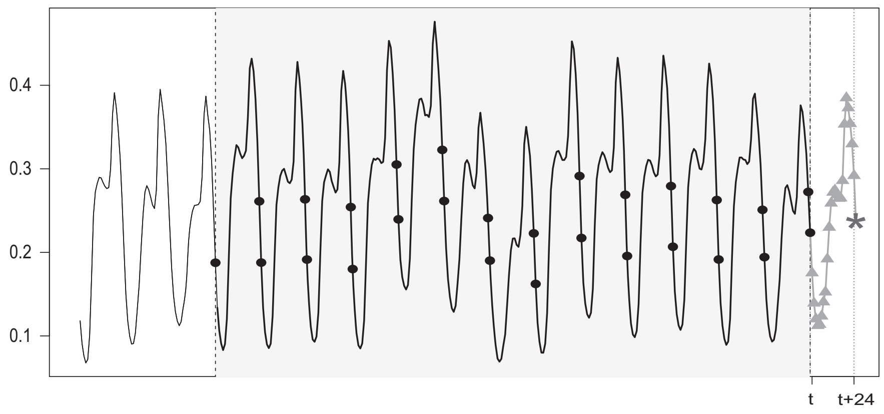

Fig. 5. Incremental prediction of electricity load for 24 h in the future: at time t the prediction for $t+24$ h is computed using the points that are denoted with black dots. The predictions on the interval $(t, t+24]$ (denoted with triangles) will be used to evaluate the model as the window slides forward and their true values become known.

the weekly period has the same values of all four attributes). These values are used in learning as they are expected to relate to daily and weekly electricity load cycles,
  - *Minus.xx*, where $xx \in \{24, 48, \ldots, 336, 25, 49, \ldots, 313\}$: value of electricity load for *xx* hours in the past. First 14 elements of *xx* represent load values for the same hour of a day for each of the past 14 days and the remaining elements represent the load values 1 h before that. Note that by defining such attributes, we simulate a buffer that contains past examples for up to 14 past days (336 examples).

- Since there are 31 attributes, we perform attribute filtering prior to learning. We used a wrapper approach with backwards selection which iteratively removes attributes that improve the root mean squared error of the model the least (Kuhn, 2012). The procedure was performed with cross-validation and was terminated when the error with removal of another attribute would start to increase.

3.3.2. Prediction problem

The goal of our prediction problem was to predict values of electricity load for 24 h in the future, which is a typical prediction question in electricity load prediction problem (Kyriakides and Polycarpou, 2007). The prediction in our incremental supervised learning setting is performed as illustrated in Fig. 5 and explained in the following.

In the experiment, the data set is processed example-by-example. We use each consecutive example to update the prediction model, which is generated from scratch only at the beginning of the experiment and incrementally modified in all the following steps. After the model is incrementally modified, we move on to the next example and repeat the process. When the first 24 examples are processed, we reach the examples for which we made predictions at the beginning of the experiment and can use their true values now to evaluate the former predictions. We perform this evaluation through the rest of the stream until all examples are processed.

Note that the attributes of examples represent the most important points in history for making a target (in our case, one-day-ahead) prediction in the future. We use this *historical* (or *lagged*) attributes as well as the separate buffer of the most recent examples to simulate learning window which shifts forward in each step. In our case, the "oldest" attribute represents a load value of 336 h (14 days) in the past (see Section 3.3.1 for details); consistently, our buffer also holds 336 examples that have been processed last. These examples in the buffer are used within reliability estimators *CNK* and *SAbias* and to evaluate the model's performance.

For predicting the future load values we experimented with an artificial neural network which has been evaluated as an appropriate incremental model for this domain in the related work (Rodrigues and Gama, 2009). We used the implementation in the library AMORE in statistical package R. We treated each of 11 areas of NYC as an independent data stream; for each of them we performed attribute selection of 10 best attributes (as described in Section 3.3.1) and selection of parameters (number of hidden neurons, learning rate). For selection of parameters, a separate validation set was used that was excluded from the learning process. With each new example in the stream, the neural network was incrementally modified by performing 500 learning epochs with that single example.

3.3.3. Evaluation of model accuracy

When evaluating performance of a model on a data stream, our goal is to follow evolution of its performance during the whole data stream and not only compute an aggregated error metric at the end (such as the mean squared error). This is important because the intermediate increasing of error metric might indicate the concept drift in data. A more flexible metric therefore has to give more weight to errors of the recent examples than to errors of the less recent ones. Metrics such as the mean squared error clearly do not satisfy this requirement because of their inability to "forget" old components; when they are updated with new error residual, their sum of errors is divided by the increasing number of examples which makes these metrics converge on the long run.

A possible approach to measuring how model accuracy evolves over time is to use the *alpha*-fading mean squared error (αMSE) which is updated recursively and focuses on model behavior on the most recent data (Rodrigues et al., 2010). The αMSE is defined as

$$s_i = (K_i - C_i)^2 + \alpha \cdot s_{i-1}$$
$$n_i = 1 + \alpha \cdot n_{i-1}$$
$$\alpha\text{MSE}_i = \frac{s_i}{n_i} \tag{3}$$

where s_i denotes recursive error sum (K_i and C_i denote the prediction and the true value of the i-th example, respectively), n_i denotes the recursively incremented counter and α denotes some weight factor which is a parameter of the process. The weight α determines how much historical values contribute when recursively incrementing the statistic.

Since the recursive definition of *alpha*-fading windows is an approximation of the common weighted sliding windows, where examples' weights decrease as the examples progress through the window, the error of this approximation should be controlled. Rodrigues et al. (2010) have shown that the *alpha*-fading approximation of the error can be within $\pm 2\varepsilon R$ of the error computed using weighted sliding windows, if $\alpha = \varepsilon^{1/w}$. In the former terms, ε is the proportion of weight given to old examples (a parameter to be chosen), R is the known range of the approximated variable, and w is the size of the used window. Considering the former, we used $\alpha = 0.01^{1/\text{buffer\_size}} = 0.01^{1/336} \sim 0.9864$ and $n_0 = 0$ and $s_0 = 1$ as the initial values for the recursive formula.

To compare performance of two models M_1 and M_2 on a stream, a Q statistic can be used (Gama et al., 2013), which is defined as a log ratio of αMSE statistics for M_1 and M_2:

$$Q_i(M_1, M_2) = \log\left(\frac{\alpha MSE_i^{M_1}}{\alpha MSE_i^{M_2}}\right) = \log\left(\frac{(K_i^{M_1} - C_i)^2 + \alpha \cdot s_{i-1}^{M_1}}{(K_i^{M_2} - C_i)^2 + \alpha \cdot s_{i-1}^{M_2}}\right) \tag{4}$$

where the terms M_1 and M_2 refer to the corresponding models. Note that just like the αMSE provides performance of a single model over the stream, the Q statistic provides the performance comparison of two models along the stream evolution. Additionally, the values of $Q(M_1, M_2)$ are domain independent and are easy to interpret: negative values denote better performance of model M_1 and positive values denote better performance of model M_2.

3.3.4. Results and discussion

As explained in Section 3.2, we computed the initial prediction K_u and two corrected predictions $K_u + f_C(CNK_u)$ and $K_u + f_S(SAbias_u)$ for each example in the data stream, which simulates outputs of three models: let us denote them with M, M_C and M_S, respectively. Fig. 6 (top) shows a 3-week segment of the data stream, displaying true load values (solid line) and their predictions (dotted line).

Fig. 6 (middle) zooms in to a particular segment of the stream in which we can observe true values, predicted load values and corrected predicted values with estimators *CNK* and *SAbias* in greater detail. We can notice that all three predictions (initial and two corrected ones) closely follow true load values; the achieved average accuracy for all 11 regions of the state of New York was

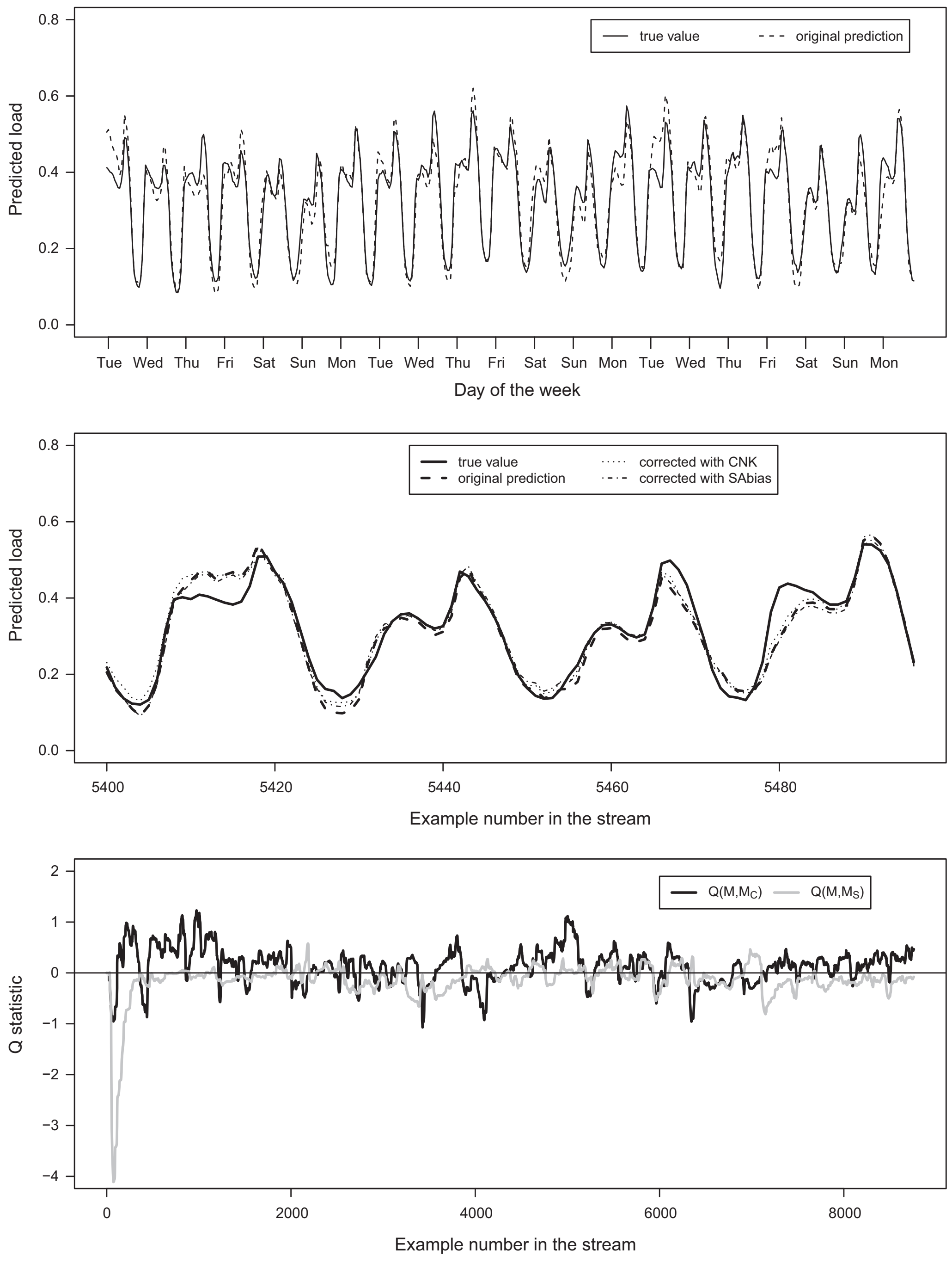

Fig. 6. *Top*: A sample of the true and predicted electricity load for the period of 3 weeks using the neural network (only a segment of the whole data stream is shown). The predictions reflect the daily and weekly cycles in the predicted load. *Middle:* Comparison of the initial and both corrected predictions on a particular data segment. *Bottom:* Values of *Q* statistics for comparing original predictions with corrected predictions using *CNK* and *SAbias*.

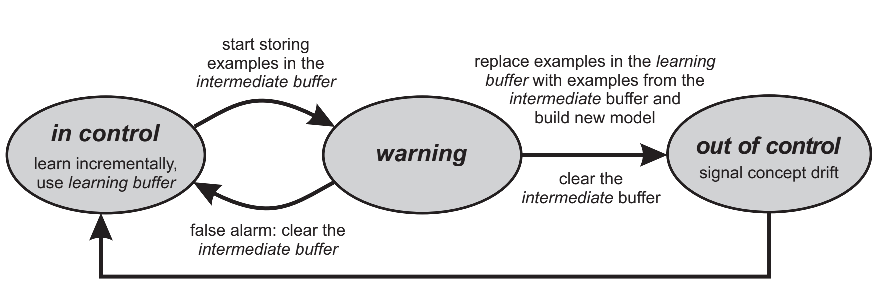
in control
learn incrementally, use *learning buffer*
start storing examples in the *intermediate buffer*
warning
false alarm: clear the *intermediate buffer*
replace examples in the *learning buffer* with examples from the *intermediate* buffer and build new model
clear the *intermediate* buffer
out of control
signal concept drift

Fig. 7. States and transitions of the statistical process control (SPC) concept drift detector.

Table 1
Average values of Q-statistics for comparing initial predictions and predictions corrected with reliability estimator *CNK* ($Q(M, M_C)$), and for comparing initial predictions and predictions corrected with reliability estimator *SAbias* ($Q(M, M_S)$).

Region	$Q(M, M_C)_{avg}$	$Q(M, M_S)_{avg}$
WEST	0.130	−0.145
GENESE	0.185	−0.036
CENTRL	−0.014	−0.190
NORTH	0.133	−0.151
MHK VL	0.151	0.276
CAPITL	0.092	−0.185
HUD VL	0.565	0.081
MILLWD	0.163	0.305
DUNWOD	0.060	−0.032
N.Y.C.	0.374	−0.041
LONGIL	0.035	−0.157

$\alpha\text{MSE}_{avg} = 0.006 \pm 0.001$ which is sufficiently low as it is only about 7% percent of the load scale.

To compare the prediction accuracies over the whole stream, we computed the αMSE statistics and compared them with the Q statistic. We compared the performance of M versus M_C and M_S by computing $Q(M, M_C)$ and $Q(M, M_S)$, respectively. The evolution of both Q statistics is shown in Fig. 6 (bottom). The figure gives an overall impression that values of $Q(M, M_C)$ are higher compared with $Q(M, M_S)$ which implies that the correction with estimator *CNK* achieves better performance than the correction with estimator *SAbias*.

Recall that positive values of Q statistics denote better performance of a corrective mechanism compared with initial (not corrected) predictions. To test whether the corrected predictions significantly improve the accuracy of initial predictions, we used the Wilcoxon test to reject the null hypothesis that the means of $Q(M, M_C)$ and $Q(M, M_S)$ are zero with significance level of $p = 0.001$.

Table 1 summarizes our results and shows the average values of $Q(M, M_C)$ and $Q(M, M_S)$ for the whole stream. All values in the table are highly significant (the highest computed p-value of the statistical test was 6×10^{-19}). The results obtained with the estimator *CNK* are surprisingly better than the results obtained with the estimator *SAbias*: while, on the average, the *CNK*-based correction improved accuracy in predictions for 10 out of 11 regions of NY, the *SAbias*-based correction improved accuracy in only 3 out of 11 regions.

The explanation for better performance of *CNK* might lie in greater sensitivity of *SAbias* to changes in the dataset compared to the sensitivity of *CNK*. Namely, by computing reliability estimates using only examples in a buffer instead using the whole dataset (as discussed in Section 3.2) we interfere with the ideas behind the estimators. In particular, for *SAbias* this means that changing more than a single example in the sensitivity analysis step (which happens in each learning step as the buffer is changed) introduces additional variance that *SAbias* cannot properly model as prediction reliability. This does not seem to interfere with the estimator *CNK*, as the locally modeled error seems to be stable even though its source – the nearest neighbors – keep changing.

Although the achieved results on 11 data streams are promising, these initial experiments call for evaluation on other problem domains and with other predictive models. We are, however, constrained with options for alternative predictive models, as the previous experiments (Bosnić and Kononenko, 2008a) have shown that the performance of reliability estimators significantly depends on the used predictive model. For example, *CNK* was shown to work well with regression trees, linear regression, generalized additive model and neural network, whereas poorly with support vector machines, random forests and locally weighted regression. An additional constraint is that we require incremental versions of predictive models, which do not exist for all of the former algorithms.

In the following section, we move from improving prediction accuracy to another relevant tool for streaming predictions – prediction explanation.

4. Explaining incremental models and predictions

Explanation of individual predictions and whole models is a form of data postprocessing. It is used to gain insight – we call it *explanation* – into the decision function of the model and therewith into its reasoning behind making a particular prediction. This transparency of the model may be obscured and it depends on the nature of the predictive model (for example, knowledge in regression trees can be easier interpreted than the knowledge in neural networks). In this work we apply the methodology (Štrumbelj and Kononenko, 2010) that produces the explanation for an arbitrary prediction model. This methodology was developed for batch learning and is model-independent (it works as a wrapper), treating the model as a black box.

In this paper we apply the former methodology to incremental learning. We (1) explore the possibility of using it to explain the changes in the modeled concept that may result due to concept drifts in data and (2) present two novel visualizations of the modeled concept over time. We begin by describing the original approach in batch learning and then describe our adaptation to incremental learning. Finally, we evaluate the approach on the electricity load prediction problem and two additional artificial classification domains, which provide a controlled environment to compare changes in our generated explanations with true concept drift in data.

4.1. Explanation in batch learning

Prediction explanation increases trust into the model's decision and allows interpretation of the modeled concept. Some models (e.g. decision trees, Bayesian networks) are interpretable by their nature and require no post-processing to extract comprehensible representation of this knowledge; many others require model-dependent explanation methods. If the model-dependent methods are used, the representation of knowledge differs across models, making them incomparable; therefore, model-independent explanation methods are more desirable.

The *Interactions-Based Method for Explanation* (Štrumbelj and Kononenko, 2009) with its efficient implementation (Štrumbelj and Kononenko, 2010), which we use in this paper, is such a model-independent method for explanation that considers attribute interactions and therefore successfully tackles the problem of redundant and disjunctive concepts in data. The possible use of this methodology is twofold: for explaining individual predictions (explanation on an instance level) and for explaining the model as a whole.

4.1.1. Contribution of individual attribute values

The explanation (Štrumbelj and Kononenko, 2009) of prediction K_u for instance x_u is defined as a vector of *contributions of individual attribute values* $(\varphi_1, \varphi_2, \ldots, \varphi_m)$ where φ_j is the contribution of the value of the j-th attribute, $j = 1, 2, \ldots, m$. Positive φ_j implies that the attribute value influenced the model to make a higher prediction K_u and negative value implies the contrary. The magnitude $|\varphi_j|$ represents the strength of the former influence.

Let us denote a set of all attributes with A. To compute the contribution φ_j, the contribution ΔS of all 2^m *subsets of attributes* $S \subseteq A$ need to be computed and then divided among the attribute values. The contributions ΔS are defined as differences between the expected prediction considering only attributes in S and the default prediction. The contributions φ_j are finally derived from all $\Delta S, S \subseteq A$ using the game theory – in particular, the Shapley value, which determines how to divide a game payoff among coalitional game players. To alleviate the time complexity of the above algorithm (2^m attribute subsets need to be considered), the original approach uses the efficient approximation algorithm with random sampling from the dataset (Štrumbelj and Kononenko, 2010), as shown in Algorithm 1.

Algorithm 1. Computation of contribution $\varphi_j(x)$ of the attribute A_j for example x.

```
$\varphi_j(x) \leftarrow 0$
take $k$ samples from data set
**for** each $y$ of $k$ samples **do**
    select random subset of attributes $S \subseteq A$
    $x_{include} \leftarrow$ replace values of $(A \backslash S) \backslash A_j$ with corresponding values of $y$
    $x_{exclude} \leftarrow$ replace values of $(A \backslash S) \cup A_j$ with corresponding values of $y$
    $\varphi_j(x) \leftarrow prediction(x_{include}) - prediction(x_{exclude})$
**end for**
$\varphi_j(x) \leftarrow \frac{\varphi_j(x)}{k}$
```

Fig. 8 in Section 4.4.2 provides an example of visualized prediction explanations for one of our evaluation datasets (discussed later). Two different explanations for the same example (which result due to concept drift) that appear in different parts of data stream motivate us to visualize the changing of contributions over time, which we show in Section 4.4.2.

4.1.2. *Explanation of the whole model*

The explanation of a single prediction can be expanded to the whole model by iterating through all possible values of all attributes. If attribute A_j has l values (if discrete) or l discretized intervals (if continuous), the average contribution of value l for A_j, which we denote with $\Phi_{j,l}$, is computed by averaging contributions of randomly sampled examples that have A_j artificially set to l. The positive and the negative contributions are averaged separately in the former procedure, which yields the average positive and the average negative contribution of each attribute.

Fig. 9 in Section 4.4.2 provides an example of visualized explanation for the whole model for one of our evaluation datasets (discussed later). Two different model explanations for the same dataset (due to concept drift) provide additional motivation to seek for alternative visual representations of changing model over time.

4.2. *Adaptation to incremental learning*

In incremental learning the model has to adapt to possible concept drifts in data. If we wish to explain the model, we have to be aware that the modeled concept depends on time. The *explanation* in incremental learning is therefore in fact represented by a *time series of individual explanations*, which reflect current distribution of data.

In our work we adapt the existing model-independent explanation method for batch learning (see Section 4.1) to generate multiple explanations in incremental learning. While processing the data stream, we trigger the basic method whenever the concept drift detector, which runs in parallel, signals that the concept drift occurred. We expect that the generated explanation series reflect consecutive concepts in streaming data, which provides greater insight into how domain progressively changes. We also propose a form of joint visualization to represent these changes over time.

In the following subsections we describe the used concept drift detector and the approach for triggering computation of new explanation in the series.

4.2.1. *Detecting concept drift*

Data stream contains (possibly infinite) sequence of pairs (x_i, C_i), where x_i is the i-th example and C_i is its target. After the model makes a prediction $K_i = \hat{C}_i$ and the example's true target value becomes available, the model's performance can be assessed. Whenever error rate significantly rises, one can assume that there has been a change in the generating distribution – the concept drift (Sebastião and Gama, 2009). This reasoning represents the basis of the *statistical process control* (*SPC*) mechanism (Gama, 2010), which we use for concept drift detection.

The SPC algorithm maintains two values: the *minimal error rate* q_{min} and the *minimal standard deviation* s_{min}. Error rate q_i of the i-th example is defined as the probability $P(\hat{C}_i \neq C_i)$, and the standard deviation is defined as $s_i = \sqrt{q_i(1-q_i)/i}$. The algorithm updates the two monitored values when $q_i + s_i < q_{min} + s_{min}$. Based on the Bernoulli distribution and central limit theorem, the SPC defines that the processed example can belong to one of the following states:

- *in control*, when $q_j + s_j < q_{min} + \beta * s_{min}$,
- *out of control* when $q_j + s_j \geq q_{min} + \alpha * s_{min}$ and
- *warning* when the system is between the former states.

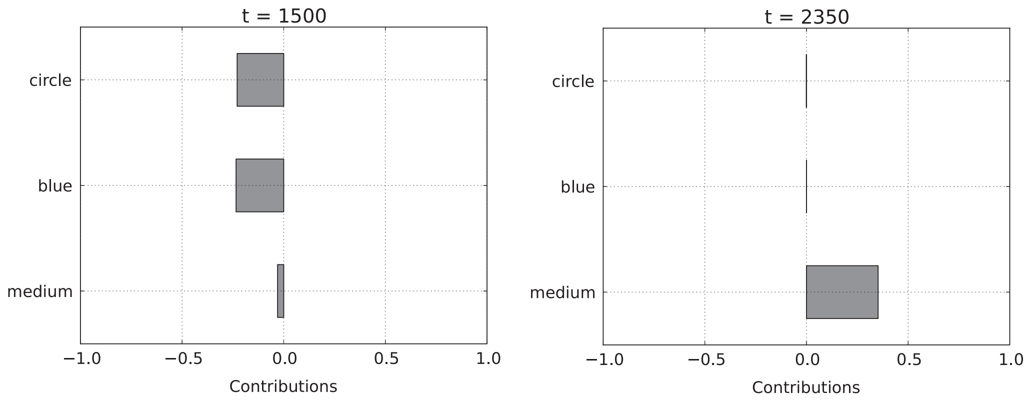

Fig. 8. Explanation of individual predictions for the same example (*shape* = *circle, color* = *blue, size* = *medium*) that appears in a stream as example no. 1500 and 2350 (dataset: STAGGER, model: Naive Bayes). The figure on the left shows that the shape and the color have negative contributions (i.e. they have the biggest influence to classify the example as negative) for classifying examples into the target concept $(color = green) \vee (shape = square)$ whereas the size is a redundant attribute. The figure on the right shows that only the size positively contributes to classifying into the target concept $(size = medium) \vee (size = large)$, whereas the remaining two attributes are redundant.

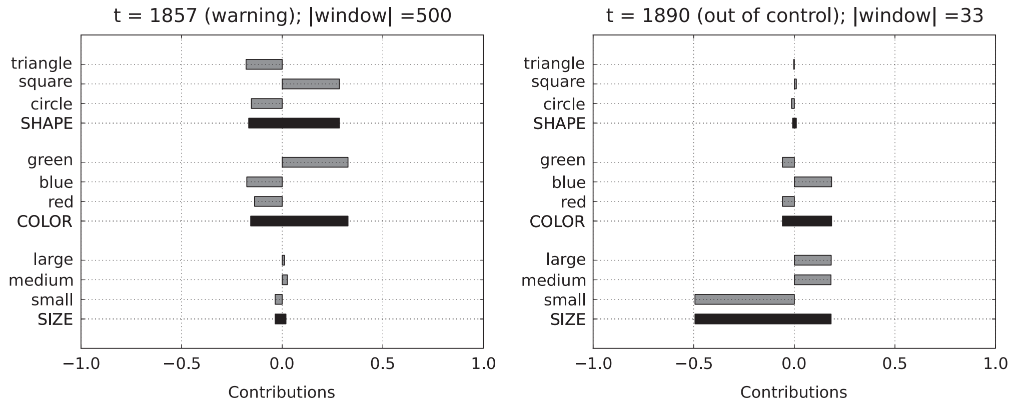

Fig. 9. Model explanations for target concept $(color = green) \vee (shape = square)$ (dataset: STAGGER, model: Naive Bayes) when the system enters the *warning* (left) and the *out of control* state (right). The figures show contributions of individual attribute values (shown as grey horizontal bars) as well as the average positive and negative contributions for the whole attributes (shown as black horizontal bars). The left explanation shows that square shape and the green color contribute to positive prediction into the final class; other shape and color attributes contribute negatively and size is redundant). The explanation on the right was generated when the system indicated concept drift and indicates gradual transition into the target concept $(size = medium) \vee (size = large)$.

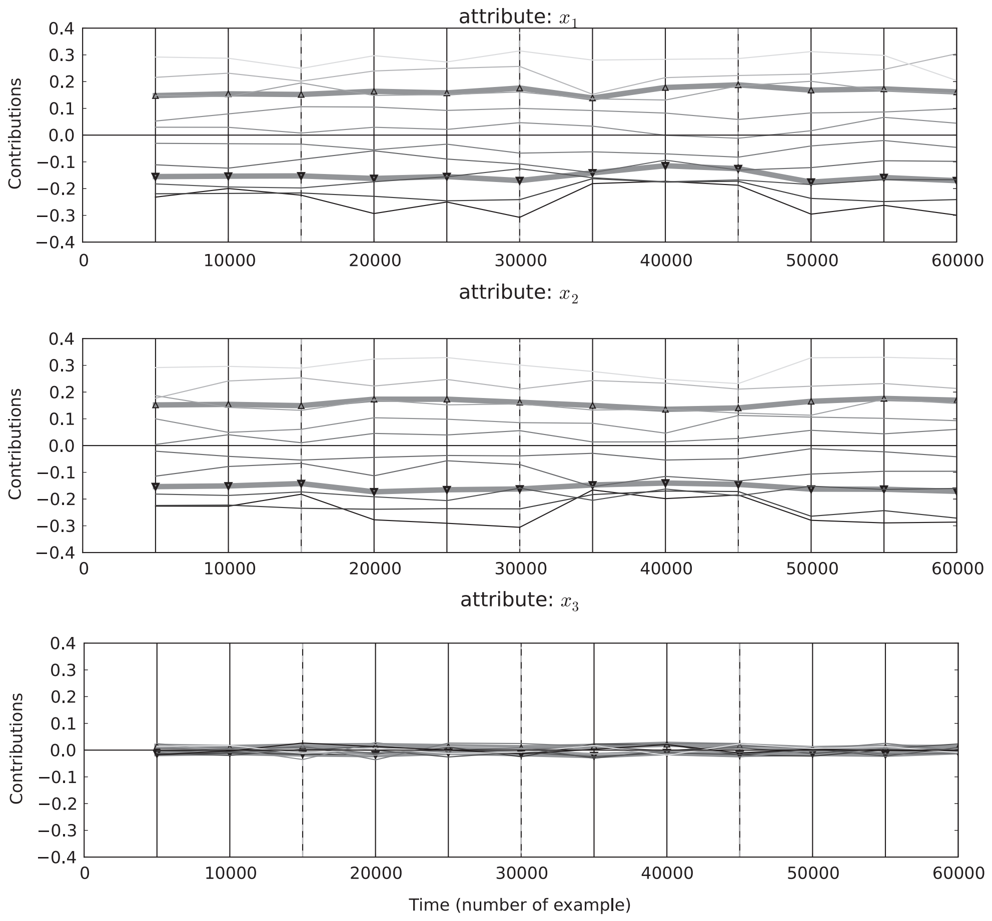

Fig. 10. Changing contributions of attribute values over time (dataset: *SEA*, explanation triggered periodically with $\omega = 15,000$). Axis x denotes time (example numbers) and axis y denotes contribution values. Each graph displays contributions for one attribute in the dataset. Contributions of *individual attribute values* are represented with thin lines; their shadings denote different attributes values – the higher the attribute value the darker the color (attribute values were equidistantly discretized into 10 intervals). Two thick lines in each graph denote the mean positive and negative contributions of the *attribute as the whole*. The contributions of attributes x_1 and x_2 reflect the target concept $x_1 + x_2 \le \beta$; lower values increase the likelihood of positive classification and vice versa. Attribute x_3 is correctly explained as irrelevant with its only contributions being the result of noise.

where α and β are multiples of standard deviation in a symmetric Gaussian distribution associated with the desired confidence levels for *out of control* and *in control* states, respectively.[3]

[3] We compute them using the inverse *erf* function, e.g. $\alpha = \sqrt{2} erf^{-1}(P)$, where P is the desired confidence interval.

The meaning of transitions between different SPC states is illustrated in Fig. 7 and explained as follows. When *in control*, the current model is incrementally updated with the current example, since the error rate is stable. If the system goes into the *warning* state, the subsequent examples are stored in a separate *intermediate* buffer, which serves as a learning set for the new model in case that the concept drift occurs. From the *warning* state, the system

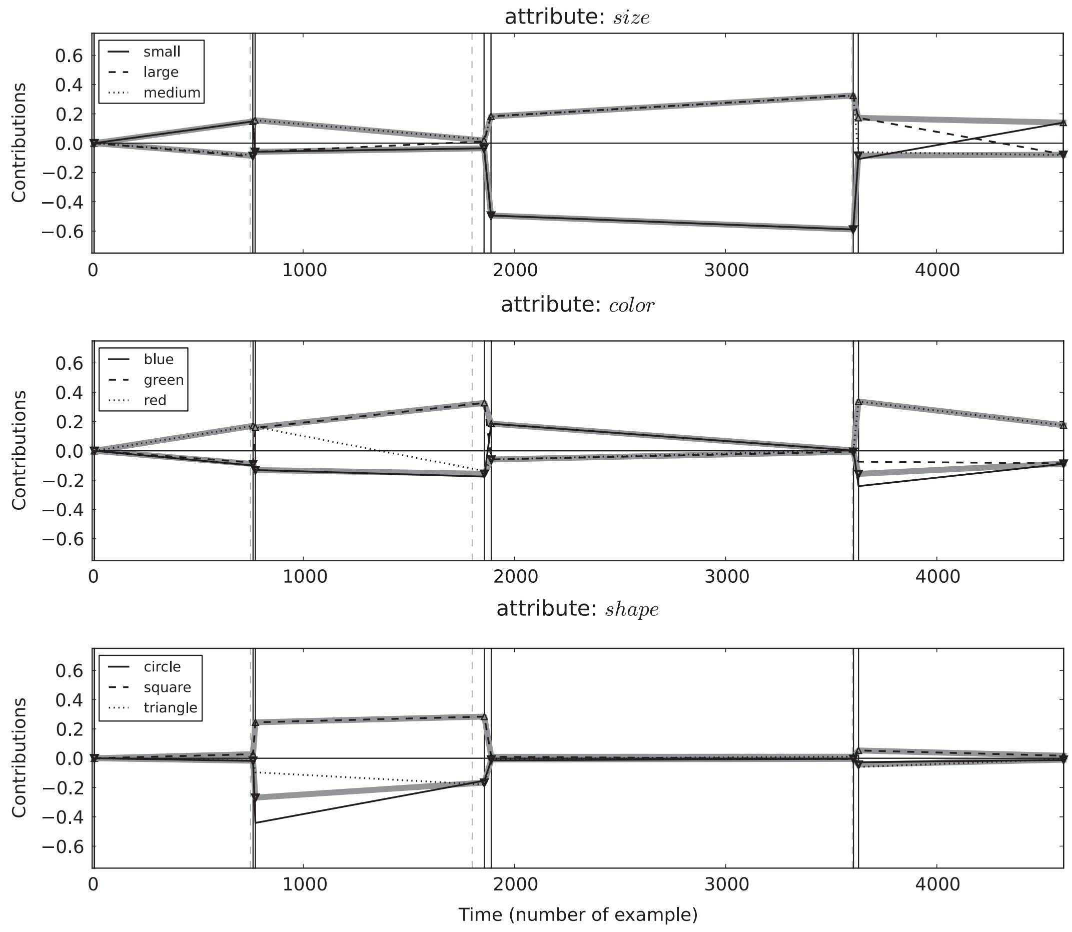

Fig. 11. Changing contributions of attribute values over time (dataset: *STAGGER*, explanation triggered only at concept drift detection). Axis *x* denotes time (example numbers) and axis *y* denotes contribution values. Each graph displays contributions for one attribute in the dataset. Contributions of *individual attribute values* are represented with thin lines; their line types denote attributes values according to respective legends. Two thick lines in each graph denote the mean positive and negative contributions of the *attribute as the whole*. Solid vertical lines indicate times where explanation of the model was triggered using the SEA algorithm and dashed vertical lines mark the places of actual concept drift in data. The figure shows how the SPC algorithm correctly detected concept drifts, as the change in the explanation follows the change in concept. The model successfully adapted to the new concept, which is shown by changing attribute contributions.

can either return to *in control* or progress to the *out of control* state. In the first case, the intermediate buffer is emptied because we deemed the error rise to be a false alarm; in the second case, the new model is constructed from the examples in the buffer and the values q_{min} and s_{min} are reset. The number of examples that were collected in the intermediate buffer between the *warning* and *out of control* state serves as an indicator of the concept drift rate: the smaller is the number of examples in the buffer, the faster the drift occurs.

Like the explanation methodology, the SPC algorithm works as a wrapper in combination with an arbitrary learning algorithm, as well. It is mostly used in batch learning and requires an adaptation to the incremental learning, which we describe in the following.

4.2.2. Adaptation of SPC

Due to the storage limits in incremental learning, we have to bound the upper size of two used buffers: (1) the *learning buffer* – the buffer that stores the most recently arrived examples, and (2) the *intermediate buffer* – the buffer used in the *warning* state of SPC. We perform this by introducing a parameter *max_window*, which determines the maximum number of examples that each of two buffers can store. If the number of examples in each buffer exceeds this threshold, the oldest examples are discarded so that only the newest *max_window* examples are kept. This sliding window decreases the time to process an example and prevents hogging of resources in case where the concept drift does not occur for the longer periods of time. It also cuts down on explanation time without a significant loss of quality, since the distribution of examples in the sliding windows is locally static. Its downside is a partial sacrifice of generality – the parameter *max_window* has to be chosen appropriately, which requires prior knowledge.

4.3. Data stream explanation triggered by concept drifts

In our work we combine the adapted explanation algorithm (see Section 4.1) and the adapted SPC algorithm (see Section 4.2.2). Our approach triggers the computation of model explanation depending on the change of the SPC state, as follows:

- If the system's state changes from *in control* to *warning* state or from *warning* to *in control* state, we trigger computation for explaining the new model. We consider the former change to indicate local peculiarities in data distribution and the latter a more substantial concept drift – both changes are interesting to be detected and visualized.
- If the state changes from *warning* to *in control*, we refrain from triggering the explanation and consider this as a false alarm.

In our system we introduce a parameter ω which determines the minimum frequency for computing model explanation. By computing the explanation at least for every ω examples, we ensure the minimal explanation frequency of the stream. Each local model explanation ϕ consists of computed explanations $\Phi_{j,l}$ for all discrete values (if discrete attribute) or discretized intervals (if continuous attribute) l of the j-th attribute. We denote the final stream of explanations as $(\phi, l, t)_i$, where ϕ is the local explanation, l the size of the sliding window, t is the trigger of explanation ($t \in \{warning, out\_of\_control, \omega\}$ and i the enumeration index in the sequence of generated explanations.

4.4. Experimental evaluation

We evaluated the adapted explanation methodology on two datasets using the basic Naive Bayes and k-nearest neighbors classifiers, which are both incremental in their nature, so no adjustments to the algorithms were necessary.

In the following subsections we present the experimental datasets and present a novel technique for visualizing explanation of the stream. Since the results obtained using the Naive Bayes classifiers were very similar to those using k-nearest neighbors, we only report on the former.

4.4.1. Experimental data

We evaluated our approach on the electricity load prediction problem dataset, which was described in Section 3.3.1, and on two additional synthetic classification problems. The reasons for using the additional two synthetic datasets were because they represent a controlled environment with known distribution of examples. In such environment we can observe if the concept drift detector was triggered at the time when the concept within the data actually changed. We are unable to do so for the electricity load prediction

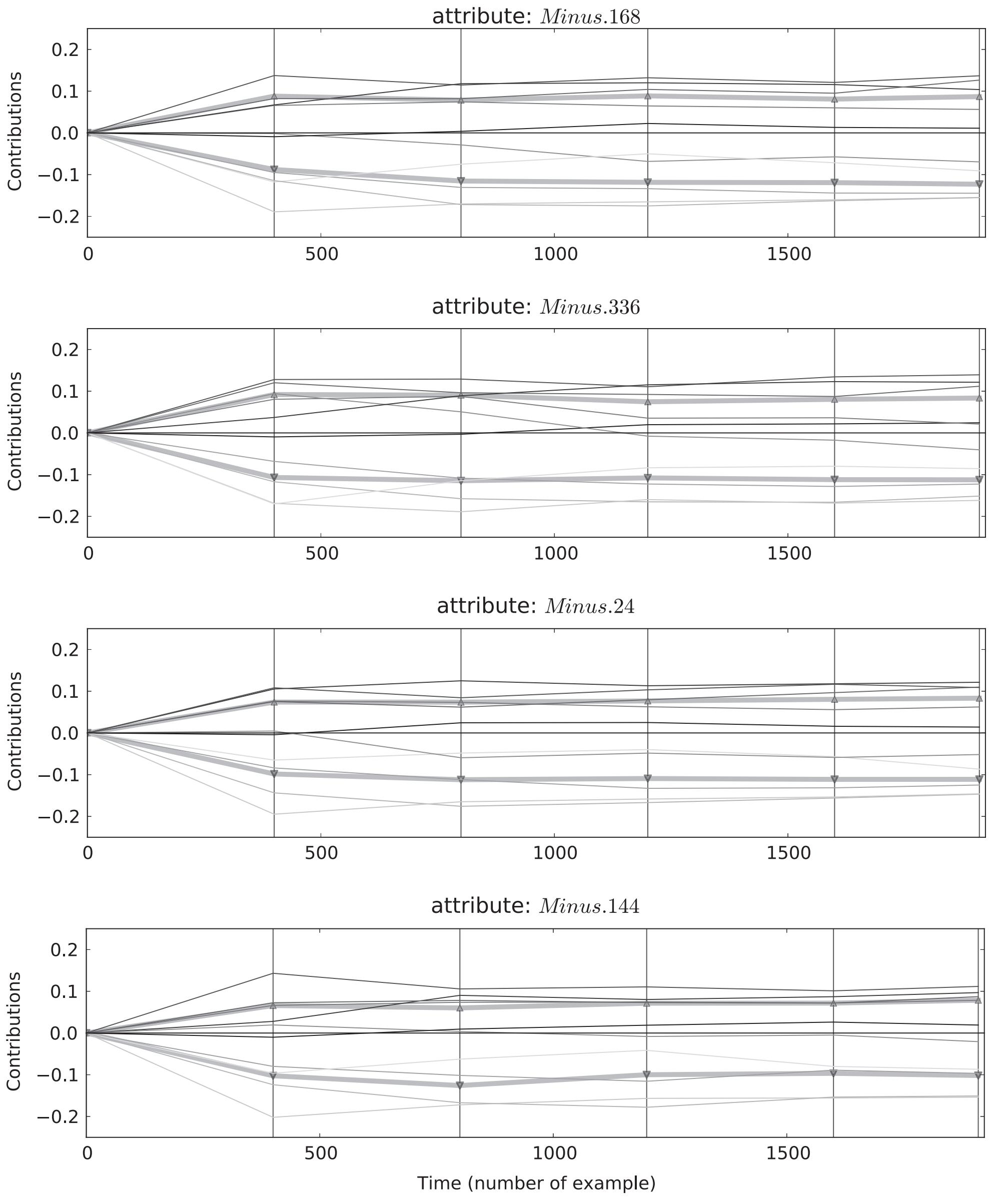

Fig. 12. Contributions of attribute values over time for the *electricity load prediction* dataset (due to space constraints only 4 attributes with the highest contribution are displayed, plots for the others look similar). The explanation was triggered periodically with $\omega = 400$). Axis x denotes time (example numbers) and axis y denotes contribution values. Contributions of *individual attribute values* are represented with thin lines; their shadings denote different attributes values – the higher the attribute value the darker the color (attribute values were equidistantly discretized into 10 intervals). Two thick lines in each graph denote the mean positive and negative contributions of the *attribute as the whole*. The figure shows that the SPC algorithm detected no concept drift and that the contribution of all four attributes remained steady through the stream.

problem, for which there are no known changes in the data distribution. Additionally, the SPC algorithm, as defined in Section 4.2.1, requires probabilities for predicted classes, which are not explicitly provided along with regression predictions. For our experiments, the electricity load target values were hence discretized into two classes: *low load* ($0 \le load \le 0.5$) and *high load* ($0.5 < load \le 1.0$).

Both synthetic datasets contained multiple concepts with various degrees of drift between them. Their brief description:

- *SEA concepts* (Street and Kim, 2001) is a data stream comprising of 60,000 examples (i.e. points in three–dimensional space) with continuous numeric attributes $x_i \in [0, 10]$, $i = 1, 2, 3$ and a binary target concept which equals $x_1 + x_2 \le \beta$ (for some threshold $\beta \in \{7, 8, 9, 9.5\}$). Therefore, only x_1 and x_2 are the relevant attributes. The target concept in the stream is changed by varying threshold β over time to equal $\beta = 8$ first, then $\beta = 9$, then $\beta = 7$ and finally $\beta = 9.5$. In addition to varying the target concept, 10% noise is added to target threshold β within each example.
- *STAGGER* is generated with *MOA* (*Massive Online Analysis*) data mining software (Bifet et al., 2010). The examples represent geometrical shapes which are described by attributes $size \in \{small, medium, large\}$, $color \in \{red, blue, green\}$ and $shape \in \{circle, square, triangle\}$. The boolean target attribute is varied over generated 4500 examples, which are divided into three sequential blocks as follows:
  1. examples 1–750 with target $(size = small) \wedge (color = red)$
  2. examples 751–1800 with target $(color = green) \vee (shape = square)$

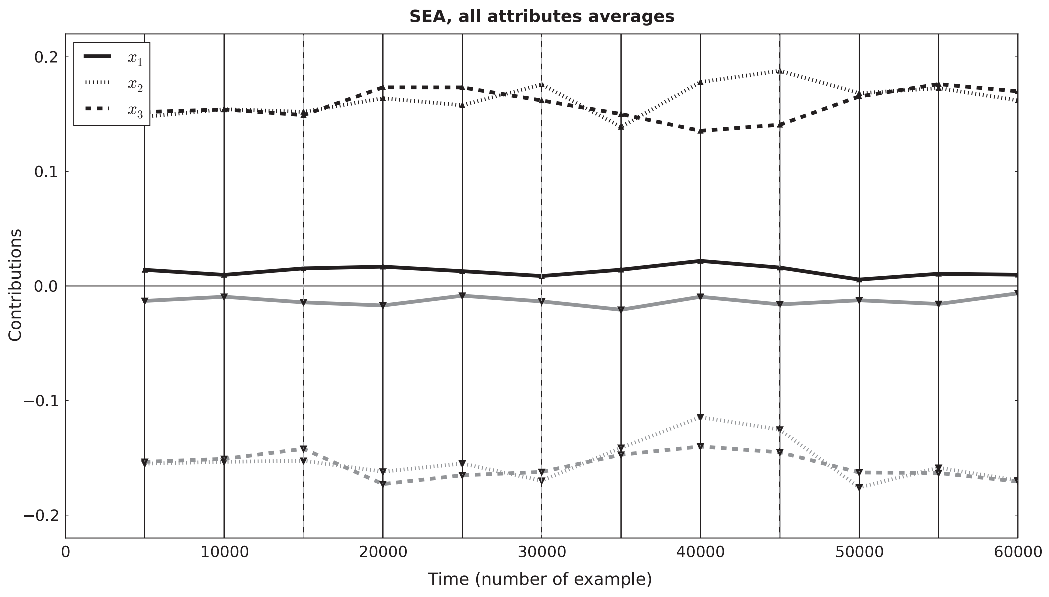

Fig. 13. The mean positive and the mean negative contributions of all attributes (data: *SEA*). The figure shows that attributes x_1 and x_2 are approximately equally important through time.

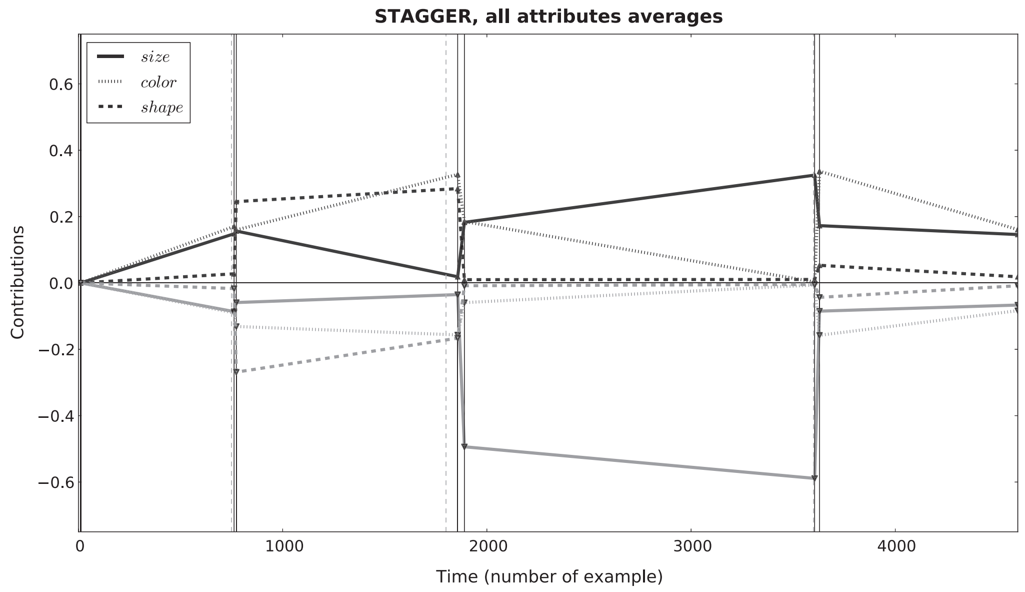

Fig. 14. The mean positive and the mean negative contributions of all attributes (data: *STAGGER*). The figure shows how the importance of attributes changes for predicting each of sequential concepts.

3. examples 1801–3600 with target $(size = medium) \vee (size = large)$
4. examples 3601–4500 with the same target as 1 above.

In this dataset, we applied the gradual change in concept drift between the four blocks by mixing them using a sigmoid function. The width of the gradual drift lasted 50 examples for the changes between concepts $1 \rightarrow 2$ and $2 \rightarrow 3$, and 150 examples for the change between concepts $3 \rightarrow 4$.

4.4.2. *Visualizing explanation in data streams*

The original explanation for individual examples (Štrumbelj and Kononenko, 2010) visualizes contributions of individual attribute values with bar charts – one for each example of which prediction needs explaining. Two such visualizations are given in Fig. 8, which contain explanations for the same example (appearing as examples 1500 and 2350) computed using the model that evolved over the stream and targeting different concepts (see Section 4.4.1).

Different explanations in Fig. 8 suggest that it would make sense to visualize how the knowledge in the model changes as the examples arrive. In batch learning this is done by plotting (1) the mean positive and the mean negative contribution of each attribute value and (2) the mean of each attribute as a whole. We consider this approach less appropriate for incremental models that are subject to concept drifts, because it would require a separate visualization for each local explanation. Two such local model explanations are shown in Fig. 9. Both explanations are generated for the second target concept in the STAGGER dataset and show how the knowledge in the model changes when the SPC algorithm indicates the *warning* and the *out of control* state.

The large number of model explanations hinders perceiving a global view of the whole stream. To include the temporal component, we propose two variations of a line plot, which display changing attribute contributions over time. The first type of visualization is shown in Figs. 10 (for the SEA dataset), 11 (for the STAGGER dataset) and 12 (for the electricity load prediction dataset). Two interesting conclusions can be drawn from the figures:

1. The figures clearly show how contributions of attribute values evolve. Fig. 10 shows that only the last two attributes are relevant for predicting the target goal (which is $x_1 + x_2 \leq \beta$ for $\beta \in \{7, 8, 9, 9.5\}$); the figure also shows that lower attribute values (denoted with lighter shades of thin lines) contribute more to positive prediction outcomes. This makes sense, because by summing lower values of x_1 and x_2 we are less likely to exceed the threshold β and produce a negative example. Fig. 11 illustrates changing contributions even more as they adapt to target concept that changes more radically over time: for example, note how the contribution of attribute *size* increases as the model adapts to the first and the third concept (in which *size* is relevant) and decreases as the model adapts to the second concept (in which *size* is irrelevant). The last, Fig. 12 shows how the contributions of all four observed attributes are relevant for predicting the target.
2. Solid and dashed vertical lines in Fig. 11, which denote SPC-triggered computation of explanation and the real point of concept drift, respectively, coincide well. The dashed vertical line is followed by two solid lines that correspond to transitions to the *warning* and *out of control* state. This indicates the appropriateness of the SPC algorithm for our task. This is less obvious in Fig. 10 in which all concept drifts, which were successfully detected by SPC, and the periodically triggered explanations do not differ greatly.

The second type of visualization, which we propose, is shown in Figs. 13–15. These figures are an aggregated versions of Figs. 10, 11 and 15 and contain the mean positive and the mean negative contributions of all attributes in one place. Such figures condense information from the former three figures to provide a quality insight into the model and enable comparison of contributions on the same scale. Fig. 13 shows how both attributes, x_1 and x_2, are approximately equally important through time, as well as all observed attributes in Fig. 15. Fig. 14, however, allows quick identification of important attributes for each sequential concept.

5. Conclusion

In our work we focused on two tasks: correcting predictions in data streams and explaining incremental models. For both tasks we adapted the existing methodologies that were originally developed for batch learning to specifics of the incremental learning environment.

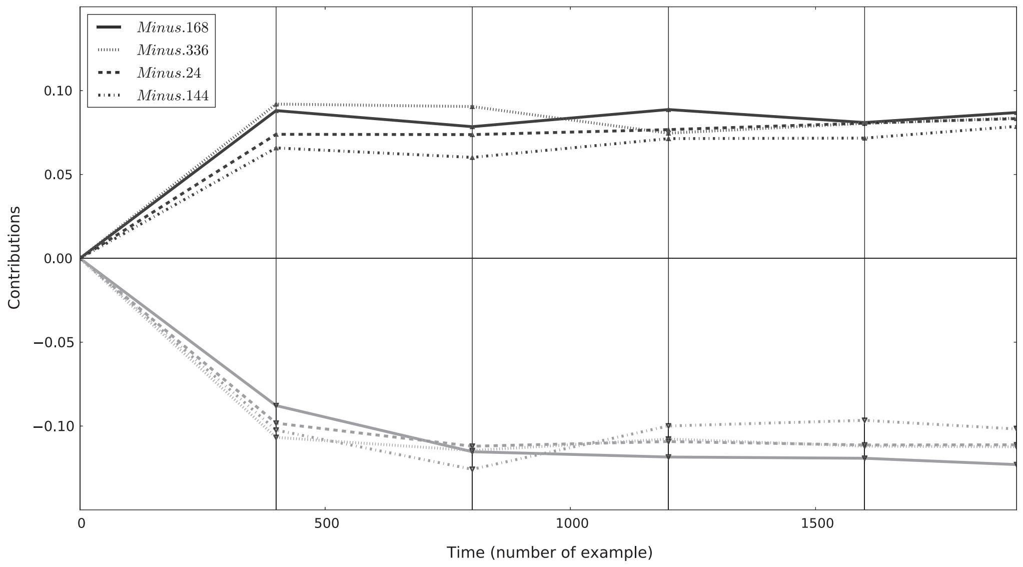

Fig. 15. The mean positive and the mean negative contributions of all attributes (data: *electricity load prediction*). The figure shows that the observed attributes are approximately equally important through time.

The application of reliability estimators *CNK* and *SAbias* to correct predictions has shown that unavailability of entire data interferes with proper sensitivity analysis and consequently the estimator *SAbias*. Since the locality based estimator *CNK* modeled reliability by using only the most similar examples in the neighborhood, it overcame this limitation and yielded promising results by improving accuracy in 10 out of 11 testing data sets. As our second task we intertwined the explanation methodology for models and predictions in batch learning with a concept drift detector. We illustrated how the evolution of attribute contributions can be followed through time and be used to explain the changing concept in the model.

In the present work, we applied such learning algorithms and parameters as suggested in related literature. Although we are limited to incremental implementations of predictive algorithms, we are aware that both approaches should be evaluated with different models and with different problem domains to ensure general usefulness. An important issue to note is the high time complexity of used reliability estimators, which is acceptable if the prediction problem demands low prediction frequency. We plan to focus on broader evaluation in our further work, as well as on comparing prediction correction mechanisms with other approaches in related work.

To conclude, the results indicated that the proposed approaches can improve prediction accuracy and allow transparent understanding of the modeled concepts. We believe that both evaluated methodologies provide useful tools for users of incremental models, as they improve their applicability in risk-sensitive environments where prediction mistakes can be costly – such as in medicine, finance and industry.

References

Andrews, R., Diederich, J., Tickle, A.B., 1995. Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl.-Based Syst. 8, 373–389.

Becker, B., Kohavi, R., Sommerfield, D., 1997. Visualizing the simple Bayesian classier. In: KDD Workshop on Issues in the Integration of Data Mining and Data Visualization.

Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B., 2010. Moa: Massive online analysis. J. Mach. Learn. Res. 11, 1601–1604.

Bosnić, Z., Kononenko, I., 2008a. Comparison of approaches for estimating reliability of individual regression prediction. Data Knowl. Eng. 67, 504–516.

Bosnić, Z., Kononenko, I., 2008b. Estimation of individual prediction reliability using the local sensitivity analysis. Appl. Intell. 29, 187–203.

Bosnić, Z., Kononenko, I., 2009. An overview of advances in reliability estimation of individual predictions in machine learning. Intell. Data Anal. 13, 385–401.

Bosnić, Z., Vračar, P., Radović, M., Devedžić, G., Filipović, N., Kononenko, I., 2012. Mining data from hemodynamic simulations for generating prediction and explanation models. IEEE Trans. Inf. Technol. Biomed.: A Publ. IEEE Eng. Med. Biol. Soc. 2, 248–254.

Bousquet, O., Elisseeff, A., 2002. Stability and generalization. J. Mach. Learn. Res. 2, 499–526.

Breiman, L., 2001. Random forests. Mach. Learn. J. 45, 5–32.

Bunn, D., Farmer, E., 1985. Comparative Models for Electrical Load Forecasting. Wiley, New York.

Carney, J., Cunningham, P., 1999. Confidence and prediction intervals for neural network ensembles. In: Proceedings of IJCNN'99, The International Joint Conference on Neural Networks, Washington, USA, pp. 1215–1218.

Craven, M.W., Shavlik, J., 1994. Using sampling and queries to extract rules from trained neural networks. In: Proceedings of International Conference on Machine Learning, pp. 37–45.

Fox, D.J., 2002. An R and S Plus Companion to Applied Regression, 1st edition Sage Publications, Inc.

Gama, J., 2010. Knowledge Discovery from Data Streams, 1st edition Chapman & Hall/CRC, chapter 3, pp. 42–47.

Gama, J., Rodrigues, P., 2007. Learning from Data Streams – Processing Techniques in Sensor Networks. Springer Verlag, chapter 3, pp. 25–39.

Gama, J., Sebastião, R., Rodrigues, P.P., 2013. On evaluating stream learning algorithms. Mach. Learn. 90, 317–346.

Gammerman, A., Vovk, V., Vapnik, V., 1998. Learning by transduction. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin, pp. 148–155.

Hamel, L., 2006. Visualization of support vector machines with unsupervised learning. In: Proceedings of 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, Toronto, Canada, pp. 1–8.

Heskes, T., 1997. Practical confidence and prediction intervals. In: Mozer, M.C., Jordan, M.I., Petsche, T. (Eds.), Advances in Neural Information Processing Systems. The MIT Press, pp. 176–182.

Hulten, G., Spencer, L., Domingos, P., 2001. Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106.

Jakulin, A., Možina, M., Demšar, J., Bratko, I., Zupan, B., 2005. Nomograms for visualizing support vector machines. In: KDD '05: Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, New York, NY, USA. pp. 108–117.

Kononenko, I., 1993. Inductive and bayesian learning in medical diagnosis. Appl. Artif. Intell. 7, 317–337.

Kuhn, M., 2012. Variable Selection Using the Caret Package. URL ⟨http://cran.cermin.lipi.go.id/web/packages/caret/vignettes/caretSelection.pdf⟩.

Kyriakides, E., Polycarpou, M., 2007. Short term electric load forecasting: a tutorial. Trends Neural Comput., 391–418.

Lemaire, V., Féraud, R., Voisine, N., 2008. Contact personalization using a score understanding method. In: International Joint Conference on Neural Networks (IJCNN).

Li, F., Wechsler, H., 2005. Open set face recognition using transduction. IEEE Trans. Pattern Anal. Mach. Intell. 27, 1686–1697.

Možina, M., Demšar, J., Kattan, M., Zupan, B., 2004. Nomograms for visualization of naive bayesian classifier. In: PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag New York, Inc., New York, NY, USA, pp. 337–348.

Nouretdinov, I., Melluish, T., Vovk, V., 2001. Ridge regression confidence machine. In: Proceedings of 18th International Conference on Machine Learning. Morgan Kaufmann, San Francisco, CA, pp. 385–392.

NYISO, 2012. New York Independent System Operator. Load Data. URL ⟨http://www.nyiso.com/public/markets_operations/market_data/load_data/index.jsp⟩.

Poulet, F., 2004. SVM and graphical algorithms: a cooperative approach. In: Proceedings of Fourth IEEE International Conference on Data Mining, pp. 499–502.

Robnik-Šikonja, M., Kononenko, I., 2008. Explaining classifications for individual instances. IEEE Trans. Knowl. Data Eng. 20, 589–600.

Rodrigues, P., Bosnić, Z., Gama, J., 2008. Online reliability estimates for individual predictions in data streams. In: Bonchi, F. (Ed.), ICDM Workshops 2008: Proceedings, Pisa, Italy. pp. 36–45.

Rodrigues, P., Gama, J., 2009. A system for analysis and prediction of electricity load streams. Intell. Data Anal. 13, 477–496.

Rodrigues, P., Gama, J., Sebastiao, R., 2010. Memoryless fading windows in ubiquitous settings. In: Proceedings of Ubiquitous Data Mining (UDM) Workshop in Conjunction with the 19th ECAI 2010, Lisbon, pp. 27–32.

de Santana, A.L., Frances, C., Rocha, C.A., Carvalho, S.V., Vijaykumar, N.L., Rego, L.P., Costa, J.C., 2007. Strategies for improving the modeling and interpretability of bayesian networks. Data Knowl. Eng. 63, 91–107.

Saunders, C., Gammerman, A., Vovk, V., 1999. Transduction with confidence and credibility. In: Proceedings of IJCAI'99, pp. 722–726.

Sebastião, R., Gama, J., 2009. A study on change detection methods. In: Lopes, L.S., Lau, N., Mariano, P., Rocha, L.M. (Eds.), Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA 2009, Aveiro, Portugal, October 12–15, 2009. Proceedings, Springer, pp. 353–264.

Street, W.N., Kim, Y., 2001. A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 377–382.

Towell, G., Shavlik, J.W., 1993. Extracting refined rules from knowledge-based neural networks, machine learning. Mach. Learn. 13, 71–101.

Štrumbelj, E., Bosnić, Z., Kononenko, I., Zakotnik, B., Grašič-Kuhar, C., 2010. Explanation and reliability of prediction models: the case of breast cancer recurrence. Knowl. Inf. Syst. 9, 305–324.

Štrumbelj, E., Kononenko, I., 2010. An efficient explanation of individual classifications using game theory. J. Mach. Learn. Res. 11, 1–18.

Štrumbelj, E., Kononenko, I., 2009. Robnik Šikonja, 2009. Explaining instance classifications with interactions of subsets of feature values. Data Knowl. Eng. 68, 886–904.

Weigend, A., Nix, D., 1994. Predictions with confidence intervals (local error bars). In: Proceedings of the International Conference on Neural Information Processing (ICONIP'94). Seoul, Korea, pp. 847–852.