On-line learning from streaming data with delayed attributes: a comparison of classifiers and strategies

Mónica Millán-Giraldo · J. Salvador Sánchez · V. Javier Traver

Received: 5 May 2010 / Accepted: 3 June 2010 / Published online: 17 June 2010 © Springer-Verlag London Limited 2010

Abstract In many real applications, data are not all available at the same time, or it is not affordable to process them all in a batch process, but rather, instances arrive sequentially in a stream. The scenario of streaming data introduces new challenges to the machine learning community, since difficult decisions have to be made. The problem addressed in this paper is that of classifying incoming instances for which one attribute arrives only after a given delay. In this formulation, many open issues arise, such as how to classify the incomplete instance, whether to wait for the delayed attribute before performing any classification, or when and how to update a reference set. Three different strategies are proposed which address these issues differently. Orthogonally to these strategies, three classifiers of different characteristics are used. Keeping on-line learning strategies independent of the classifiers facilitates system design and contrasts with the common alternative of carefully crafting an ad hoc classifier. To assess how good learning is under these different strategies and classifiers, they are compared using learning curves and final classification errors for fifteen data sets. Results indicate that learning in this stringent context of streaming data and delayed attributes can successfully take place even with simple on-line strategies. Furthermore, active strategies behave generally better than more conservative passive ones. Regarding the classifiers, it was found that simple instance-based classifiers such as the well-known nearest neighbor may outperform more elaborate classifiers such as the support vector machines, especially if some measure of classification confidence is considered in the process.

Keywords Streaming data · On-line classification · Delayed attributes · Semi-supervised learning

1 Introduction

Most of traditional learning algorithms assume the availability of a training set of labeled objects (examples or instances) in memory. In recent years, however, advances in information technology have lead to a variety of applications in which huge volumes of data are collected continuously, thus making impossible to store all data, or to process any particular object more than once. Under these circumstances, data are not available as a batch but each instance comes one object at a time (called streaming data). In general, a data stream is defined as a sequence of instances [2, 17]. Data streams differ from the conventional model in important elements [4] that bring new challenges: (1) The objects in the stream arrive on-line; (2) The system has no control over the order in which incoming data arrive to be processed; and (3) Data streams are potentially unbounded in size.

Classification is perhaps the most widely studied problem in the context of data stream mining. Although substantial progress has been made on this topic [1, 8, 21], a number of issues still remain open. For example, many classification models do not make adequate use of the history of data streams in order to accommodate changes in class
distribution (known as concept drift [12, 22, 23, 25]). The scenario we consider in this paper faces a new problem that may appear in several real-world applications. We assume that each object of a data stream is a vector of $d$ attribute values without a class label. The aim of the classification model is to predict the true class of each incoming object as soon as possible (ideally, in real time). However, suppose that the attribute values are obtained from different sensors. These may produce that the attribute values will become available at different times if some sensor requires more processing time to compute an attribute value than the others or even, if some sensors fail. Therefore, we are considering the problem of classifying streaming data where one or more attributes arrive with a delay. As an example, when a sensor fails in a production process, it might not be feasible to stop everything and in this case, the system should employ the information available at present time. Three main issues here are: (1) How to classify the incoming sample that contains missing attributes; (2) Whether to update the training (reference) set after predicting the class label of an incomplete object or wait until the attribute vector has been completed; and (3) What to do when the missing attributes arrive at the system.

In the literature, there exist many algorithms for handling data with missing attributes in off-line learning [9, 13, 14, 19, 20], but no one is absolutely better than the others. The most representative categories of these are:

- **Removing incomplete examples**: The simplest way of dealing with missing values is to discard the examples that contain the missing values in order to create a new complete data set. Nevertheless, this technique may lose relevant information.

- **Projection**: The $l$ missing attributes are ignored. This implies to map the $d$ dimensional input vectors onto an $(d - l)$ instance space.

- **Imputation**: It tries to guess the missing values. In fact, usually missing values depend on other values, and if we find a correlation between two attributes, we may use it to impute missing attributes. Imputations may be deterministic or random (stochastic). In the first case, imputations are determined by using the complete data, and are the same if the method is applied again. In the second case, imputations are randomly drawn.

Despite the problem of missing attributes has been widely studied in off-line learning, to the best of our knowledge it has not previously been considered in the context of on-line learning with streaming data, which makes the problem considerably more challenging. Only very recently, the problem of missing attributes in streaming data has been considered [10]. However, interesting, their approach is an ad hoc solution for decision trees, while our focus is on strategies that are orthogonal to the choice of the classification method being used. In addition, they did not consider that the missing attributes may arrive after a time delay. Another promising idea is that of working in the dissimilarity space rather than in the original feature space, which offers advantages under the missing attributes scenario [16].

This paper reports a preliminary study of three straightforward strategies for an early classification of streaming data with missing (delayed) attributes. By *early* we mean that classification of an incoming object is (may be) done before the whole attribute vector is known. Many applications can benefit from performing this early classification, since there may be some kind of loss associated with waiting for the missing attributes to arrive. In the present work, we concentrate on the case of a single missing attribute, which happens to be the same and arrive with a constant delay.

This article extends the conference paper [15] by comparing three different classifiers and providing further experimental insight into how these classifiers and the different strategies considered behave with respect to the baseline reference strategy of only considering an initial reference set, which is labeled and with all attributes, but with a reduced number of instances.

## 2 On-line classification of data with delayed attributes

At time step $t$ in the scenario of attributes arriving with a delay, we have a reference set $S_t$ (a set of labeled examples with all attributes available). Then, a new unlabeled object $x_{t+1}$ with one missing attribute $x_{t+1}^{(l)}$ arrives. After predicting the label for $x_{t+1}$, the system receives the value of the attribute $x_{t+1}^{(l)}$ corresponding to the object that came $\tau$ steps earlier, $x_{t-\tau+1}$. Therefore, objects from $x_{t-\tau+2}$ to $x_{t+1}$ are still with one missing attribute.

Here, one key question is whether to use the unlabeled data with missing attributes to update the reference set $S_t$ and in such a case, how to do it. In addition, we have to decide how to best utilize the value of the missing attribute $x_{t+1}^{(l)}$ when it arrives.

When a new unlabeled object $x_{t+1}$ arrives, the system has to provide a prediction for its label based on the information available up to time $t$. In this situation, it could be desirable to make use of the *confidence* with which the previous classifications have been made. That is why a modification of the $k$-Nearest Neighbors ($k$-NN) rule [24] will be used here and adapted to the problem at hand, since its stochastic nature proves to be suitable to properly manage the confidence measurements. Apart from this, the plain 1-NN classifier and a Support Vector Machine (SVM) will be also considered in the experiments for comparison purposes. On the other hand, for handling the missing attribute of object
\(x_{t+1}\), we will employ the projection strategy because of its simplicity and its proven good behavior.

2.1 A confidence-based classifier

All instances in the reference set have a confidence value for each class, indicating the probability of belonging to the corresponding class. When a new unlabeled object \(x_{t+1}\) from the data stream arrives, its confidence values (one per class) are estimated. Thus the object will be assigned to the class with the highest confidence value.

To estimate the confidence values of the incoming object \(x_{t+1}\), its \(k\) nearest neighbors from the reference set \(S_t\) are used. The confidences of its \(k\) nearest neighbors and the distances between each of them to the new object \(x_{t+1}\) are also employed. More formally, let \(k\) be the number of nearest neighbors, \(n_j\) the \(j\)-th nearest neighbor of \(x_{t+1}\), \(P_m(n_j)\) the confidence (probability) of the \(j\)-th nearest neighbor belonging to class \(m\), and \(d(x_{t+1}, n_j)\) the Euclidean distance between the object \(x_{t+1}\) and \(n_j\). The confidence of the object \(x_{t+1}\) in relation with the class \(m\), say \(P_m(x_{t+1})\), is given by:

\[
P_m(x_{t+1}) = \sum_{j=1}^{k} \frac{P_m(n_j)}{\varepsilon + d(x_{t+1}, n_j)},
\]

where \(\varepsilon\) is a constant value (\(\varepsilon = 1\)), which is employed to avoid numerical problems in the division when the object \(x_{t+1}\) is very similar to its \(j\)-th nearest neighbor.

The aforementioned expression states that the confidence that an object \(x_{t+1}\) belongs to a class \(m\) is the weighted average of the confidences of its \(k\) nearest neighbors belonging to class \(m\). Each of these weights is inversely proportional to the distance from the object to the corresponding nearest neighbor. In order to get a proper probability, the confidence \(P_m(x_{t+1})\) in Eq. (1) is divided by the sum of the confidences of the \(k\) nearest neighbors to all the \(c\) classes:

\[
p_m(x_{t+1}) = \frac{P_m(x_{t+1})}{\sum_{r=1}^{c} P_r(x_{t+1})}.
\]

As the objects of the reference set \(S_r\) are labeled elements, their confidence values were initially set to 1 for the true class (the class to which they belonged), and to 0 for the remaining classes. During the on-line classification, the confidence of all new objects added to the training set will be updated according to the probability values estimated according to Eq. (2).

2.2 Strategies for handling delayed attributes

Assuming that at step \(t\) we have a reference set \(S_t\) available, on-line classification of incomplete data streams consists of three elements: (1) The technique to handle the situation of a missing attribute \(x_{t+1}^{[i]}\) of the new unlabeled object \(x_{t+1}\); (2) The classifier to predict the class label for this object; and (3) The strategy to manage the new information derived from the value of the attribute \(x_{t+1}^{[i]}\) when it arrives \(\tau\) steps later.

Regarding the first issue, as stated before, the projection strategy is used: the arriving object as well as those in the reference set are simply mapped onto the \((d-1)\) dimensional space. Second, as for the prediction of the class label for \(x_{t+1}\), three different classifiers are used: the \(k\)-NN classifier based on posterior probabilities (Sect. 2.1), the plain 1-NN rule and a SVM. Finally, since it is not obvious which is the best way to profit from the new information gained with the arrival of the attribute \(x_{t+1}^{[i]}\) at time step \(t+1\), three different strategies are explored:

- **Do-nothing**: This is a passive strategy where, while the incoming object is incorporated into the current reference set \(S_t\), nothing is done when the value of the missing attribute \(x_{t+1}^{[i]}\) arrives after \(\tau\) time steps. However, the attribute value of the corresponding object, \(x_{t-1+t+1}\), is set to the value \(x_{t+1}^{[i]}\).
- **Put-and-reclassify**: This is a proactive strategy that differs from the do-nothing approach in that the object \(x_{t-1+t+1}\) is also reclassified, but this time using all attributes.
- **Wait-and-classify**: This is a reactive strategy where, unlike the two previous strategies, the new object \(x_{t+1}\) is not included in the reference set \(S_t\) until its missing attribute is received after \(\tau\) time steps. Only by then, the complete object is classified and incorporated into the reference set \(S_{t+1}\).

The different nature of these strategies will allow to gain some insight into which may be the best way to proceed in the context of on-line classification of streaming data with missing (but delayed) attributes. This will also provide cues on what further research avenues to follow. The assessment of the different strategies proposed has been done on extensive experimental work, which is subsequently presented.

3 Experiments and results

The experiments here carried out are aimed at empirically evaluating each strategy described in the previous section, in order to determine which of them might be more suitable for the classification of incomplete streaming data. The ultimate purpose of this study is to investigate whether the use of attribute values that arrive with a delay allows to improve the system performance. Also, these experiments are intended to explore the importance of using a classifier
with some confidence measurement in a scenario of early classification.

3.1 Experimental setup

Experiments were conducted as follows:

**Data sets:** Fifteen real data sets (a summary of which is given in Table 1) were employed in the experimental work. Most of these data sets have been commonly used in previous research papers on data stream classification. Data were normalized in the range [0, 1] and all features were numerical. In the table, the data sets are sorted by increasing size.

**Partitions:** For each database, five runs were carried out. A random stratified sample of \( d \times c \), with \( d \) being the number of attributes and \( c \) the number of classes, was taken as the initial labeled reference set \( S_0 \). The remaining part of each database was used as the incoming on-line streaming data. Besides, since many data sets are relatively small (only a few hundred of instances), new objects were artificially generated with the aim of having a large number \( N \) of instances in the streaming data set \( (N = 3,000 \) was used here) so that on-line learning effects can be studied for a longer time. These new objects were created by introducing random noise in one or more random attributes of the original objects which were themselves randomly selected. The probability of introducing noise to each attribute of each new object was 0.1. The amount of noise added to each attribute was randomly chosen uniformly in \([0, 0.5]\), taking into account that data were already normalized and guaranteeing that the values were not out of the range \([0, 1]\) to keep them normalized. Besides, to simulate independent and identically distributed sequences, the data were shuffled before each of the five runs.

**Incomplete objects:** A new object with one missing attribute from the on-line data was fed to the system at a time step. Both the most and the least relevant attributes of each data set were simulated to be missing. Attribute relevance was estimated by means of the Jeffries-Matusita distance [5].

**Delay:** The value of the missing attribute comes after \( \tau = 5 \) time steps. When the delayed attribute arrives, the corresponding object is completed with the true attribute value.

**Classification:** At each time step \( t \), the respective strategy to handle delayed attributes was applied. The accumulated classification error (the total number of misclassifications divided by the number of samples processed up to \( t \)) was computed. In this way, we created a learning curve (trend line), which is the classification error as a function of the number of on-line objects seen by the classifier. The results were averaged across the five runs giving a single incremental learning curve for each data set.

**Classifiers:** The plain 1-NN rule, the 5-NN classifier with confidence values (see Sect. 2.1) and a SVM were employed in the experiments. For SVM classification, the kernel was a radial basis function (RBF) whose parameters were optimized for each data set. The RBF kernel was chosen because of its generality and reported good results [11]. Parameter optimization was performed using the functionality and guidelines provided within the LIBSVM library [6].

For each of the five runs of the experiment, all strategies received the same partitions of the data into initial labeled

| Data set       | Number of sets | Number of objects | Number of features | Size of initial classes | Database source |
|---------------|----------------|-------------------|-------------------|--------------------------|-----------------|
| iris          | 150            | 4                 | 3                 | 12                       | UCI\(^a\)        |
| crabs         | 200            | 6                 | 2                 | 12                       | Ripley\(^b\)    |
| sonar         | 208            | 60                | 2                 | 120                      | UCI             |
| laryngeal1    | 213            | 16                | 2                 | 32                       | Library\(^c\)   |
| thyroid       | 125            | 5                 | 3                 | 15                       | UCI             |
| intubation    | 302            | 17                | 2                 | 34                       | Library         |
| ecoli         | 336            | 7                 | 8                 | 56                       | UCI             |
| liver         | 345            | 6                 | 2                 | 12                       | UCI             |
| wbc           | 569            | 30                | 2                 | 60                       | UCI             |
| laryngeal2    | 692            | 16                | 2                 | 32                       | Library         |
| pima          | 768            | 8                 | 2                 | 16                       | UCI             |
| vehicle       | 846            | 18                | 4                 | 72                       | UCI             |
| vowel         | 990            | 11                | 10                | 110                      | UCI             |
| german        | 1,000          | 24                | 2                 | 48                       | UCI             |
| image         | 2,310          | 19                | 7                 | 133                      | UCI             |

\(^a\) UCI [3]  
\(^b\) Ripley [18]  
\(^c\) Library [http://www.bangor.ac.uk/~mas00a/activities/real_data.htm]
reference set and streaming data set. These on-line data were presented to all methods in the same order so that performance differences cannot be attributable to different data (order).

3.2 Results

Detailed results of the plain 1-NN classifier on three representative data set are provided in Fig. 1, with the $x$ and $y$ axes representing, respectively, the number of objects fed to the system at each time step, and the accumulated classification error averaged over the five runs for each strategy. For visualization purposes, the learning curves were smoothed, which affects mainly the curves at the first time steps, when the classification error is still unstable. To evaluate whether the performance improves at all with streaming data, we have also included the error using only the initial reference set $S_0$. This strategy is simply referred to as “Reference Set” in figures in this section. Some remarks can be done from these plots:

- The accumulated classification error decreases over time, which is a clear evidence of how the system is learning from the incoming, incomplete, unlabeled samples.
- When the delayed attribute was the most relevant, the strategies Put-and-reclassify and Wait-and-classify can be seen to work better than Do-nothing. A likely explanation for this behavior is that, since the attribute is important for the correct classification, it is worth waiting for the delayed attribute to arrive either for reclassifying the object (Put-and-reclassify), or for incorporating the object into the reference set only once it is complete (Wait-and-classify).
- Using only the initial reference set $S_0$ seems to behave slightly better than the strategies here proposed when the delayed attribute was the least relevant. In this case, it might happen that this attribute is hindering the classifier rather than helping it. As a consequence, and in the context of the 1-NN classifier, it turns out to be better to passively ignore all the incoming objects than trying to make the most of them.
As a final comment for the 1-NN classifier, although differences between the early classification techniques and the baseline case (which uses only the full and labeled instances in the initial reference set $S_0$) do not appear to be significant, it seems that the use of some active strategy (examples of which are our Put-and-reclassify and Wait-and-classify) allows to improve the classification results of incomplete data streams.

Results obtained with the SVM on the same three data sets are plotted in Fig. 2. In this case, although the accumulated classification error also decreases over time, it is apparent that the early classification strategies proposed in the present paper cannot outperform the baseline approach, irrespectively of the relevance of the attribute that arrives with a time delay. These results suggest that the proposed strategies might be more suitable for instance-based classifiers, such as the nearest neighbor, than for an SVM-based classifier. While beyond the scope of this paper, it should be explored whether other configurations of the SVM or more elaborate strategies (or both) would be required for the challenging scenario of early classification of incomplete streaming data.

Figure 3 depicts the incremental learning curves when using the confidence-based 5-NN classifier. Apart from several characteristics common to the results of the two classifiers previously analyzed (i.e., the decrease in the accumulated classification error), the curves corresponding to this new classifier allow to add a couple of remarkable comments:

- In the case of a delay in the most relevant attribute, there exist significant differences between the three early classification strategies and the baseline; these differences are especially significant for the vowel data set, which corresponds to a non-binary problem with 10 classes. The active strategies Put-and-reclassify and Wait-and-classify perform better than Do-nothing, like in the case of the 1-NN rule.
When the delayed attribute was the least relevant, the early classification techniques also outperform the baseline case, but now there do not exist significant differences between the active strategies (Put-and-reclassify and Wait-and-classify) and the passive one (Do-nothing). In fact, for the crabs data set this passive strategy is even better than the active ones.

The analysis of the results attained with the three classifiers here explored suggests that the use of a confidence-based classifier provides some advantages over the conventional algorithms when employed in a context of streaming data with delayed attributes. Also, it seems that the Put-and-reclassify and Wait-and-classify strategies work better than Do-nothing, especially when the delayed attribute corresponds to the most relevant.

As a further confirmation of the findings using the incremental learning curves, and to gain insight into the general performance of the proposed strategies and classifiers over the data sets used, additional alternative representations of the results are offered.

First, for each classifier, a histogram of how many times the proposed strategies outperform the baseline case of just using the initial reference set $S_0$ across all data sets, was computed. To perform this count, we considered the final classification error (i.e. after the time step $t = N$, when all streaming objects have already been seen). A strategy $A$ with final classification error $e_A$ is considered to outperform strategy $B$ with error $e_B$ if $\frac{e_A - e_B}{e_B} \times 100 < \theta_e$ for a given threshold $\theta_e \geq 0$ on the relative error improvement.

Figure 4 shows these histograms (for $\theta_e = 0$) which further supports the claim that the best option to classify streaming data with delayed attributes corresponds to the combined use of the proposed strategies with the confidence-based 5-NN classifier. In fact, when using this classifier, the three strategies outperform the baseline case in 11 out of the 15 data sets (73%), both for the least and the most relevant attributes being delayed. The opposite
case occurs with the SVM, where the use of the initial reference set clearly appears as the best alternative. The 1-NN classifier exhibits an intermediate behavior between the other two classifiers.

To get an idea of how much improvement is obtained when the three strategies outperform the baseline strategy, Fig. 5 represents, for the confidence-based 5-NN classifier, the evolution of the number of data sets in which this happens, by varying the threshold $\theta_e$ used to define when a strategy beats another one. It can be noticed that, while the performance decays with increasing $\theta_e$, as it can naturally be expected, the behavior is reasonably good even for significant reductions in the final classification error. For instance, in the case of the most relevant delayed attribute, for an error improvement $\theta_e = 5\%$ the strategies beat the baseline in as many as 10 out of the 15 data sets considered (i.e. 67%), and for $\theta_e = 10\%$, the improvement still occurs in nearly half the number of data sets (7/15 or 47%).

A second view of the results is based on a method of ranking [7]: the strategies were ranked for each data set and each classifier, according to their final classification errors. As there are four competing algorithms (the three proposed strategies plus the baseline case), the possible ranks for each data set and each classifier range from 1 (best) to 4 (worst). Figure 6 plots the averaged ranks versus the classifiers (the average was taken across the data sets). Unlike the histograms above (Fig. 4, the averaged ranks allow to assess the behavior of each strategy with respect to each classifier. As it can be observed in this figure, for the most relevant delayed attribute, Put-and-reclassify and Wait-and-classify significantly outperform the Do-nothing strategy, irrespectively of the classifier used. It is also interesting to note that there is not a significant difference between the two active strategies.

4 Conclusions

We have presented a preliminary study for handling on-line data where the complete attribute vector arrives with a
constant delay. More specifically, we have explored three strategies for learning from streaming data with a single delayed attribute. Besides, three classifiers have been tested and combined with these strategies. Interestingly, in the proposed approach, the strategies and the classifiers are kept highly independent, which simplifies the design of the system. In spite of their simplicity, the results of the three strategies have shown some gains in performance when compared to the use of the initial reference set, especially in the case of incorporating some confidence measure into the classifier. Although these benefits are still marginal in some data sets, the most important finding is that it seems possible to design some method to consistently handle the incomplete data in on-line classification of data streams.

There are several very interesting open research questions to work on. For instance, while recent results [15] suggest that different strategies may be differently sensitive to different amounts of time delays, a more thorough and extensive analysis on this influence deserves further future consideration. Other issues include the design of smarter strategies which can better cope with missing/delayed attributes, and consider a time-varying environment (concept change or concept drift).

Acknowledgments This work has been supported in part by the Spanish Ministry of Education and Science under grants CSD2007-00018 Consolider Ingenio 2010 and TIN2009-14205, and by Fundació Caixa Castelló—Bancaixa under grant P1-1B2009-04.

References

1. Agarwal C (2004) On-demand classification of data streams. In: Proceedings of the ACM international conference on knowledge discovery and data mining, pp 503–508
2. Agarwal C (2007) Data streams: models and algorithms. Springer, New York
3. Asuncion A, Newman DJ (2007) UCI machine learning repository. School of Information and Computer Science, The University of California, Irvine, CA. http://www.archive.ics.uci.edu/ml/
4. Babcock B, Babu S, Datar M, Motwani R, Widom J (2002) Models and issues in data stream systems. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, pp 1–16
5. Bruzzone L, Roli R, Serpico SB (1995) An extension of the Jeffreys–Matusita distance to multiclass cases for feature selection. IEEE Trans Geosci Remote Sens 33(6):1318–1321
6. Chang CC, Lin CJ (2001) LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
7. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
8. Ganti V, Gehrke J, Ramakrishnan R (2001) Demon: mining and monitoring evolving data. IEEE Trans Knowl Data Eng 13(1):50–63
9. Gelman A, Meng XL (2004) Applied Bayesian modeling and causal inference from incomplete data perspectives. Wiley, Chichester
10. Hashemi S, Yang Y (2009) Flexible decision tree for data stream classification in the presence of concept change, noise and missing values. Data Min Knowl Discov 19(1):95–131
11. Keerthi SS, Lin CJ (2003) Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Comput 15(7):1667–1689
12. Kuncheva LI (2008) Classifier ensembles for detecting concept change in streaming data: overview and perspectives. In: Proceedings of the 2nd workshop on supervised and unsupervised ensemble methods and their applications, pp 5–10
13. Maimon O, Rokach L (2005) Data mining and knowledge discovery handbook. Springer Science Business Media, New York
14. Marwala T (2009) Computational intelligence for missing data imputation, estimation and management: knowledge optimization techniques. Information Science Reference, Hershey
15. Millán-Giraldo M, Sánchez JS, Traver VJ (2009) Exploring early classification strategies of streaming data with delayed attributes. In: 16th International conference on neural information processing, LNCS 6863, part I, Bangalore, pp 875–883
16. Millán-Giraldo M, Duin RPW, Sánchez JS (2010) Dissimilarity-based classification of data with missing attributes. In: The 2nd international workshop on cognitive information processing (submitted)
17. Muthukrishnan S (2005) Data streams: algorithms and applications. Found Trends Theor Comput Sci 1(2):117–236
18. Ripley BD (1996) Pattern recognition and neural networks. Cambridge University Press, Cambridge
19. Little RJA, Rubin DB (1987) Statistical analysis with missing data. Wiley, New York
20. Saar-Tsechansky M, Provost F (2007) Handling missing values when applying classification models. J Mach Learn Res 8:1625–1657
21. Street WN, Kim Y (2001) A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the 7th international conference on knowledge discovery and data mining, pp 377–382
22. Takeuchi J, Yamanishi K (2006) A unifying framework for detecting outliers and change points from time series. IEEE Trans Knowl Data Eng 18(4):482–492
23. Tsymbal A (2004) The problem of concept drift: definitions and related work. Technical report, Department of Computer Science, Trinity College, Dublin
24. Vázquez F, Sánchez JS, Pla F (2005) A stochastic approach to Wilsons editing algorithm. In: Proceedings of the 2nd Iberian conference on pattern recognition and image analysis, pp 35–42
25. Widyantoro DH, Yen J (2005) Relevant data expansion for learning concept drift from sparsely labeled data. IEEE Trans Knowl Data Eng 17(3):401–412