Nearest Neighbour Classifiers for Streaming Data with Delayed Labelling

Ludmila I. Kuncheva
School of Electronics and Computer Science
University of Wales
Bangor, Gwynedd, LL57 1UT, UK
l.i.kuncheva@bangor.ac.uk

J. Salvador Sánchez
Dept. Llenguatges i Sistemes Informàtics
Universitat Jaume I
Av. Sos Baynat s/n, 12071 Castelló, Spain
sanchez@uji.es

Abstract

We study streaming data where the true labels come with a delay. The question is whether the online nearest neighbour classifier (IB2 and IB3 here) should employ the unlabelled data. Three strategies are examined: do-nothing, replace and forget. Experiments with 28 data sets show that IB2 benefits from unlabelled data, while IB3 does not.

1. Introduction

Most traditional learning algorithms assume the availability of a training set of labelled objects. In many domains, however, collecting labelled training objects may be costly, time-consuming, dangerous or destructive, while it is relatively easy to obtain unlabelled objects. This has provoked significant interest in semi-supervised learning methods [13, 15, 17].

Amongst the success reports with semi-supervised learning, researchers voice their concerns that the classifiers might deteriorate rather than improve with unlabelled data [2–4, 17]. The main result by Cozman et al. [3, 4] demonstrates theoretically that if the model is guessed correctly, unlabelled data is expected to improve on the error. However, if there is a modelling error (incorrect guess about the shape of the data distribution), using unlabelled data may do more harm than good.

The problem is exacerbated further when the data is not available as a batch but comes one object at a time (called *streaming data*), especially where a concept drift might be present. The scenario we consider in this study was inspired by an example due to Kelly et al. [7]. Suppose that a bank manager is using a decision support system to help them decide whether to approve a loan to client X. The input to the system includes X's current salary. The problem description changes with time because of gradual changes in many demographic factors, or simply because of inflation. A person with X's salary may no longer be eligible for a loan in a few years time. The classifier will become obsolete if it is not being updated with the incoming data. The problem, however, is that the true label of the data point does not always become available straight after the classifier makes the prediction, so the new object cannot be used as feedback. In the above example, if the loan was approved, the true label (either good loan or bad loan) will become available a couple of years later, when X is either making regular repayments or failing to do so. The question here is whether we should update the classifier using the unlabelled data until the labels become available, or, in the light of the criticism above, wait until we get the labels. There is different type of risk associated with the two strategies.

Here we study this problem which we call *delayed labelling*. Although any online classifier can be considered, we chose, as a starting point, to study the online nearest neighbour classifier (NNC). The main reason was that, being a non-parametric classifier, nearest neighbour does not make any modelling assumption about the probability densities, and so cannot make wrong guesses to harm the semi-supervised learning. Also, before embarking on the full scenario with semi-supervised learning *and* concept drift, here we look at delayed labelling for static problems.

2. Online nearest neighbour classifiers

A good online classifier for both static and dynamic environments should have the following qualities [5, 11]: (i) Single pass through the data (the classifier must be able to learn from each data point without revisiting it); (ii) Limited memory and processing time (each data point should be processed in a constant time regardless of the number of points processed in the past); and (iii) Any-time-learning (if stopped at time t, the current classifier should be equivalent to a classifier trained on the batch data up to time t).

Online NNCs can draw upon a rich body of data editing methods [16]. The current 'classics' are still the three intuitive nearest neighbour online versions (Instance-Based learning) IB1, IB2 and IB3 by Aha et al. [1]. The main decision in designing an online NNC is how to maintain the reference set. The three trivial possibilities with respect to the memory size are [8]: (1) Full memory, in which the learner retains all training objects; (2) Partial memory, in which it retains only some of the training objects; and (3) No memory, in which it retains none.

Online NNC should be in the partial memory group. Ideally, we need to keep a (small) reference set which, when deemed necessary, is expanded or shrunk within given limits. This means that there must be some mechanism to forget (remove) objects. How to forget is a difficult problem that can be tackled through several strategies.

• Passive forgetting [10] (also called time-weighted forgetting [14] and implicit forgetting [14]) is based only on the time elapsed since the object was added to the reference set. It assumes that the importance of data decreases over time. Passive forgetting acts as a moving window where the reference set is the last data batch. Its size is a parameter of the algorithm.

• Active forgetting [10] (also referred as to explicit forgetting [14]) implies that more information from data is used to decide which objects should be dropped.

(i) Density-based forgetting follows the intuition of the "life" game. If a region is too crowded, we sieve out some objects (locally-weighted forgetting [14]). On the other hand, if a region is too distant, and not providing nearest neighbours, it is removed altogether [14]. In the jargon of data editing, the former strategy corresponds to condensing, while the latter corresponds to editing.

(ii) Error-based forgetting is perceived as the most successful of the forgetting heuristics [1, 10, 14]. In this case each object in the reference set has a classification record. The more streaming data it labels correctly as their nearest neighbour, the stronger its record becomes. The objects with weak records are cleared at regular intervals.

2.1. IB2 and IB3

In this paper, we are interested in partial memory NNCs. Our pilot experiments favoured error-based forgetting compared to passive forgetting and density-based active forgetting. Although IB2 (with no forgetting) and IB3 [1] lead to relatively large reference sets, we chose them because of their sensible results in terms of classification error.

IB2 starts with an empty training set S. Upon receiving a new object, it classifies it using the objects currently held in memory. If the classification is correct, the object is discarded. Conversely, if the classification is incorrect, the object is added into S. This constitutes a one-pass version of the Hart's condensing algorithm [6], and will grow the reference set as long as there are misclassifications on the streaming data.

Note that in the original IB2 algorithm, the first object is added automatically to the empty reference set. In our case, an initial labelled reference set is already in place, therefore the first object from the online data may or may not be added to that, depending on the class labels it receives.

IB3 extends IB2 by adding a forgetting mechanism as shown in Table 1. We found that the choices in the initialisation of IB3 have a serious effect on the algorithm's performance, so we specify these, as well as our data organisation, in greater detail than we have seen elsewhere.

IB3 starts with an initial reference set S, where all objects are designated as 'accepted'. When a new object $\mathbf{x}$ is received, IB3 searches for its nearest neighbour s^* among the accepted objects in S. If there are no accepted objects, a random object s^* is picked and nominated to serve as the nearest neighbour. If the new object is not correctly classified by its nearest neighbour, it is added to S. Next the algorithm revises the acceptance records of those objects in S that are no further from $\mathbf{x}$ than s^* is. If they label $\mathbf{x}$ correctly, their accuracy scores increase. The prior probabilities are updated. A check is carried out to find out whether the accuracies of the objects being updated are significantly higher than the prior probabilities for their respective class (90% confidence interval). If so, they are designated as 'accepted'. If the accuracy score of such an object is significantly lower than the respective prior (70% confidence interval), the object is removed from S. All objects in-between the two decisions stay as 'non-accepted' in S and are re-evaluated if they happen to be selected for updating.

At the beginning of the algorithm, the accuracy records have very high variance because they are based on few objects. Having a conservative acceptance policy, IB3 quickly reduces the initial reference set to status 'non-accepted' and practically starts from scratch building up a new reference set. A warm-up stage, where all objects are accepted or at least no object is removed, may help in stabilising the starting phase.

3. Strategies to handle delayed labelling

At step t in the delayed labelling scenario, we have a reference set S_t and a new unlabelled object $\mathbf{x}_{t+1}$. After predicting the label for $\mathbf{x}_{t+1}$ we receive the label of the object that came τ steps earlier, $\mathbf{x}_{t-\tau+1}$. Objects from $\mathbf{x}_{t-\tau+2}$ to $\mathbf{x}_{t+1}$ are still without labels. The question is whether we want to use the unlabelled data to update S, and, if so, how?

The algorithms selected for this study are error-driven because they update the reference set if there is an error in the prediction. They were chosen exactly for this property as the other forgetting strategies were found to be inferior to that. If the true label of $\mathbf{x}_{t+1}$ is unknown, there is no obvious way to check the accuracy of the prediction. We propose to use a heuristic, which we call conditional labelling. The two nearest neighbours of $\mathbf{x}_{t+1}$ are found in S, say $s^{(1)}$

Table 1. The IB3 algorithm

```
1. Choose a random initial reference set $S$.
2. For every $s \in S$, set up its accuracy record as $A(s) = 1$, its acceptance record as $B(s) = 1$, and its count of number of updates $C(s) = 1$.
3. Estimate the prior probabilities for the $c$ classes using the current reference set, $P(i) = N_i/N$, where $N_i$ is the number of reference objects from class $i$, and $N$ is the total number of objects, $i = 1, \dots, c$.
4. Let $\mathbf{x}$ be the new object, and let $y$ be its true class label. Let $\beta = \{s|B(s) = 1\}$ be the set of all *accepted* objects in $S$. If $\beta = \emptyset$ (no accepted objects), then select a random object $s^*$ from $S$, else take $s^*$ to be the element in $\beta$ that is closest to $\mathbf{x}$.
5. Assign the label of $s^*$ to be the predicted label, $y_{\text{predicted}}$, of $\mathbf{x}$. If $y_{\text{predicted}} = y$, continue from step 7 with the next object. Otherwise proceed with step 6.
6. Add $\mathbf{x}$ to $S$. Assign accuracy $A(\mathbf{x}) = 1$, acceptance $B(\mathbf{x}) = 1$ and count $C(\mathbf{x}) = 1$.
7. Update the priors and the total counts. $P(i) \leftarrow P(i)N/(N + 1)$ for $i \neq y$, and $P(i) \leftarrow (P(i)N + 1)/(N + 1)$ for $i = y$. Set $N_y \leftarrow N_y + 1$ and $N = \sum N_i$
8. Loop through all objects which are at the same distance to $\mathbf{x}$ as $s^*$ is, or closer. Let $s$ be one such object, and let $y_s$ be its label.
    • Update the accuracy records. If $y_s = y$ ($s$ predicts the correct label for $\mathbf{x}$), update $A(s) \leftarrow (A(s)C(s) + 1)/(C(s) + 1)$, else ($y_s \neq y$) $A(s) \leftarrow A(s)C(s)/(C(s) + 1)$.
    • Update the accuracy counts. $C(s) \leftarrow C(s) + 1$
    • Revise the acceptance status of $s$. Let $p = P_{y_s}$ be the prior probability for the class which $s$ belongs to, and $a = A(s)$ be the updated accuracy record. Calculate the 90% confidence intervals for $a$ and $p$, $[L90_a, U90_a]$ and $[L90_p, U90_p]$ using
      $$CI = \frac{\left(\xi + \frac{z^2}{2n} \pm z\sqrt{\left(\frac{\xi(1-\xi)}{n} + \frac{z^2}{4n^2}\right)}\right)}{1 + \frac{z^2}{n}} \quad (1)$$
      with $z = 0.9$, $\xi = a$ and $n = C(s)$ (or $\xi = p$ and $n = N_{y_s}$). If $L90_a > U90_p$, then $B(s) = 1$ (accept $s$ for the next reference set) else $B(s) = 0$. Calculate in the same way the 70% confidence intervals, using $z = 0.7$. If $U70_a < L70_p$, remove $s$ from the data set altogether.
9. Continue from step 4. At any time $t$, the reference set of IB3 is the set of accepted objects only.
```

and $s^{(2)}$. If they have the same class label, we assume that the prediction for $\mathbf{x}_{t+1}$ is correct, and if the two labels do not match, we count a wrong prediction. As $y_{\text{predicted}}$ is needed as the label with which $\mathbf{x}$ will be stored in S, we take $y_{\text{predicted}}$ to be the label of the nearest neighbour $s^{(1)}$.

The strategies proposed to handle delayed labelling are:

• **Do-nothing.** This is the passive strategy where we let the classifier run without any update on the unlabelled data. The online nearest neighbour method will be behind by τ time steps. To make a classification decision at step t, the classifier would have seen the labels of objects up to $t - \tau$.

• **Replace.** With this strategy we assume that conditional labelling is carried out and some of the objects from the unlabelled batch, $\mathbf{x}_{t-\tau+1} \dots \mathbf{x}_t$, would be included in S_t. If $\mathbf{x}_{t-\tau+1}$ is not one of the objects included in S_t, we proceed with the next object. If $\mathbf{x}_{t-\tau+1}$ is in S_t, we check its label. If $y_{\text{predicted}}$ is the correct label for $\mathbf{x}_{t-\tau+1}$, then we do nothing and continue with the next object. If, however the predicted and the true label of $\mathbf{x}_{t-\tau+1}$ mismatch, we replace $y_{\text{predicted}}$ with the correct label.

• **Forget.** This is the same as Replace, up to the decision about what to do if the predicted and the true labels of $\mathbf{x}_{t-\tau+1}$ do not match. Here we remove $\mathbf{x}_{t-\tau+1}$ from S_t.

The case of $\tau \rightarrow \infty$ means that the true labels are never received. Thus the classifier keeps updating the reference set according to the conditional labelling heuristic, and never stops to revise past labels. Hence the 'replace' and the 'forget' strategies do not get a chance to be applied, and the two methods are equivalent to conditional labelling all the way through.

4. Experiments

Twenty eight real data sets were employed in the present experiment (see a summary in Table 2). Data were normalised in the range $[0, 1]$. All features were numerical and there were no missing values. In the table, the data sets are sorted by the total number of objects.

Although there is no strict guideline about what a sufficient data size is, the common wisdom [9] is that the size of the training data should be around $10 \times n \times c$, where n is the number of features and c is the number of classes in a problem. Our small initial reference set was of size $1 \times n \times c$.

The experimental set-up was as follows:

• 100 runs were carried out with 90% of the data used for training and 10% used for testing. The splits were done

Table 2. Data sets used in the experiment

Data set	Features	Classes	Objects	Source
iris	4	3	150	UCI[1]
wine	13	3	178	UCI
spiral	2	2	194	UCI
crabs	6	2	200	Ripley [12]
sonar	60	2	208	UCI
laryngeal1	16	2	213	Library[2]
glass	9	6	214	UCI
thyroid	5	3	215	UCI
votes	16	2	232	UCI
voice3	10	3	238	Library
breast	9	2	277	UCI
intubation	17	2	302	Library
heart	13	2	303	UCI
ecoli	7	8	336	UCI
liver	6	2	345	UCI
spect	44	2	349	Library
ionosphere	34	2	351	UCI
laryngeal3	16	3	353	Library
voice9	10	9	428	Library
wbc	30	2	569	UCI
palynomorphs	31	3	609	Private[3]
australian	42	2	690	UCI
laryngeal2	16	2	692	Library
pima	8	2	768	UCI
vehicle	18	4	846	UCI
vowel	11	10	990	UCI
german	24	2	1000	UCI
image	19	7	2310	UCI

[1]UCI `http://archive.ics.uci.edu/ml/`

[2]Library `http://www.informatics.bangor.ac.uk/~kuncheva/activities/real_data_full_set.htm`

[3]Images of pieces of kerogen extracted from microscope images of palynomorphs

using stratified sampling.

• From each training part of the data, a random stratified sample of $N_l = 1 \times n \times c$ was taken as the initial labelled references set.

• The remaining part of the training data was used as the new coming online data. To simulate an i.i.d. sequence, the data was shuffled before each of the 100 runs.

• One point from the online data was fed to the system at a time. The point was processed according to the respective strategy to handle delayed labelling. The classification error was evaluated on the testing set. In this way we created a "progression curve" which is the error as a function of the number of online objects seen by the classifier.

• The results were averaged across the 100 runs giving a single progression curve for the data set.

The methods we compare through this experiment are:

• **1-NN whole.** The NNC on the whole 90% training data taken as the reference set.

• **Initial.** To evaluate whether the nearest neighbour improves at all with streaming data (labelled or with delayed labels), we report the testing error using only the initial reference set.

• **IB2 do-nothing, IB2 replace, IB2 forget.** The values of the delay τ were 0, 5, 10, 15, 20, 25, 30, 35, 40, and ∞. For $\tau = 0$, 'IB2 do-nothing' is the standard IB2. For $\tau \rightarrow \infty$, all three strategies reduce to IB2 with conditional labelling.

• **IB3 do-nothing, IB3 replace, IB3 forget.** τ = 0, 5, 10, 15, 20, 25, 30, 35, 40, and ∞.

For each of the 100 runs of the experiment, all methods received the same partitions of the data into initial, online and testing sets. The online data was presented to all methods in the same order.

4.1. The results: classification error

The methods were ranked for each data set and for each τ using the classification errors. For IB2 and IB3 which give progression curves, we took the error at the end, when all streaming objects have been seen. As there are 8 competing methods, the ranks for each data set and each τ were from 1 (best) to 8 (worst).

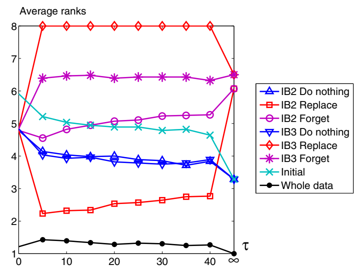

Figure 1. Ranks for the IB2 and IB3 variants.

Figure 1 plots the averaged ranks versus τ (the average was taken across the data sets). The figure reveals that the three strategies for handling delayed labelling behave very differently with the two methods. While IB2 benefits substantially from the 'replace' strategy, for IB3 this strategy is a catastrophe. What is more interesting though is that the 'replace' strategy with IB2 confidently outperforms the 'do-nothing' strategy. We used the same unlabelled data submitted in the same order for both IB2 and IB3, with the same philosophy adopted by the conditional labelling. It is curious that the success of a simple heuristic depends so heavily on the details of the online NNC.

The 'do-noting' strategy is ranked higher than the 'forget' strategy with both IB2 and IB3. For larger τ the 'forget' strategy is even worse than Initial. It is not surprising that

Table 3. Comparison of strategies

	(2)	(3)	(4)	(5)	(6)
(1) 1-NN whole	230/50/0	256/24/0	191/66/23	249/31/0	242/28/10
(2) Initial		65/89/126	33/38/209	78/145/57	33/127/120
(3) IB2 do-nothing			76/91/113	145/96/39	90/123/67
(4) IB2 replace				224/56/0	128/89/63
(5) IB2 forget					38/86/156
(6) IB3 do-nothing					

'1-NN whole' was the clear winner of this contest, as it uses the whole training set as a reference.

As a further confirmation of the findings using the ranks, we ran paired t-test between each pair of methods, for each data set, and for each τ. We dropped from the analyses 'IB3 replace' and 'IB3 forget' because they would only contaminate the comparison. The three values in the cells of Table 3 show how many times the method of the row has been significantly-better/same/significantly-worse than the method of the column, with a confidence interval of 95%. Note that for each pair of methods there are a total of 280 tests, since we have 28 databases and 10 values of τ.

Table 4. Index of performance

	Wins	Losses	Index
(1) 1-NN whole	1168	33	1135
(4) IB2 replace	697	363	334
(3) IB2 do-nothing	437	540	−103
(6) IB3 do-nothing	416	531	−115
(2) Initial	209	742	−533
(5) IB2 forget	134	852	−718

Table 4 reports a ranking of the methods in terms of an index of performance. For each method A, the index of performance is calculated as the difference between *wins* and *losses*, where *wins* is the total number of times that A has been significantly better than another method and *losses* is the total number of times that A has been significantly worse than another method. From the results in Table 4, it seems that the 'IB2 replace' strategy is significantly better than the other methods.

4.2. The results: size of the reference set

The main question is whether the error reduction with the 'replace' strategy is compensated for by larger reference sets. To answer this question we compare 'IB2 replace' with the next best method from Table 4, 'IB2 do-nothing'. For data set i we calculated: the Error Difference

$$\Delta_{\text{error}} = E_{i,\text{IB2 do-nothing}}(\tau) - E_{i,\text{IB2 replace}}(\tau),$$

where $E_{i,X}(\tau)$ is the error of method X on data set i for delay τ, and the Size Difference

$$\Delta_{\text{size}} = S_{i,\text{IB2 do-nothing}}(\tau) - S_{i,\text{IB2 replace}}(\tau),$$

where $S_{i,X}(\tau)$ is the size of the final reference set of method X on data set i for delay τ.

If $\Delta_{\text{error}} < 0$ and $\Delta_{\text{size}} < 0$, 'IB2 do-nothing' is better than 'IB2 replace' on both criteria, and when $\Delta_{\text{error}} > 0$ and $\Delta_{\text{size}} > 0$, 'IB2 do-nothing' is worse than 'IB2 replace' on both criteria. Out of the 28 data sets, for $\tau = 10$, 'IB2 do-nothing' was better on 8 and 'IB2 replace' was better on 10 data sets. Figure 2 shows a scatterplot of the 28 data sets on Δ_{error} versus Δ_{size}. The regions where one of the method is clearly better than the other are shaded. Note that most of the remaining data sets fall in the quadrant where 'IB2 replace' is better on the error and worse on the size. However, for $\tau = 30$ the success is reversed. 'IB2 replace' is clearly better (on both criteria) on 8 data sets and 'IB2 do-nothing' is clearly better on 12 data sets.

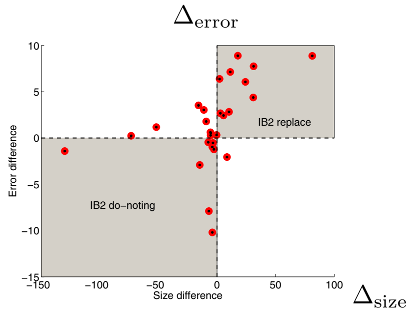

Figure 2. Scatterplot of the 28 data sets for 'IB2 replace' and 'IB2 do-nothing' for $\tau = 10$.

IB3 usually produces smaller reference sets than IB2. Hence we decided to compare 'IB2 replace' with the third best method,'IB3 do-nothing'. The results are shown in Table 5 for all values of τ from 5 to 40. They indicate that 'IB2 replace' does indeed pay the price for its low error by selecting larger reference sets compared to the 'do-nothing' strategy, especially with IB3.

It would have been nice if 'IB2 replace' was the undisputed champion and we had a ready made recommendation for handling delayed labelling. Unfortunately, this is not the case. The purpose of this study was to explore some possible avenues, and gather insight into the problem.

Table 5. Number of data sets where the first method is better than the second on both error and size.

τ	5	10	15	20	25	30	35	40
$(3) > (4)$	8	8	8	10	11	12	12	12
$(4) > (3)$	11	10	10	9	8	8	7	7
$(6) > (4)$	9	9	9	11	12	13	13	13
$(4) > (6)$	0	0	0	0	0	0	0	0

5. Conclusions

We explored three strategies for handling delayed labelling using two online NNCs, IB2 and IB3. The results identified IB2 with the 'replace' strategy and IB3 with 'do-nothing' strategy as the best compromises between classification error and the size of the reference set. Thus the main question stated in the Introduction is still open for debate: Is it better to use the unlabelled data (IB2 with replace) or ignore it altogether (IB3 with do-noting)? We did not find clear evidence either way, which resonates well with the mixed opinions about semi-supervised learning.

Where do we go from here? This study only opened the question about delayed labelling. It is interesting to investigate the case where the environment does change with time, and the reference sets will have to follow the changes. IB3 has a natural mechanism to do so by forgetting, while IB2 does not. The preferences with respect to the three strategies is likely to change in favour of the forgetting. It will contribute to reducing the size as well as keeping the reference set up to date. The "lazy" do-nothing strategy is expected to be inferior to the more proactive conditional labelling or another version thereof. Thus one direction for future studies would be investigating the same set-up with concept drift.

Both IB2 and IB3 may grow the reference sets without limit. In our experiments we observed constant steady grow with both methods, more markedly so with IB2, seeming never to reach a saturation stage. Large reference sets are not necessarily the best, and are unsuitable for online classifiers. Having a progressively growing reference set violates point (ii) from the desiderata list in Section 2. It is interesting to attempt to restrict the size of the reference set. One possible way to do so is to take advantage of the variety of the available data editing methods.

Acknowledgment

Work partially supported by the Spanish Ministry of Education and Science, grants DPI2006–15542 and CSD2007–00018, and by EPSRC, grant #EP/D04040X/1.

References

[1] D. Aha, D. Kibler, M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[2] O. Chapelle, B. Schölkopf, A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[3] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, T. S. Huang. Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(12):1553–1568, 2004.

[4] F. G. Cozman, I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *Proc. 15th Intl. FLAIR Conf.*, Pensacola, FL, pp. 327–331, 2002.

[5] P. Domingos, G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 12(4):945–949, 2003.

[6] P.E. Hart. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory*, 14(3):515–516, 1968.

[7] M. G. Kelly, D. J. Hand, N. M. Adams. The impact of changing populations on classifier performance. In *Proc. 5th ACM Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, pp. 367–371, 1999.

[8] M. A. Maloof, R. S. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41(1):27–52, 2000.

[9] G. Nagy. Classifiers that improve with use. In *Proc. Conf. on Pattern Recognition and Multimedia*, Tokyo, Japan, pp. 79–86, 2004.

[10] H. Nakayama, K. Yoshii. Active forgetting in machine learning and its application to financial problems. In *Proc. Intl. Joint Conf. on Neural Networks*, Como, Italy, pp. 123–128, 2000.

[11] N. C. Oza. *Online Ensemble Learning*. PhD thesis, University of California, Berkeley, CA, 2001.

[12] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[13] F. Roli. Semi-supervised multiple classifier systems: Background and research directions. In *Proc. 6th Intl. Workshop on Multiple Classifier Systems*, Seaside, CA, pp. 1–11, 2005.

[14] M. Salganicoff. Density-adaptive learning and forgetting. In *Proc. 10th Intl. Conf. on Machine Learning*, Amherst, MA, pp. 276–283, 1993.

[15] M. Seeger. Learning with labeled and unlabeled data. Technical Report, University of Edinburgh, UK, 2002.

[16] D. R. Wilson, T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[17] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin, Madison, WI, 2005.