Combining Online Classification Approaches for Changing Environments

Juan J. Rodríguez[1] and Ludmila I. Kuncheva[2]

[1] Lenguajes y Sistemas Informáticos, Universidad de Burgos, Spain
jjrodriguez@ubu.es
[2] School of Computer Science, Bangor University, UK
l.i.kuncheva@bangor.ac.uk

Abstract. Any change in the classification problem in the course of online classification is termed changing environments. Examples of changing environments include change in the underlying data distribution, change in the class definition, adding or removing a feature. The two general strategies for handling changing environments are (i) constant update of the classifier and (ii) re-training of the classifier after change detection. The former strategy is useful with gradual changes while the latter is useful with abrupt changes. If the type of changes is not known in advance, a combination of the two strategies may be advantageous. We propose a classifier ensemble using Winnow. For the constant-update strategy we used the nearest neighbour with a fixed size window and two methods with a learning rate: the online perceptron and an online version of the linear discriminant classifier (LDC). For the detect-and-retrain strategy we used the nearest neighbour classifier and the online LDC. Experiments were carried out on 28 data sets and 3 different scenarios: no change, gradual change and abrupt change. The results indicate that the combination works better than each strategy on its own.

1 Introduction

A change in a real-life classification problem may arise from demographic fluctuations, economic trends, seasonal drifts, change of measuring equipment, etc. Here we use the term *changing environments* to denote any change in the classification problem (also called concept drift, population drift, concept shift). Two general strategies to ensure that the classifier is up-to-date are the constant update of the classifier and update upon change detection. Various methods for classification in changing environments have been proposed [6]. Within the constant-update strategy group, many methods use a fixed window of objects or a fixed learning rate. Within the detect-and-retrain strategy group, changes are detected explicitly [3]. Ensembles of classifiers for changing environments have also been considered [7, 9, 10, 4], mostly following the constant-update strategy. Which method will be adequate for a particular problem depends on the types and frequencies of the changes. These are rarely known in advance. Hence, this work proposes to combine methods coming from the two strategies, using an ensemble of classifiers.

The rest of the paper is organised as follows. Section 2 describes the online classification methods used in the ensemble. Section 3 contains the experiment, and Section 4 gives the concluding remarks.

2 Methods

2.1 Classification Methods

Three families of methods are considered:

- The nearest neighbour (1-NN) classifier with a fixed window. The reference set of the nearest neighbour are the latest M objects in the streaming data. Each new object is added to the reference set and the oldest one is discarded.
- The perceptron with a fixed learning rate η. The coefficients of a linear function are updated if the current object is misclassified. This method is designed for two-class problems. For multiclass problems, pairwise classification can be used [2] (either one versus all or all pairs of classes).
- An online version of the linear discriminant classifier (O-LDC). The classifier is updated with each new object by recalculating the means and the inverse of the covariance matrix using a shortcut formula. The strength of the updates is governed by a learning rate parameter, λ, with a value between 0 and 1. For $\lambda = 0.5$, all the objects have the same importance; if $\lambda > 0.5$, the new objects are more important than the older ones.

2.2 Change Detection Methods

The two detection methods considered here monitor the probability of error of the streaming objects. When a change in the error is detected (significant raise) the current classifier is abandoned and a new one is trained.

- The Drift Detection Method (DDM) proposed in (Gama et al., 2004) operates with the help of two thresholds, t_1 and t_2. When the average error rate e goes above t_1, a warning mode is activated and the time of the activation, t, is recorded. If e falls back bellow t_1, the warning mode is disabled. The second threshold is used for detecting the change. As soon as $e > t_2$, a change is detected. The new classifier is constructed using the objects that arrived after the latest recorded onset of change, t.
- The Sequential Probability Ratio Test (SPRT) charts are used in industrial engineering for process quality control [8]. Instead of the error rate e, the cumulative sum of errors, e_c, is being monitored. There are three thresholds, t_0, t_1, t_2. An SPRT is started, and at each step the cumulative error is compared with a reference value t_0. If the difference $e_c - t_0$ is greater than t_1, then change is signalled, the current SPRT is terminated, and a new SPRT is started. Otherwise, if $e_c - t_0 < t_2$, then the error rate is considered to be steady, the current SPRT is terminated, and a new SPRT is started. If $t_2 \leq e_c - t_0 \leq t_1$, SPRT continues with the next object.

Note that the change detection is not tied to the classifier model. Most often an online classifier model is used (constant-update), and upon detection of change, the window size or the learning rate are corrected to account for the change (detect-and-retrain). In our study, a classifier that does not have a forgetting mechanism is used. The current classifier is discarded, a new empty classifier is constructed and initialised with the object that triggered the detection.

2.3 Winnow

Winnow is a method for combining the votes of several experts using streaming data [5]. Substituting 'classifier' for 'expert', Winnow becomes an online strategy for combining classifiers [4]. Each classifier has a weight. Initially all the weights are set to 1. Given an object, the support for class k is calculated as the sum of the weights of the classifiers that predict that class. The prediction of the ensemble is the class with the largest support. If the prediction of the ensemble is wrong, the weights are updated. If the base classifier is correct, its weight is multiplied by α (a parameter of the method). If the classifier is incorrect, its weight is divided by α.

Winnow can be applied to classifiers of any type, both online and static. The update of the weights itself takes care of the changing environments. The novel idea in this study is to use Winnow with online classifiers in order to combine the benefits of the two strategies for handling changing environments.

3 Experiments

The question we seek to answer is whether combining classifiers with different abilities to react to changes is better than using a single online classifier.

The 28 real data sets used in the experiments are shown in Table 2. The features in all data sets are numerical and there are no missing values. Each dataset was normalized once, before starting the experiments. After the normalization, all the numeric features had mean 0 and standard deviation 1. A random subsample of the dataset was taken to be the testing set (10%), and the remaining data (90%) was the training set. One hundred such splits were made, and the classification error was estimated as the average of the 100 testing errors.

3.1 Scenarios

Three scenarios are considered: no change, gradual change and abrupt change.

No change. An initial classifier is constructed using a sample of 50 objects from the training data. Then, for 1000 iterations, a training object is selected randomly, the classifier is updated using this object and the error of the current classifier is estimated using all the testing data. The error of the method is the average error of the classifier for the 1000 iterations. The reported results are obtained as the average for 100 different random partitions of the data.

Table 1. Datasets used in the experiments. Notes: #F is the number of features, #O is the number of objects, #C is the number of classes.

Dataset	#F	#O	#C	Source
breast	9	277	2	UCI
german	24	1000	2	UCI
glass	9	214	6	UCI
heart	13	297	5	UCI
image	19	2310	7	UCI
ionosphere	34	351	2	UCI
iris	4	150	3	UCI
laryngeal1	16	213	2	Collection
laryngeal2	16	692	2	Collection
laryngeal3	16	353	3	Collection
liver	7	344	2	UCI
palynomorphs	31	609	3	Private
pendigits	16	10992	10	UCI
phoneme	5	5404	2	UCI

Dataset	#F	#O	#C	Source
pima	8	768	2	UCI
sat	36	6435	6	UCI
scrapie	14	3113	2	Private
sonar	60	208	2	UCI
spam	57	4601	2	UCI
spect	44	349	2	UCI
thyroid	5	215	3	UCI
vehicle	18	846	4	UCI
voice3	10	238	3	Collection
voice9	10	428	9	Collection
votes	16	232	2	UCI
vowel	10	990	11	UCI
wbc	30	569	2	UCI
wine	13	178	3	UCI

Gradual change. For each training/test partition the numeric features are paired randomly. If the number of features is odd, one of them will be unpaired. In each iteration, the values of the training and test data sets are rotated in the plane defined by each feature pair with an angle of $\pi/1000$. This means that in the last iteration the numeric values of the features will be the original ones with a sign change.

Abrupt change. For each training/test partition a pair of classes is randomly selected. At iteration 400, these two labels are swapped for the training and test data sets. The labels are swapped back at iteration 800.

3.2 Parameter settings

The following parameter values were used in the experiments.

– Four moving window sizes were used with the nearest neighbour (NN) classifier: ∞ (the classifier does not forget), 50, 100 and 150. With smaller fixed window (used as the reference set), the classifier adapts quicker to changes than with larger window. On the other hand, with larger windows the classifier is likely to be more accurate.

– The learning rate η for the perceptron classifier was $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Smaller values of the learning rate mean slower training, hence slower adaptation to changes, and vice versa.

– The learning rate for O-LDC was $\{0.5, 0.7, 0.9, 0.95\}$. For $\lambda = 0.5$ O-LDC does not "forget" any observation, and at time t the classifier is equivalent to a linear discriminant classifier (assuming Gaussian densities) trained on all observations up until t. For $\lambda < 0.5$, O-LDC is "reluctant" to learn: new examples are considered less important than previous examples, and with $\lambda = 0$ O-LDC does not update at all. For $\lambda > 0.5$ the new example outweigh the old examples and the classifier can respond to changes. The higher the value, the more flexible the classifier.

Ideally, the parameter values should be tuned to suit the expected type of change in the environment. If the type and magnitude of the changes can not be evaluated in advance, a meta heuristic should be used to tune the parameters online. Instead of doing this, we propose to use an ensemble of classifiers that consists of classifiers suitable for different types of changes. We chose Winnow as the combination method because it largely follows the classifier selection strategy by constantly revising the weights of the classifiers in the ensemble. Thus, for any type of change, the classifiers (parameter values) that respond best to it will be awarded highest weights. This gives the ensemble a chance to outperform any single online classifier, specialised for a single type of change.

The change detection methods (DDM and SPRT) were combined with the two classifier methods that do not have a forgetting mechanism: NN_∞ and O-$\text{LDC}_{0.5}$.

The 17 classifiers considered in the experiment are

NN window	Perceptron η	O-LDC λ	Detection-based method-classifier
∞	0.1	0.5	SPRT-NN_∞
50	0.3	0.7	SPRT-O-$\text{LDC}_{0.5}$
100	0.5	0.9	DDT-NN_∞
150	0.7	095	DDT-O-$\text{LDC}_{0.5}$
	0.9		

For Winnow, α was set to 2, because good results were reported with this value [5]. Winnow was applied separately to the 4 NN models, 5 Perceptron variants and 4 O-LDC variants and the 4 detection-based classifiers. Finally Winnow was applied to all 17 classifiers.

3.3 An illustration

The example below illustrates the effect of the parameter values on the classification. Consider a two-dimensional data set as shown in figure 1. There are four equiprobable Gaussian classes, each one with identity covariance matrix. The centres for the four classes are at $x = \pm 2$, $y = \pm 2$. The training data consists of 1000 objects, 250 from each class. Another set of 1000 objects (250 per class) is generated for testing. The results reported below are calculated on the testing set.

Figure 2 shows the progression of the testing error for the different scenarios for O-LDC with $\lambda \in \{0.5, 0.7, 0.9, 0.95\}$. On each plot we also give the progression of the ensemble error where Winnow is applied to the 4 O-LDCs.

For the no-change scenario, as the learning rate increases, the error of O-LDC increases too. The reason is that the learning rate acts as a soft window. Large λ corresponds to a small window of recent objects, and the training is limited to that. On the other hand, for $\lambda = 0.5$ the training set increases progressively with the new objects, which benefits the classifier. If a single classifier in the ensemble is the best method all the time, Winnow will not be able to improve on the best method as shown in plots (a), (d), (g) and (j). Winnow manages to get a similar error rate to O-LDC$_{0.5}$, which is the best classifier in the ensemble.

When the change is gradual, the results improve as the learning rate increases as seen in plots (b), (e), (h) and (k). Winnow's error follows a pattern similar to that of O-LDC with $\lambda = 0.95$, choosing again the one best classifier in the ensemble. The peaks in the Winnow error progression signify occasional slips where Winnow briefly fails to give the maximum weight to the best classifier.

For the abrupt change, the results are more interesting because at different times, different classifiers are the best ones. In the interval [400-800] the best results are for $\lambda = 0.95$ (plot(l)), while for [800-1000] the best results are for $\lambda = 0.5$ (plot(c)). In this case Winnow follows the best classifier rather well.

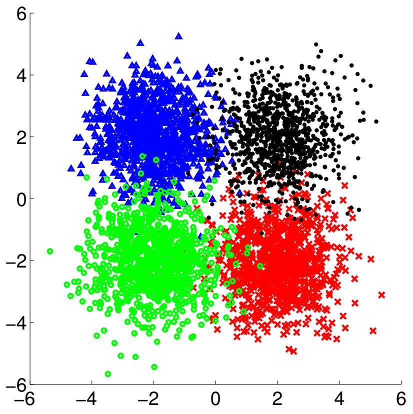


Fig. 1. 4-Gaussians dataset.

3.4 Results with the real datasets

Using Winnow, 5 ensemble designs were considered: the nearest neighbour classifiers, the Perceptrons, the O-LDC classifiers, the classifiers based on change detection and, finally, all the classifiers together. The average ranks [1] of the 17 individual classifiers and the 5 ensemble classifiers are shown in table 2.

The average ranks of the individual classifiers differ dramatically for the different scenarios. For instance, while O-LDC$_{0.5}$ is the best individual classifier for the no change scenario, it is the worst classifier for abrupt change. This variability indicates the dependency of the methods on the values of their parameter, as also illustrated in the previous subsection. In general, it is not possible to say that one method is better than another one with one exception: for the abrupt change scenario the methods based on change detection are the best.

The Perceptron is least sensitive to its parameter values; the ranks for the different η are close for all scenarios.

The Perceptron is designed for two classes. For c-class problems, $c(c-1)/2$ perceptron classifiers were trained. On the other hand, O-LDC can deal directly with multiclass datasets. Perhaps the results for O-LDC would be even better if a classifier was constructed for each pair of classes.

If we want a single method that could be used in the three scenarios, good options are DDM NN, DDM O-LDC and SPRT NN, because they have the lowest sum of ranks among the individual classifiers across the three scenarios.

The ensemble approach using Winnow outperforms the individual classifiers of all 4 types. Winnow with all the classifiers has the best average rank for

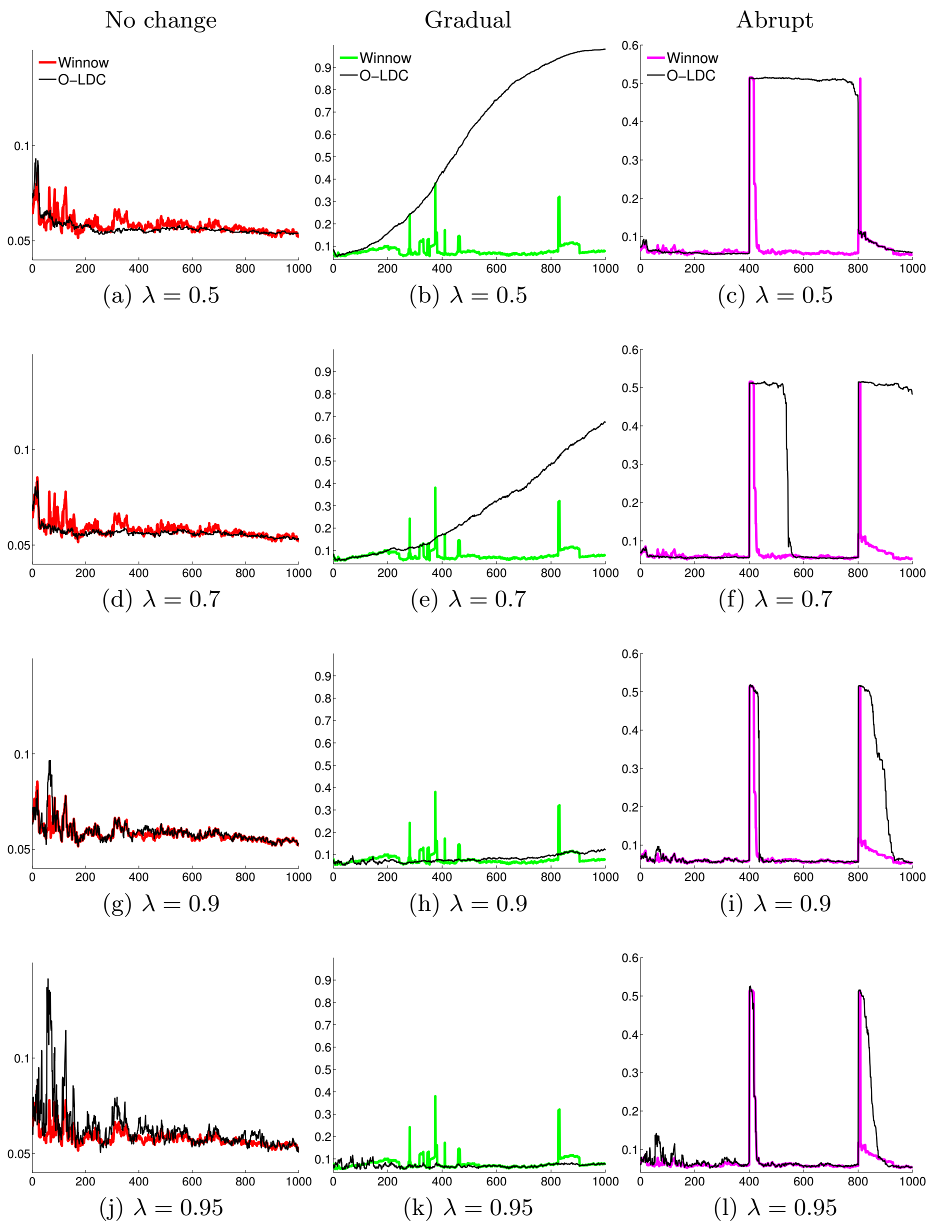


Fig. 2. Classification errors for LDC with different rates for the three scenarios. All the graphs include the errors for the method Winnow O-LDC.

Table 2. Average ranks for the 17 individual classifiers and the 5 ensembles for the three scenarios.

(a) No change

Rank	Method
5.96	Winnow All
6.96	O-LDC, rate=0.5
7.14	O-LDC, rate=0.7
7.48	NN
7.50	Winnow Detection
7.82	DDM O-LDC
8.79	Winnow O-LDC
8.82	Winnow NN
8.88	DDM NN
10.79	Winnow Perceptron
11.04	SPRT NN
11.93	SPRT O-LDC
12.54	O-LDC, rate=0.9
13.43	NN, window=150
14.32	Perceptron, rate=0.3
14.46	Perceptron, rate=0.5
14.68	Perceptron, rate=0.1
14.71	Perceptron, rate=0.9
14.82	Perceptron, rate=0.7
15.46	NN, window=100
16.82	O-LDC, rate=0.95
18.68	NN, window=50

(b) Gradual change

Rank	Method
4.43	Winnow All
6.89	Winnow O-LDC
7.25	Winnow Detection
7.79	O-LDC, rate=0.7
7.89	O-LDC, rate=0.9
8.14	NN, window=150
9.75	Winnow NN
10.93	Winnow Perceptron
11.00	NN, window=100
12.04	DDM O-LDC
12.09	DDM NN
12.23	NN
12.29	SPRT NN
12.75	SPRT O-LDC
13.57	O-LDC, rate=0.95
13.61	Perceptron, rate=0.9
14.25	Perceptron, rate=0.7
14.32	Perceptron, rate=0.5
14.46	Perceptron, rate=0.3
15.54	NN, window=50
15.86	Perceptron, rate=0.1
15.93	O-LDC, rate=0.5

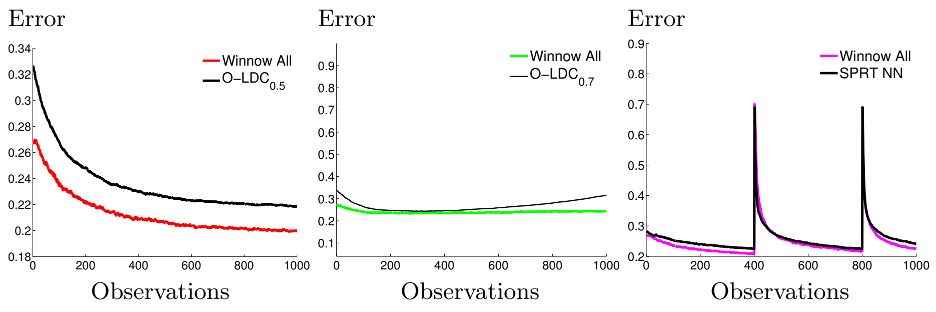
(c) Abrupt change

Rank	Method
3.04	Winnow All
3.75	Winnow Detection
5.43	SPRT NN
6.91	DDM NN
7.86	Winnow Perceptron
8.43	SPRT O-LDC
8.68	DDM O-LDC
10.07	Winnow NN
11.11	Perceptron, rate=0.3
11.14	Perceptron, rate=0.7
11.21	Perceptron, rate=0.5
11.79	Perceptron, rate=0.9
11.93	Winnow O-LDC
12.93	NN, window=50
13.18	Perceptron, rate=0.1
13.32	NN, window=100
14.43	NN, window=150
15.89	O-LDC, rate=0.95
15.93	O-LDC, rate=0.9
18.05	NN
18.14	O-LDC, rate=0.7
19.79	O-LDC, rate=0.5

the three scenarios. Winnow with the detection methods is among the 4 best methods for the three scenarios.

Figure 3 shows the error progression of Winnow All. For each scenario we plot also the error progression of the *best* individual classifier from table 2: O-LDC$_{0.5}$ (static classifier, no update) for the no-change scenario; O-LDC$_{0.7}$ for the gradual change scenario, and SPRT-NN (detect-and-update) for the abrupt change scenario. The errors are the averages across all the data sets. The graphs are meant to give us a general idea of the behaviour of the error rates for the different scenarios. Winnow All is better than the best individual classifiers for the three scenarios.

Figure 4 displays a scatterplot of the 22 classifiers in the 2-d space of the ranks for the gradual and the abrupt change scenarios. If there was an ideal classifier that was always the best in all experiments, it would be displayed as a point at (1,1). The closer the point to that ideal pair of ranks, the better the classifier, compared to the other classifiers in the experiment. The individual classifiers of the same types are shown with the same marker. The Winnow ensembles are indicated by a box. The figure reconfirms the finding that all 4 classifier models benefit from ensembles with Winnow. The best classifier is Winnow All, followed by Winnow Detection.



(a) No change (b) Gradual change (c) Abrupt change

Fig. 3. Progression of the classification error with the online observations for Winnow All and the best individual classifier from table 2.

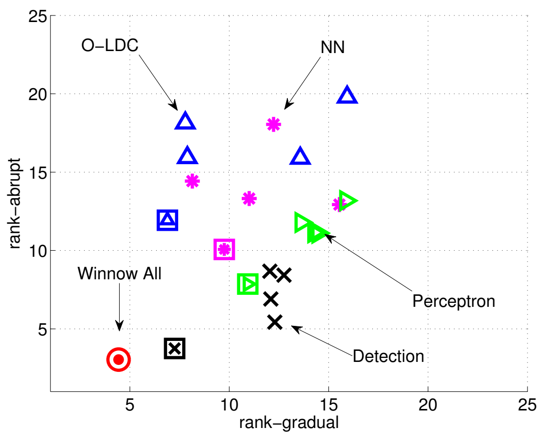


Fig. 4. Scatterplot of the 17 individual classifiers and the 5 Winnow ensembles.

We apply the statistical analyses proposed in [1] on the ranks in table 2. Friedman's ANOVA followed by calculating the Iman and Davenport statistic F_F indicate that the entries in the table are statistically significant at significance level 0.05.

To determine which classifiers are significantly different from Winnow All (the best method for the three scenarios), the two-tailed Bonferroni-Dunn test was used as a post-hoc test. A classifier is significantly different from Winnow All if its average rank is greater than the average rank of Winnow All plus the CD. The critical difference for this test is defined as $CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$ where N is the number of data sets. For $k = 22$ classifiers and $\alpha/(k-1) = 0.05/22$, $CD = 5.273$. The horizontal double lines are used in table 2 to delimit the classifiers that are significantly worse compared to Winnow All. The only classifier that is not significantly worse than Winnow All for the three scenarios is Winnow Detection.

4 Conclusion

The performance of different methods depends greatly on the type and frequencies of changes. If only one method had to be selected, we propose to choose one of the methods based on change detection, because they give the most consistent results. Our experiments showed that the combination of methods through Winnow leads to a significant improvement on the error rate of the individual methods.

Acknowledgements. This work has been supported in part by the "Junta de Castilla y León" project BU007B08 and the Spanish Ministry of Education and Science grant PR20060503.

References

1. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006.
2. Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, March 2002.
3. João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004; 17th Brazilian Symposium on Artificial Intelligence*, volume 3171/2004 of *Lecture Notes in Computer Science*, pages 286–295. Springer, 2004.
4. Ludmila I. Kuncheva. Classifier ensembles for changing environments. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *Multiple Classifier Systems, 5th International Workshop, MCS 2004*, Lecture Notes in Computer Science, pages 1–15. Springer, 2004.
5. Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, 2(4):285–318, April 1988.
6. Marlon Núñez, Raúl Fidalgo, and Rafael Morales. Learning in environments with unknown dynamics: Towards more robust concept learners. *Journal of Machine Learning Research*, 8:2595–2628, November 2007.
7. Robi Polikar, Lalita Udpa, Satish S. Udpa, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 31(4):497–508, 2001.
8. Marion R. Reynolds and Zachary G. Stoumbos. The SPRT chart for monitoring a proportion. *IIE Transactions*, 30(6):545–561, June 1998.
9. K. O. Stanley. Learning concept drift with a committee of decision trees. Technical Report AI-03-302, Computer Science Department, University of Texas-Austin, 2003.
10. Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, New York, NY, USA, 2003. ACM.