Improving Automatic Classifiers Through Interaction

Silvia Acid(✉) and Luis M. de Campos

Department of Computer Science and Artificial Intelligence, Advanced Technical School for Information Technology and Telecomunication, CITIC-UGR, University of Granada, 18071 Granada, Spain
{acid,lci}@decsai.ugr.es

Abstract. We consider a scenario where an automatic classifier has been built, but it sometimes decides to ask the correct label of an instance to an oracle, instead of accepting its own prediction. This interactive classifier only knows with certainty the labels provided by the oracle. Our proposal is to use this information to dynamically improve the behavior of the classifier, either increasing its accuracy when it is being used autonomously or reducing the number of queries to the oracle. We have tested our proposal by using twenty data sets and two adaptive classifiers from the Massive Online Analysis (MOA) open source framework for data stream mining.

Keywords: Stream mining · Online classification · Interactivity · Reject option

1 Introduction

In this paper we are concerned with the following situation: we have built a classifier for a given classification problem, that is being used to classify new instances. But the classifier can sometimes request the intervention of a human (the oracle), who gives it the correct label of the instance being considered. This may happen because the classifier is not enough sure about the label that it is proposing for this instance. If the classifier does not ask the human the correct label, there is some risk of permanently accepting a wrong classification (the decisions made by the classifier will not be further revised). For example, this may be the case of a model built to classify text documents into a set of categories, or to classify emails.

In a previous paper [1] we studied two important questions related with this interactive use of a classifier, namely how the performance of the interactive classifier should be evaluated[1] and the way in which the system decides whether or not to request a label to the oracle. In this paper we want to add a dynamic component to the interactive classifier: as we assume that the oracle always gives

[1] The evaluation should not be based only on its predictive accuracy but should also take into account the cost of the human intervention.

L. Rutkowski et al. (Eds.): ICAISC 2016, Part II, LNAI 9693, pp. 3–13, 2016.
DOI: 10.1007/978-3-319-39384-1_1

the correct answer when she is asked for the label of an instance, we could try to use this information to improve the behavior of the classifier, either increasing its accuracy when it is being used autonomously or reducing the number of queries to the oracle, or both.

It can be noticed that the proposed scenario shares many similarities with the learning with a reject option and the online learning approaches to classification.

In the scenario considered by the learning with a reject option approach [4,7], they try to build a model with the capacity to decide not to classify a given instance, because the model is not enough confident about its own prediction. In this case the instance must be manually classified by a human. However, they do not consider the possibility of updating the model as a consequence of the new information provided by the oracle for this instance (its label), together with the instance itself.

On the other hand, in the scenario proposed by the online learning [3,13] or learning from data streams [8,9] approaches, the goal is to update the learned model as soon as a new instance arrives and after we know its true class; it is assumed in this approach that soon after the prediction of the model for an instance is made, its true label will be revealed. This is only true in any problem that consists in predicting the (near) future.

The situation that we consider in this paper combines the two previous scenarios: we want to update the learned model, but we have not access to all the labels of the new instances arriving to the system but only to the labels of the examples classified by the oracle. We call this scenario interactive online classification.

The rest of the paper is organized in the following way: in Sect. 2 we give some necessary background about interactive classifiers. Section 3 describes our proposal to convert an interactive classifier into an interactive online classifier. In Sect. 4 we test our proposal by using twenty data sets and two adaptive classifiers from the Massive Online Analysis (MOA) [5] open source framework for data stream mining. Section 5 ends the paper with the conclusions and some proposals for future work.

2 Preliminaries

In order to evaluate the performance of an interactive classifier, we must take into account both its accuracy when it is used autonomously and the degree of interaction with the oracle. So, let n_c and n_w be the number of times that the classifier is used and it selects the correct and the wrong class for an instance, respectively. Also, let n_i be the number of times that the classifier asks the oracle (we assume that we have N instances to classify, so that $n_c + n_w + n_i = N$). In [1], the properties that a reasonable EMIC (Evaluation Measure for Interactive Classification) must possess were studied and two different EMIC were proposed. In this paper we will use the so called F_β measure [14]:

$$F_\beta = \frac{(1+\beta^2)n_c}{(1+\beta^2)N - n_i} \tag{1}$$

β is a parameter that controls the balance between precision and recall. In the experiments of this paper we will use $\beta = 0.5$ (which gives more importance to precision than to recall[2]).

Concerning the strategy that the interactive classifier can use to decide to ask the true class of an instance instead of classifying it, in [1] several alternatives based on the uncertainty sampling method [6,11] used in the active learning literature [12] were considered. We assume that given an instance $\mathbf{x}$, the classifier is able to obtain a posterior probability distribution of the class variable given the instance, $p(C|\mathbf{x})$, and then it predicts the most probable class, $c^*_{\mathbf{x}} = \arg\max_{c_i} p(c_i|\mathbf{x})$. Uncertainty sampling selects an instance for querying based on the level of uncertainty about its correct class. Let $\phi(p)$ be a measure of the confidence degree in the prediction obtained from the posterior probability $p(C|\mathbf{x})$. By fixing a threshold, α, on the degree of confidence, a decision rule according to which the system will decide whether or not to query the oracle can be established:

$$\begin{array}{l} \text{if } \phi(p) < \alpha \text{ then ask the oracle} \\ \qquad\qquad\quad \text{else classify the instance} \end{array} \tag{2}$$

In [1] several confidence measures were studied. In this paper we will use the measure of Maximum probability $\phi_m(p)$:

$$\phi_m(p) = \max_{c_i} p(c_i|\mathbf{x}) = p(c^*_{\mathbf{x}}|\mathbf{x}) \tag{3}$$

$\phi_m(p)$ varies in the range $[\frac{1}{m}, 1]$, where m is the number of possible classes. $\phi_m(p)=\frac{1}{m}$ represents absolute lack of confidence (because in this case $p(c_i|\mathbf{x})=\frac{1}{m}$ for all c_i), whereas $\phi_m(p) = 1$ means total confidence.

It is obvious that the threshold α used in combination with the confidence measure is a key parameter for the interactive classifier's decision strategy: the greater α the more interactive the classifier. If α is set to its maximum value (1 in the case of the confidence measure being considered) then the classifier is useless, it always queries; on the other extreme, if α is set to its minimum value, we obtain a non-interactive classifiers that never queries. Moreover, the same α may behave very differently depending on the specific algorithm used to build the base classifier and even the classification problem being considered. For these reasons it is very convenient to try to estimate a reasonable value for α taking into account both the problem to be solved and the base classifier to be used.

We will use the following strategy, proposed in [1]: we rank the training instances used to train the base classifier in decreasing order of their confidence measure and then look for the threshold which optimizes the selected performance measure (F_β in our case). We do it by tentatively making the threshold equal to the confidence measure of each training instance and computing the corresponding performance measure (for all the instances in the training set). We then select the threshold producing the best result. An example of this computation can be seen in Table 1.

[2] In [1], it was shown that with values of β lesser than or equal to 0.5, the interactive classifiers systematically outperform their non-interactive counterparts.

Table 1. Example of the strategy used to select the threshold α. Column ϕ represents the value of the confidence measure for each of the seven training instances; column c/w means whether the label predicted by the classifier coincides with the true label (c) or not (w); columns n_c, n_w and n_i count the number of correct and wrong classifications, and interactions, respectively, if we fix the value α to the corresponding ϕ value; column $F_{0.5}$ gives the values of the performance measure. In this case, in bold is marked the highest value for $F_{0.5}$, as well as its corresponding value for ϕ; thus, the threshold that optimizes the EMIC, is $\alpha = 0.74$.

ϕ	c/w	n_c	n_w	n_i	$F_{0.5}$
1.00	c	1	0	6	0.455
0.95	c	2	0	5	0.667
0.90	w	2	1	4	0.526
0.75	c	3	1	3	0.652
0.74	c	4	1	2	**0.741**
0.60	w	4	2	1	0.645
0.57	c	5	2	0	0.714

3 Interactive Online Classifiers

As mentioned in the introduction, our goal is to transform our interactive but static classifier into an online (dynamic) classifier that can improve its behavior from the information provided by the oracle when the classifier decides to query her instead of classifying an instance.

Obviously, any non-interactive classifier can be transformed into an interactive classifier, provided that it is able to give a numerical output for each possible category of the instance to be classified (which can be transformed into a posterior probability distribution where we apply the confidence measure and the decision rule in Eq. (2)). We will assume that the classifier being considered can work online, that is to say it can modify itself when a new instance together with its correct label are provided.

However, in the case of an online (non-interactive) classifier, it has access to all the new instances and their correct labels. Therefore, the new instances used to update the classifier are extracted from the same underlying distribution than the ones used to initially train it[3]. But it is not necessarily the case of the interactive online classifier, because the only instances that can be used to update it are those where the classifier is unsure, and this may introduce a bias in the classifier, perhaps decreasing its performance.

Therefore, what we want to evaluate is whether this is or not the case, i.e. whether the interactive online classifier can improve the performance of the initially trained (static) interactive classifier.

[3] In this paper we assume that the process is stationary and no concept drift occurs. The case of non-stationary problems, where a concept drift can modify the probability distribution, will be considered in future research.

Table 2. In the example in Table 1, we add a new instance with ϕ value 0.53, which is correctly classified. In bold are the new highest value for $F_{0.5}$ and the new best threshold, that decreases to $\alpha = 0.53$. If the new instance were classified incorrectly, then the value of $F_{0.5}$ for the confidence measure of the new instance would be 0.625 instead of 0.750 and the best threshold would still be $\alpha = 0.74$ as it was before, in italic.

ϕ	c/w	n_c	n_w	n_i	$F_{0.5}$
1.00	c	1	0	7	0.417
0.95	c	2	0	6	0.625
0.90	w	2	1	5	0.500
0.75	c	3	1	4	0.625
0.74	c	4	1	3	*0.714*
0.60	w	4	2	2	0.625
0.57	c	5	2	1	0.694
0.53	c	6	2	0	**0.750**

Another important point concerns the threshold α used by the decision rule in Eq. (2). As this threshold is computed from the initial training set, perhaps we should dynamically adapt it as new instances are used to update the classifier. Our proposal is to recompute this threshold each time a new instance (labeled by the oracle) is available, following the method explained in Sect. 2. It can be proven[4] that if we update the threshold in this way, it can only either remain equal or decrease. This is in agreement with our intuition that if we use more information to train a classifier, it will gain more confidence in its own predictions (thus decreasing α). An example of the re-computation of the threshold can be seen in Table 2.

4 Experimental Framework

In this section we are going to experimentally study the behavior of the interactive online classifiers. For all the experiments carried out we have used the MOA software [5], an open-source workbench for online learning from data streams, closely related to WEKA. We have selected two online base classifiers (to transform them into interactive and interactive online classifiers) and 20 different data streams, all easily available in MOA.

4.1 From Non-interactive to Online Interactive Classifiers

The first interactive classifier that we are going to test, proposed in [1], called *Theshold+NoLearn* (2TNL for short), comes from using any non-interactive base classifier to predict the class for a new arriving instance and, when this prediction

[4] By lack of space, we omit the proof of this result.

has low confidence, it will query the oracle for the true label. The degree of confidence will be measured using Eq. (3) and the uncertainty will be considered by using the decision rule in Eq. (2). The online base classifier is learned from the training data stream, and the threshold, α, is then estimated in the way described in Sect. 2 (see an example in Table 1). The threshold and the classifier, even though interactive, will remain unchanged for any further instances after the training. Although it does not learn any more, it will restrict itself to classify instances having a relatively high degree of confidence (trying to increase the EMIC measure in Eq. (1)).

As it has already been emphasized, the classifier only knows for sure the instances labeled by the oracle. Those labeled instances queried to the oracle have the same informative quality as the training set[5], and it could be a waste to miss some kind of learning. Thus, we propose a new interactive online classifier *Theshold+Learn* (3TLe for short), that adds any new acquired knowledge to the model by means of the own online classifier used in the training. Therefore, the new labeled instances are used to update the classifier. The threshold, the confidence indicator, remains as it was estimated from the training data stream, as in the previous interactive classifier.

Nevertheless, those instances labeled by the oracle could also be used to readjust the initial threshold estimated from the original training set. Thus, we propose another interactive online classifier called *DynamicThreshold* (4Dyn). At any time this classifier will evaluate the confidence degree of the current instance, comparing it to the threshold (as in the previous cases), but once a query is submitted to the oracle the threshold can be updated (as explained in Sect. 3, see an example in Table 2). The new value of the threshold will be used by the interactive online classifier with the next instances, until a new query to the oracle occurs, in this case the threshold may be updated again.

As the baseline for our experiments, we will consider the non-interactive base classifier that never queries, denoted as 1NQ.

4.2 The Data Streams and the Base Classifiers

The two base online classifiers used are *NaiveBayes* and the *HoeffdingTreeNB* [10]. The second one is an incremental decision tree for streaming data with NaiveBayes classification at leaves.

For the experiment we have used 20 data streams created by means of four different streams generators available in MOA.

The *Agrawal* generator [2], simulates the granting of credit by means of 10 different functions or rules, that take into account 9 different attributes, e.g. academic degree, salary, value of the loan, etc. 3 of them are categorical and 6 numerical. *Hyperplane* generates data for a problem of predicting the class of a rotating hyperplane. We produce 3 different problems. For example, *hyperpl01* contains 10 continuous attributes, and the class attribute has two categories. The *sea* generator simulates 4 different concepts by means of rules based on 3

[5] The new instances are supposed to come from the same data distribution.

continuous attributes. Finally, we used the *Random tree* generator, that produces concepts that theoretically should favor decision tree learners. It constructs decision tree randomly according to several parameter settings, as number of classes, number of categorical and numerical attributes. The tree is used to determine the class label for new examples generated by assigning an uniformly distributed random value to the attributes. The attribute values as well as the class label determined via the tree become part of the data stream. We used 3 different concepts by using 2 simple and a complex random tree. More details of the data sets can be seen in Table 3.

Table 3. A summary of the 20 data streams used for the training and the evaluation of the classifiers. The total length of each data set is set to 101,000. The first thousand are used for the initial training process.

Name	Categorical	Numeric	Class	Name	Categorical	Numeric	Class
agrawal01	3	6	2	*hyperpl01*	0	10	2
agrawal02	3	6	2	*hyperpl02*	0	10	3
agrawal03	3	6	2	*hyperpl03*	0	50	5
agrawal04	3	6	2	*sea01*	0	3	2
agrawal05	3	6	2	*sea02*	0	3	2
agrawal06	3	6	2	*sea03*	0	3	2
agrawal07	3	6	2	*sea04*	0	3	2
agrawal08	3	6	2	*tree01*	5	5	2
agrawal09	3	6	2	*tree02*	5	5	3
agrawal10	3	6	2	*tree03*	10	40	5

By using any of the described generators we can obtain streams of any length, ready as input for the online classifiers. We used 1,000 instances for each of the described problems, as training data stream. Additionally we used 100,000 instances for testing and evaluating every classifier. Given the stochastic nature of the generators, the data were generated once and then saved. Thus, in order to perform a fair comparison, all the classifiers work on exactly the same data sets, so they begin from an identical starting point.

When evaluating online classifiers on a stream, every instance is used first for testing and after for updating the model. But, as the interactive classifiers have not access to the true label unless it is requested, the updating will be considered, only for the interactive online classifiers 3TLe and 4Dyn, when they ask the oracle. The main metric we compute during the evaluation process is the EMIC value in Eq. (1). Additionally we also compute two other metrics, Accuracy and Interactivity. Accuracy is the success rate over the instances actually classified autonomously, $\frac{n_c}{n_c+n_w}$[6]. Interactivity is the number of labels requested by the classifier, expressed in percentage, $100 * \frac{n_i}{N}$. This shows the degree of autonomy of the classifier.

[6] Without taking into account those instances labeled by the oracle, n_i.

Table 4. Average of the metrics used for the evaluation of the interactive classifiers based on *NaiveBayes*, using 100,000 instances. The average is computed over the 20 data streams.

	EMIC	Interactivity	Accuracy
1NQ	0.824	0.00	0.824
2TNL	0.835	18.74	0.876
3TLe	0.824	36.12	0.945
4Dyn	0.882	1.37	0.884

Table 5. Average of the metrics used for the evaluation of the interactive classifiers based on *HoeffdingTreeNB*, using 100,000 instances. The average is computed over the 20 data streams.

	EMIC	Interactivity	Accuracy
1NQ	0.820	0.00	0.820
2TNL	0.829	16.16	0.864
3TLe	0.711	2.51	0.714
4Dyn	0.837	1.13	0.839

A summary of the results of the comparative study of the interactive classifiers is displayed in Tables 4 and 5. We have made a separate study for each of the base classifiers *NaiveBayes* and *HoeffdingTreeNB* respectively. The values displayed are an average over the 20 data streams.

Regarding Table 4, the non-interactive classifier *NaiveBayes* 1NQ reaches an average Accuracy of 0.824. Due to its lack of interactivity, the EMIC value coincides with the Accuracy. 2TNL requests to the oracle the true label of 18.7 % of the 100,000 instances. This increases slightly Accuracy and EMIC, thus it avoids to commit some classification errors. 3TLe increases remarkably the Accuracy, reaching 0.945, at the expense of a high percentage of interactivity, 36.12 %. This means that, when the classifier is sure enough, its classification is mainly correct, but it asks for too much labeled instances, as many of them would be correctly classified otherwise. That penalty is reflected in the EMIC value, which is equal to the 1NQ's EMIC. This may be due to a threshold too high. On the other hand, the classifier 4Dyn performs very well. Its Accuracy is better than that of 1NQ and 2TNL, i.e., when it classifies it makes less errors. Moreover it pays a low price for this improvement, with an Interactivity of only 1.4 %. As a consequence, the EMIC value is much greater than the one of its competitors. As we will illustrate next, the queries to the oracle are posed more frequently at the beginning of the process, and after a while the classifier will continue almost autonomously.

The results in Table 5 are similar in many aspects to those in Table 4. 4Dyn is still the best classifier, although the differences with respect to the other classifiers are smaller, except with 3TLe, which performs quite poorly (with low

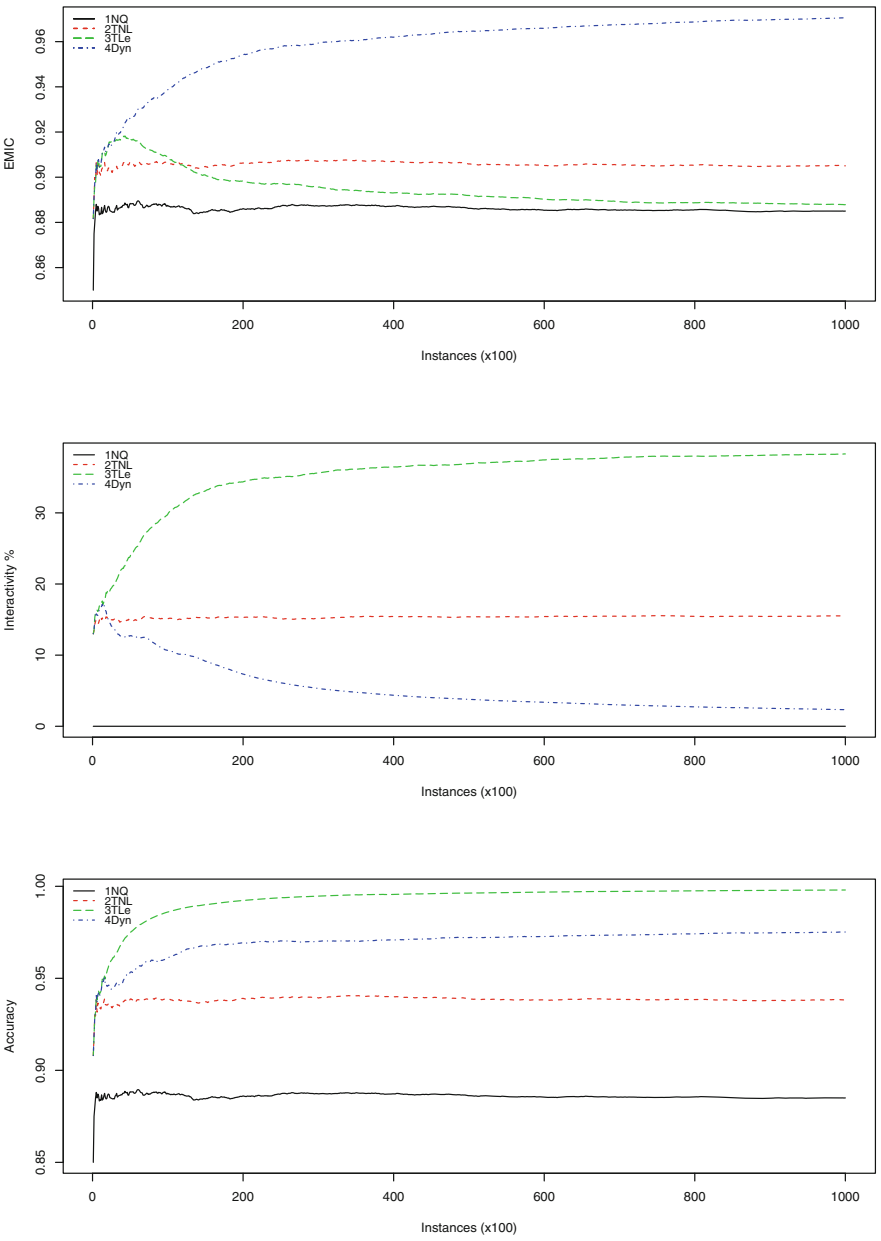

Fig. 1. Progress of the EMIC, Interactivity and Accuracy metrics on the data stream *agrawal07*, for each of the interactive classifier considered built over the base classifier *NaiveBayes* at the same points (steps of 100 samples) of the overall process. We can appreciate changes in the degree of the interactivity for $4Dyn$ due to updating of α to lower values, that have impact on the EMIC curve, as this measure is a compound of the others.

Interactivity but also low Accuracy, leading also to low EMIC). Another difference between the two cases is that for *NaiveBayes* all the differences in performance between classifiers (focusing on the EMIC metric) are statistically significant, according to a Friedman test and the subsequent post hoc tests (significance level 0.05). The only exception is with 1NQ and 3TLe, where no significant differences were found. However, in the case of *HoeffdingTreeNB*, all the differences are significant except for the comparison between 2TNL and 4Dyn.

In order to illustrate the behavior of every interactive classifier in the study, we chose to display the evaluating process with the *NaiveBayes* base classifier for the specific problem *agrawal07*. Through the graphics included in Fig. 1 we can observe the progress of the metrics along all the sequence of testing stream, the 100.000 samples. The three graphics display the EMIC, the Interactivity and the Accuracy at the same points of the overall process.

Firstly, 1NQ starts from EMIC/Accuracy of 0.850, after some oscillations due to the variability on few data, it reaches a plain around 20,000 samples to end with 0.885 the complete testing stream. Naturally, the value for interactivity is permanently 0. Secondly, the curve described by 2TNL is quite similar to 1NQ, except for starting from a higher EMIC value of 0.882, because in the first 100 samples it requests 13 samples to the oracle. This percentage remains around 15 % since the beginning. The Accuracy of 2TNL is superior to that of 1NQ, that means that many requested samples would be classification errors. Thirdly, as described previously, 3TLe get bad values for EMIC and Interactivity. It has the same beginning as 2TNL, but progressively the Interactivity increases until reaching 38.2 %, at the same time the EMIC suffers a progressive decrease, reaching the baseline of 1NQ, with value 0.888. On other problems, the curve of 3TLe for the EMIC even can go underneath. So, it deteriorates clearly the interactive system. Finally, 4Dyn starts as the previous ones, from a value of 0.882 for EMIC. On the first 100 samples it requests 13 samples, and continues increasing its Interactivity until reaching 16 % after the first 1800 samples. Then it readjusts the threshold, and the Interactivity drops progressively to a value of 2.3 %, because it is less and less unsure for its predictions.

5 Concluding Remarks

In this paper we have proposed a method for adding interactivity to the behavior of automatic classifiers that can work online, but have not access to the labels of the incoming instances unless they ask them to an oracle. The decision about asking or not is based on a threshold relative to the confidence degree of the classifier in its own prediction.

We have tested our proposals using a variety of data streams generated using the MOA framework and two online base classifiers. The performance of the interactive online classifier that does not update its initial threshold is rather poor (probably due to the bias induced by the new instances used to update the classifier, which are always near the decision frontier). However, the interactive online classifier that recomputes its threshold performs excellently, both

increasing the accuracy of the classifier when it is used autonomously and decreasing the number of queries to the oracle.

For future work, we plan to test our methods with other incremental/online classifiers beyond *NaiveBayes* and *HoeffdingTreeNB*. We also want to design alternative rules to decide whether the classifier should ask the oracle, for example allowing (in an random way) to occasionally ask when the classifier is quite sure of its prediction, in order to prevent the possible bias. Finally, we plan to study the problem of incorporating interactivity in the presence of concept drift.

Acknowledgements. This work has been funded by the Spanish Ministry of Economy and Competitiveness under the project TIN2013-42741-P and the European Regional Development Fund (ERDF-FEDER).

References

1. Acid, S., de Campos, L.M., Fernández, M.: Evaluation methods and strategies for the interactive use of classifiers. Int. J. Hum.-Comput. Stud. **70**(5), 321–331 (2012)
2. Agrawal, R., Imielinski, T., Swami, A.: Database mining: a performance perspective. IEEE Trans. Knowl. Data Eng. **5**(6), 914–925 (1993)
3. Auer, P., Cesa-Bianchi, N., Gentile, C.: Adaptive and self-confident on-line learning algorithms. J. Comput. Syst. Sci. **64**, 48–75 (2002)
4. Barlet, P.L., Wegkamp, M.H.: Classification with a reject option using a hinge loss. J. Mach. Learn. Res. **9**, 1823–1840 (2008)
5. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
6. Fu, Y., Zhu, X., Li, B.: A survey on instance selection for active lerning. Knowl. Inf. Syst. **35**, 249–283 (2013)
7. Fumera, G., Roli, F., Giacinto, G.: Reject option with multiple thresholds. Pattern Recogn. **33**(12), 2099–2101 (2000)
8. Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall/CRC, Boca Raton (2010)
9. Gama, J.: A survey on learning from data streams: current and future trends. Prog. Artif. Intell. **1**, 45–55 (2012)
10. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD-01, pp. 97–106. ACM Press (2001)
11. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: Proceedings of the ACM SIGIR Conference, pp. 3–12 (1994)
12. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2009)
13. Shalev-Shwartz, S.: Online learning and online convex optimization. Found. Trends Mach. Learn. **4**(2), 107–194 (2012)
14. van Rijsbergen, C.J.: Foundation of evaluation. J. Documentation **30**(4), 365–373 (1974)