On-line Bagging Negative Correlation Learning

L. Minku and X. Yao

***Abstract*— Negative Correlation Learning (NCL) has been showing to outperform other ensemble learning approaches in off-line mode. A key point to the success of NCL is that the learning of an ensemble member is influenced by the learning of the others, directly encouraging diversity. However, when applied to on-line learning, NCL presents the problem that part of the diversity has to be built *a priori*, as the same sequence of training data is sent to all the ensemble members. In this way, the choice of the base models to be used is limited and the use of more adequate neural network models for the problem to be solved may be not possible. This paper proposes a new method to perform on-line learning based on NCL and On-line Bagging. The method directly encourages diversity, as NCL, but sends a different sequence of training data to each one of the base models in an on-line bagging way. So, it allows the use of deterministic base models such as Evolving Fuzzy Neural Networks (EFuNNs), which are specifically designed to perform on-line learning. Experiments show that on-line bagging NCL using EFuNNs have better accuracy than NCL applied to on-line learning using on-line Multi-Layer Perceptrons (MLPs) in 4 out of 5 classification databases. Besides, on-line bagging NCL using EFuNNs manage to attain similar accuracy to NCL using *off-line* MLPs.**

I. Introduction

On-line learning has been showing to be very useful for a growing number of applications in which training data is available continuously in time and/or there are time and space constraints. Examples of such applications are industrial process control [1], computer security, intelligent user interfaces and market-basket analysis [2], information filtering [3], prediction of conditional branch outcomes in microprocessors [4] and RoboCup [5].

On-line learning algorithms process each training instance once "on arrival" without the need for storage and reprocessing, and maintain a current hypothesis that reflects all the training instances so far [6]. In this way, the learning algorithms take as input a single labelled training instance as well as a hypothesis and output an updated hypothesis [4].

Recently, ensembles of classifiers have been successfully used to improve the accuracy of single classifiers in on-line learning [7], [6], [4]. Negative Correlation Learning (NCL) [8], [9] is a ensemble learning method that has been showing to outperform other ensemble learning methods in off-line mode [10], [11], [12], including bagging [13] and boosting [14]. A key point to the success of NCL is that the learning of an ensemble member is influenced by the learning of the others, directly encouraging diversity. This key point makes NCL a potentially powerful approach to on-line learning.

However, when applied to on-line learning, NCL presents the problem that part of the diversity has to be built *a priori*, as the same sequence of training data is sent to all the ensemble members. In this way, the choice of the base models to be used is limited and the use of more adequate neural network models for the problem to be solved may be not possible.

This paper proposes a new method to perform on-line learning based on NCL and On-line Bagging [6]. The method directly encourages diversity, as NCL, but sends a different sequence of training data to each one of the base models in an on-line bagging way. So, it allows the use of deterministic base models such as Evolving Fuzzy Neural Networks (EFuNNs) [15], which are specifically designed to perform on-line learning.

Experiments show the importance of a wider range of base model choices and reveal that on-line bagging NCL using EFuNNs have better (lower) classification error than NCL applied to on-line learning using on-line Multi-Layer Perceptrons (MLPs) in 4 out of 5 classification databases from the UCI Machine Learning Repository [16]. Besides, on-line bagging NCL using EFuNNs manage to attain similar classification error to NCL using off-line MLPs, even being able to use each training instance only once, while off-line MLPs can use the whole training set a certain number of epochs.

This paper is further organized as follows: section II presents related work, section III presents the proposed approach, section IV presents the experiments done to show the importance of the proposed method and section V presents the conclusions of the work.

II. Related Work

Recently, ensemble learning methods have been showing to outperform single classifiers in on-line learning. Section II-A presents some successful on-line ensemble classification methods existent in the literature and explain how the method proposed in this paper can overcome their weaknesses. Section II-B contains some comments about the base models used in this work.

A. On-line Ensemble Learning

Some of the on-line ensemble learning methods existent in the literature send the same sequence of data to all the ensemble members. These methods present the problem that part (or all) of the model diversity has to be built *a priori* rather than emerging from data itself. In many cases, these methods consist of different ways of combining the ensemble

Fernanda L. Minku and Xin Yao are with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK (emails: {F.L.Minku, X.Yao}@cs.bham.ac.uk).

This work was supported by the United Kingdom Government and the School of Computer Science of the University of Birmingham in the form of an Overseas Research Students Award (ORSAS) and a School Research Scholarship.

members [17], [18] or finding alternatives to create different ensemble members when the same sequence of training data is sent to all of them [19], [20]. These methods, except NCL, perform independent training of the ensemble members.

Nevertheless, ensemble learning methods which directly encourage diversity by considering the interaction among all ensemble members during the learning, such as NCL, have been showing to outperform other ensemble learning approaches in off-line mode [10], [11], [21]. So, ensemble learning methods in which the ensemble members are not trained independently have a potential advantage over methods which perform independent training in on-line mode.

Some other on-line ensemble learning methods existent in the literature do not send the same sequence of training data to all the ensemble members. Instead, they consist of different ways of sending training data to the ensembles members, so that diversity does not have to be partly or completely designed *a priori*. Most of these methods are on-line versions of bagging [13] and boosting [14].

An example is Modified Adaptive Boosting (MadaBoost) [22]. This method is a modification of AdaBoost [14] which can be used in the filtering framework without having extremely high execution time. This modification bounds the weight that is attributed to the training instances, reducing the time necessary for the filter to choose an example to be used in the learning process. The weighting update scheme is also slightly modified, in order to obtain a formal proof of convergence to MadaBoost. This method can reduce considerably the execution time of AdaBoost in the filtering framework and, at the same time, obtain similar or better generalization.

Two notable on-line ensemble learning methods for classification are On-line Bagging and On-line Boosting [6]. On-line bagging is based on the fact that, when the number of training instances tends to infinite in off-line bagging, each ensemble member i contains K_i copies of each of the original training instances, where the distribution of K_i tends to a $Poisson(1)$ distribution. So, in on-line bagging, whenever a training instance is available, it is presented K_i times to each ensemble member i, where K_i is drawn from a $Poisson(1)$ distribution. The classification is done by unweighted majority vote, as it is done in Bagging.

On-line boosting works in a similar way to on-line bagging, but it uses a $Poisson(\gamma_d)$ distribution. The parameter γ_d associated to an instance d is increased when presented to the next ensemble member if the current ensemble member misclassifies the instance. Otherwise, it is decreased. In this way, γ_d has the same role as the weight of the instance d in AdaBoost. The on-line boosting algorithm gives the instances misclassified by one stage (one classifier) half the total weight in the next stage and the correctly classified instances are given the remaining half of the weight. To use the system, the classification made by the whole ensemble is by weighted majority vote, with weights based on the accuracy of the ensemble members.

Both on-line bagging and boosting are able to get better generalization than a single classifier when their off-line correspondent algorithms also can. Very similar on-line bagging and boosting methods were also proposed in [4].

The on-line approaches based on bagging and boosting present some of the problems that bagging and boosting present. For example, diversity among the ensemble members is very important to produce successful ensembles [23]. However, as on-line bagging is an approximation of bagging, the ensemble members are also created independently. So, there is no warranty that they will be enough diverse to produce an ensemble with good accuracy.

The ensemble members of the on-line methods based on boosting have some influence from the others. However, the first ensemble members trained are not influenced by the last ones and diversity is not directly encouraged. Besides, boosting algorithms tend to overfit training instances [24]. According to [25], this happens because the boosting method to update the probabilities associated to each training instance may over-emphasize noisy training instances. Besides, the classifiers created by boosting are combined using weighted voting. Previous work [26] has shown that optimizing the combining weights can lead to overfitting, while an unweighted voting scheme is generally robust to overfitting.

NCL is an ensemble learning method that can be applied to on-line learning. In this method, the training of an ensemble member is influenced by the others, directly encouraging diversity through the use of a penalty correlation term in the error function of the base model learning algorithm. In off-line mode, ensembles which directly encourage diversity considering the interaction among all the ensemble members during the learning [10], [11], [21] have been showing to outperform other ensemble learning methods such as bagging and boosting. So, this is a potentially powerful advantage of NCL over the other on-line ensemble methods existent in the literature. The negative correlation among the ensemble members and the use of unweighted voting also makes NCL robust in relation to overfitting.

Recently, NCL has been applied to incremental learning[1] [27], showing to be a promising approach in this area. However, when applied to on-line learning, NCL presents the problem that part of the diversity has to be built *a priori*, as the same sequence of training data is sent to all the ensemble members. In this way, the choice of the base models to be used is limited and the use of more adequate neural network models for the problem to be solved may be not possible. The method proposed in this paper can overcome this problem, taking advantage of the NCL features to overcome the problems of the other methods existent in the literature and allowing the choice of a wider range of base models, including deterministic classifiers.

[1]We consider that an incremental learning algorithm can learn training data gathered in several batches, instead of learning each example separately.

B. Base Models

In this work we use two different base models. One of them is Multi-Layer Perceptrons (MLPs) [28] and the other is Evolving Fuzzy Neural Networks (EFuNNs) [15].

The algorithm used to train the MLPs is the stochastic back-propagation [29]. Some authors refer to this algorithm as on-line back-propagation. It is important not to confuse the term on-line learning used in this work with the term on-line learning related to the back-propagation algorithm. In the later, the term is used to indicate that the weights are updated right after the presentation of each training instance. However, the whole training set can be presented several times to the neural network. In order to avoid confusion, we will not refer to the stochastic back-propagation algorithm as on-line back-propagation in this work.

Stochastic back-propagation can be applied to perform both on-line and off-line learning. In on-line learning, each training instance has to be processed only once and then discarded. So, it is possible to use only 1 epoch for learning. We will refer to the stochastic back-propagation which uses only 1 epoch as on-line stochastic back-propagation. It would be possible to present each training instances a certain number of times "on-arrival", as it was done by [30]. However, as it is commented in section IV-A, this does not improve the resulting ensemble's classification error.

EFuNNs are neural network models specifically designed to perform on-line learning. They are fast (only one pass through the training examples is necessary), local and constructive. Local and constructive learning is very important to avoid catastrophic forgetting, which problem known for making on-line learning more challenging.

EFuNNs have a five-layer architecture. The first layer represents the input vector, the second represents the fuzzy quantification of the input vector, the third represents the associations between fuzzy input space and fuzzy output space, the fourth represents the fuzzy quantification of the output vector and the fifth represents the output vector.

Learning occurs at the rule nodes layer. Each node *rj* of this layer is represented by two vectors of connection weights (*W1(rj)* and *W2(rj)*). *W1* represents the coordinates of the nodes in the fuzzy input space and it is adjusted through unsupervised learning. *W2* represents the coordinates of the nodes in the fuzzy output space and it is adjusted through supervised learning. The learning rules are the following:

- $W1(r_j) = W1(r_j) + lr1(r_j) * (x_f(d) - W1(r_j))$
- $W2(r_j) = W2(r_j) + lr2(r_j) * (t_f(d) - A2) * A1(r_j)$

where: $x_f(d)$ and $t_f(d)$ are the fuzzy input and fuzzy output vectors of the training pattern d; $lr1(r_j)$ and $lr2(r_j)$ are the learning rates for the $W1$ and $W2$ weights of the node r_j at a particular time during the learning; $A2$ is the fuzzy output activation vector and $A1(r_j)$ is the activation value of the rule node r_j. The learning rate of a node can be the inverse of the number of training patterns accommodated so far by that node.

The EFuNN learning algorithm is briefly described below. For more details, it is recommended to read [15].

Algorithm 1 EFuNN Learning Algorithm

Inputs: current EFuNN, training pattern d, number of training patterns presented so far and training parameters (number of membership functions; type of membership functions; initial sensitivity threshold S of the nodes, which is also used to determine the initial radius of the receptive field of a node r_j when it is created ($R(r_j) = 1 - S$); error threshold E; aggregation parameter $Nagg$; pruning parameters OLD and Pr; *m-of-n* value, which is the number of highest activation nodes used in the learning; maximum radius of the receptive field $Mrad$; rule extraction thresholds $T1$ and $T2$).
Output: updated EFuNN.

1) If this is the first learning of EFuNN, set the first rule node r_0 to memorize d: $W1(r_0) = x_f(d)$ and $W2(r_0) = t_f(d)$.
2) Else
   - 2.1 Calculate the activations $A1$ of all rule nodes, e.g., $A1 = 1 - D(W1(r_j), x_f(d))$, where D is a distance measure.
   - 2.2 Select the rule node r_k that has the smallest distance $D(W1(r_k), x_f(d))$ and that has activation $A1(r_k) >= S(r_k)$. In the case of *m-of-n* learning, select *m* nodes instead of just one node.
   - 2.3 If this node does not exist, create a new rule node.
   - 2.4 Else
     - 2.4.1 Determine the activation $A2$ of the output layer and the normalized output error $Err = subabs(t(d), F_{ef})/Nout$, where $t(d)$ is the desired output, F_{ef} is the obtained output and $Nout$ is the number of nodes of the output layer.
     - 2.4.2 If $Err > E$, create a new rule node.
     - 2.4.3 Else, apply the learning rules to $W1(r_k)$ and $W2(r_k)$ (in the case of *m-of-n* learning, the rules are applied to the *m* rule nodes).
   - 2.5 Apply aggregation procedure after the presentation of $Nagg$ patterns.
   - 2.6 Update the parameters $S(r_k)$, $R(r_k)$, $Age(r_k)$ and $TA(r_k)$. $TA(r_k)$ can be, for example, the sum of the activations $A1$ obtained for all examples that r_k accommodates.
   - 2.7 Prune rule nodes according to OLD and Pr.
   - 2.8 Extract rules according to $T1$ and $T2$.

III. On-line Bagging NCL

This section proposes a new on-line ensemble learning method called on-line bagging NCL. As it was discussed in section II-A, this method is able to take advantage of the NCL strong points and, at the same time, overcome the NCL problem that part of the diversity has to be built *a priori*, which limits the choices of base models to be used.

On-line bagging NCL uses a penalty term in the error function to be optimized by the neural network learning

algorithm. In the same way as in NCL, the penalty term is used to penalize positive correlation of errors from different neural networks, i.e. , to encourage negative correlation between the error of an ensemble member and the error of the rest of the ensemble.

Let $F(d)$ be the arithmetic average of the ensemble member outputs for the training pattern d:

$$F(d) = \frac{1}{M}\sum_{i=1}^{M} F_i(d) \ , \tag{1}$$

where $F_i(d)$ is the output of the *i*th individual neural network on the training pattern d.

If the error function used by the base models is the mean squared error, the error E_i for the *i*th ensemble member on the training pattern d can be adapted in the following way to accommodate the penalty term:

$$E_i(d) = \frac{1}{2}(F_i(d) - t(d))^2 + \gamma p_i(d) \ , \tag{2}$$

where $t(d)$ is the target output of the training example d, p_i is the correlation penalty function and γ is a parameter used to adjust the strength of the penalty. The penalty function p_i may use the following equation:

$$p_i(d) = (F_i(d) - F(d))\sum_{i\neq j}(F_j(d) - F(d)) \ . \tag{3}$$

The partial derivative of $E_i(d)$ with respect to the output of the network *i* on the *d*th training pattern is:

$$\frac{\partial E_i(d)}{\partial F_i(d)} = F_i(d) - t(d) - \gamma\left[2\left(1 - \frac{1}{M}\right)(F_i(d) - F(d))\right] \ . \tag{4}$$

This partial derivative can be used to perform the weight adjustments of the neural networks that belong to the ensemble.

Consider an ensemble composed by on-line base models which use error functions adapted to the use the penalty function. Similarly to on-line bagging, on-line bagging NCL presents each training instance k_i times to the on-line learning algorithm, where k_i is drawn from a $Poisson(1)$ distribution. However, before the learning of a training example d, the arithmetic average of the outputs of the ensemble members on d has to be calculated, in order to be used by the on-line learning algorithms of the ensemble members. The algorithm is presented below:

Algorithm 2 On-line Bagging NCL

Inputs: ensemble of neural networks **h**; training example d; strength parameter γ; on-line learning algorithm L_o which uses an error function adapted to the use of p_i.
Output: updated ensemble of neural networks **h**.

1) Calculate $F(d)$.
2) For each ensemble member h_i, do:
   2.1 Set k_i according to Poisson(1).
   2.2 Do k_i times:
      2.2.1 $h_i = L_o(h_i, F(d), \gamma, d)$.

IV. Experiments

This section presents the experiments done with NCL and On-line Bagging NCL in order to validate and check the importance of the new method. Section IV-A presents the databases and the experimental setup used. Section IV-B shows that NCL with on-line MLPs is not suitable to perform on-line learning. Section IV-C shows that on-line bagging NCL is able to outperform NCL using on-line MLPs when EFuNNs are used as the base models and shows that MLPs are not so suitable to on-line learning as other models. Besides, this section shows that on-line bagging NCL using EFuNNs can get similar classification error to NCL using off-line MLPs, emphasizing even more the importance of having a wider range of choices for the base model. The results are presented using classification error, instead of accuracy, in order to provide a better visualization of the graphics.

A. Databases and Experimental Setup

The databases used in the experiments were Adult, Letter Recognition, Mushroom, Optical Recognition of Handwritten Digits and Vehicle Silhouettes, from the UCI Machine Learning Repository [16]. The number of input attributes, classes and instances of each database are shown in table I.

TABLE I
Databases

Database	Inputs	Classes	Instances
Adult	14	2	45222
Letter	16	26	20000
Mushroom	21	2	8124
Optdigits	64	10	5620
Vehicle	18	4	846

The parameters used in the experiments, except the learning rate used for the stochastic back-propagation learning, were chosen after visual inspection of some preliminary executions varying the parameters. After that, 5 runs of 2-fold cross-validation were performed with the chosen parameters. All the comparisons presented in sections IV-B and IV-C are confirmed by 5x2 cross-validation F tests with 95% of confidence, as recommended in [31] and [32].

For the on-line stochastic back-propagation learning, 5 runs of 2-fold cross-validation were performed for each of the 6 learning rates from 1 to 0.00001, to guarantee that the classification errors obtained are not result from a bad learning rate choice. The best classification error averages were attained by using learning rate 0.1 for all databases but Vehicle, in which the best classification error average was obtained by using 0.01. The results reported in the rest of the paper are the ones obtained with the best learning rates for each database. Experiments using 5 runs of 2-fold cross-validation were also done presenting each training instance a certain number of times (more than 1) "on-arrival" and then discarded, as it was done in [30] to perform on-line learning.

TABLE II
MLP PARAMETERS

Database	Hidden nodes	Epochs	Learning rate
Adult	5	100	0.1
Letter	40	300	0.1
Mushroom	25	100	0.05
Optdigits	10	100	0.1
Vehicle	30	1500	0.1

However, the accuracies did not improve in comparison with the use of off-line MLPs. So, the rest of the paper shows only the results obtained by presenting each training instance only once "on-arrival".

The ensembles created in the experiments were composed by 10 MLPs combined by majority-vote and the penalty strength was $\gamma = 0.4$. It is important to notice that the preliminary executions showed that the on-line ensemble learning methods are quite robust to the choice of γ, as long as its value is not close to the upper boundary [33] of this parameter. So, the differences in the classification error are highly dependent on the base learner, making its choice an important step when using on-line NCL or on-line bagging NCL. However, further detailed study about the influence of γ on the classification error in on-line mode is important and proposed as a future work.

The EFuNN parameters were the following: error threshold $E = 0.1$, initial sensibility threshold $S = 0.9$, maximum radius of receptive field $Mrad = 0.5$, membership functions number = 3, membership functions type = triangular, $m - of - n = 3$, no rules extraction, no aggregation and no pruning, except for Adult, in which pruning was used with $Pr = 1$ and node age $OLD = 200$, and Vehicle, in which the error threshold was $E = 0.001$.

The number of hidden nodes used for both on-line and off-line MLPs trained with stochastic back-propagation learning and the number of epochs and the learning rate used for off-line stochastic bask-propagation learning are shown in table II.

B. On-line NCL Vs. Off-line NCL

This section presents a comparison between NCL using on-line and off-line stochastic back-propagation MLPs. It shows that NCL with on-line MLPs is unsuitable for on-line learning, due to the high classification errors obtained.

The classification error averages of the ensembles of on-line MLPs and off-line MLPs produced by NCL are shown in figure 1 It is possible to observe that both the train and test (generalization) errors obtained by the ensembles of on-line MLPs are usually considerably and sometimes even drastically increased in comparison with the use off-line MLPs. Except for the Adult database, the classification errors obtained using on-line MLPs are always more than twice the classification errors obtained using off-line MLPs. In Letter and Vehicle, the classification error averages using on-line MLPs are even higher than 50%, being unacceptable.

For all databases but Adult, 5x2 cross-validation F tests [32] indicate that NCL with on-line MLPs produces worse classification errors than NCL with off-line MLPs. Table III shows the averages (Av), standard deviations (SD) and statistic f of the 5x2 cross-validation F tests. The statistics f higher than 4.74 indicate that there is statistical significant difference between the averages with 95% of confidence. These statistics are marked with the symbol "*" in this and all the other tables of the paper.

The Adult database has a large number of training instances, causing the classification errors of NCL with on-line and off-line MLPs to be statistically the same. It is important to notice that, even if for large databases on-line MLPs can get classification errors similar to off-line MLPs, this would mean that in order to achieve a similar performance, on-line MLPs would need a greater number of training instances than off-line MLPs and the learning system as a whole would take more time to start making predictions with acceptable accuracy.

The experiments presented in this section indicate that NCL is not suitable to on-line learning when on-line MLPs are used as the base models. As the only difference between NCL applied in on-line and off-line mode is the base model, it is possible that the bad classification errors obtained are due to the on-line MLPs. On-line bagging NCL allows the use of more suitable base models to perform on-line learning, such as EFuNNs. Section IV-C presents experiments with this method, confirming that on-line MLPs are not so suitable for on-line learning as other classifiers such as EFuNN. So, the possibility of a wider range of choice for base model is an important characteristic of on-line bagging NCL.

C. On-line NCL Vs. On-line Bagging NCL

The classification error averages of the ensembles of EFuNNs produced by on-line bagging NCL and of the ensembles of on-line MLPs produced by NCL are shown in figure 1. Table IV shows the classification error averages, standard deviations and statistics f of the 5x2 cross-validation F tests done to compare these methods. The experiments show that for all databases but Adult, on-line bagging NCL with EFuNNs obtains statistically better (lower) classification error than NCL with on-line MLPs. Again, the large number of training instances of Adult causes the classification error averages to be statistically the same.

These results show that the on-line bagging using EFuNNs can get better classification error than NCL using on-line MLPs, validating the proposed method. However, it is important to check whether on-line bagging NCL with EFuNNs outperforms NCL with on-line MLPs because of the possibility to choose EFuNNs as the base models or because of on-line bagging. In order to do that, the following 2 comparisons were done:

1) NCL with on-line MLPs versus on-line bagging NCL with on-line MLPs - to check whether on-line bagging NCL by itself is improving or not the classification error in relation to NCL.
2) On-line bagging NCL with on-line MLPs versus on-line bagging NCL with EFuNNs - to complement the

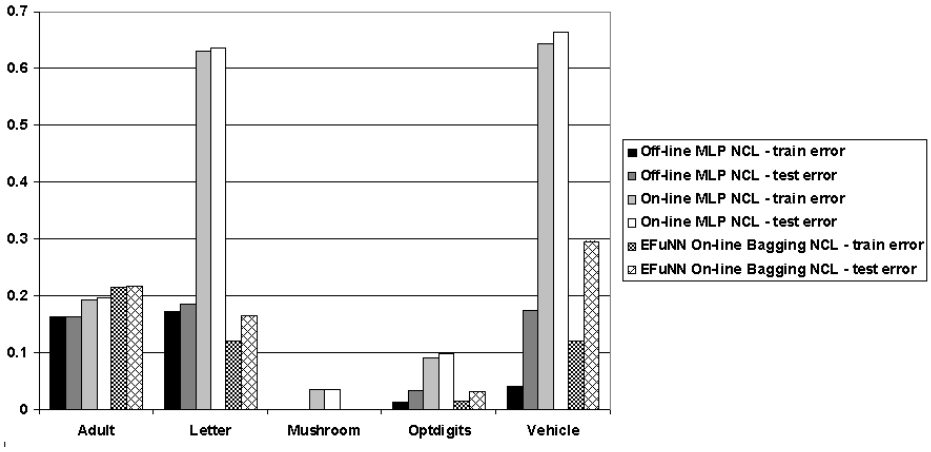

Fig. 1. Average classification error

TABLE III

CLASSIFICATION ERROR AVERAGES, STANDARD DEVIATIONS AND f STATISTICS OF THE 5X2 CROSS-VALIDATION F TESTS [32] OF THE ENSEMBLES OF ON-LINE MLPS AND OFF-LINE MLPS TRAINED WITH NCL. VALUES f MARKED WITH THE SYMBOL "*" INDICATE STATISTICAL SIGNIFICANT DIFFERENCE WITH 95% OF CONFIDENCE.

	Train error			Test error		
	NCL with On-line MLPs Av +- SD	NCL with Off-line MLPs Av +- SD	f	NCL with On-line MLPs Av +- SD	NCL with Off-line Av +- SD	f
Adult	0.19383 +- 0.01980	0.16193 +- 0.01438	3	0.19633 +- 0.01848	0.16247 +- 0.01222	3
Letter	0.63195 +- 0.04651	0.17184 +- 0.00693	*97	0.63435 +- 0.04768	0.18577 +- 0.00863	*77
Mush.	0.03508 +- 0.00418	0.00000 +- 0.00000	*56	0.03543 +- 0.00495	0.00020 +- 0.00062	*46
Opt.	0.09135 +- 0.01967	0.01331 +- 0.00170	*12	0.09868 +- 0.02200	0.03274 +- 0.00428	*9
Vehicle	0.64255 +- 0.04712	0.04043 +- 0.00682	*614	0.66478 +- 0.05651	0.17494 +- 0.02654	*111

TABLE IV

CLASSIFICATION ERROR AVERAGES, STANDARD DEVIATIONS AND f STATISTICS OF THE 5X2 CROSS-VALIDATION F TESTS [32] OF THE ENSEMBLES OF ON-LINE MLPS TRAINED WITH NCL AND OF THE ENSEMBLES OF EFUNNS TRAINED WITH ON-LINE BAGGING NCL. VALUES f MARKED WITH THE SYMBOL "*" INDICATE STATISTICAL SIGNIFICANT DIFFERENCE WITH 95% OF CONFIDENCE.

	Train error			Test error		
	NCL with on-line MLPs Av +- SD	On-line bagging NCL with EFuNNs Av +- SD	f	NCL with on-line MLPs Av +- SD	On-line bagging NCL with EFuNNs Av +- SD	f
Adult	0.19383 +- 0.01980	0.21567 +- 0.01858	0	0.19633 +- 0.01848	0.21852 +- 0.01963	0
Letter	0.63195 +- 0.04651	0.12192 +- 0.00457	*120	0.63435 +- 0.04768	0.16530 +- 0.00357	*85
Mush.	0.03508 +- 0.00418	0.00007 +- 0.00012	*55	0.03543 +- 0.00495	0.00010 +- 0.00013	*49
Opt.	0.09135 +- 0.01967	0.01630 +- 0.00157	*11	0.09868 +- 0.02200	0.03128 +- 0.00395	*8
Vehicle	0.64255 +- 0.04712	0.12222 +- 0.00965	*1365	0.66478 +- 0.05651	0.29551 +- 0.01142	*231

first comparison, checking the importance of the base learner to the performance of on-line bagging NCL.

Table V shows the classification error averages, standard deviations and statistics f of the 5x2 cross-validation F tests for the first comparison. It is possible to observe that on-line bagging NCL with on-line MLPs obtains either statistically equal or worse classification error averages than NCL with on-line MLPs. So, on-line bagging NCL by itself cannot improve the classification in relation to NCL.

Table VI shows the classification error averages, standard deviations and statistics f of the 5x2 cross-validation F tests for the second comparison. It is possible to observe that on-line bagging NCL with EFuNNs obtains statistically better classification error than on-line bagging NCL with on-line MLPs for all databases but Adult.

These two comparisons show that EFuNNs play a very important role in improving on-line bagging NCL's classification. So, the possibility to choose deterministic classifiers such as EFuNNs is an important characteristic of on-line bagging NCL, making it possible to improve its generalization

TABLE V

CLASSIFICATION ERROR AVERAGES, STANDARD DEVIATIONS AND f STATISTICS OF THE 5X2 CROSS-VALIDATION F TESTS [32] OF THE ENSEMBLES OF ON-LINE MLPS TRAINED WITH NCL AND OF THE ENSEMBLES OF ON-LINE MLPS TRAINED WITH ON-LINE BAGGING NCL. VALUES f MARKED WITH THE SYMBOL "*" INDICATE STATISTICAL SIGNIFICANT DIFFERENCE WITH 95% OF CONFIDENCE.

	Train error average			Test error average		
	NCL with On-line MLP Av +- SD	On-line Bagging NCL with On-line MLP Av +- SD	f	NCL with On-line MLP Av +- SD	On-line Bagging NCL with On-line MLP Av +- SD	f
Adult	0.19383 +- 0.01980	0.21034 +- 0.04975	1	0.19633 +- 0.01848	0.21266 +- 0.05008	1
Letter	0.63195 +- 0.04651	0.76792 +- 0.05509	3	0.63435 +- 0.04768	0.77045 +- 0.05412	3
Mush.	0.03508 +- 0.00418	0.06278 +- 0.01293	*7	0.03543 +- 0.00495	0.06381 +- 0.01530	*7
Opt.	0.09135 +- 0.01967	0.17651 +- 0.02607	*6	0.09868 +- 0.02200	0.18552 +- 0.02737	*6
Vehicle	0.64255 +- 0.04712	0.67021 +- 0.04291	2	0.66478 +- 0.05651	0.68156 +- 0.04213	0

TABLE VI

CLASSIFICATION ERROR AVERAGES, STANDARD DEVIATIONS AND f STATISTICS OF THE 5X2 CROSS-VALIDATION F TESTS [32] OF THE ENSEMBLES OF ON-LINE MLPS TRAINED WITH ON-LINE BAGGING NCL AND OF THE ENSEMBLES OF EFUNNS TRAINED WITH ON-LINE BAGGING NCL. VALUES f MARKED WITH THE SYMBOL "*" INDICATE STATISTICAL SIGNIFICANT DIFFERENCE WITH 95% OF CONFIDENCE.

	Train error average			Test error average		
	On-line Bagging NCL			On-line Bagging NCL		
	On-line MLP Av +- SD	EFuNN Av +- SD	f	On-line MLP Av +- SD	EFuNN Av +- SD	f
Adult	0.21034 +- 0.04975	0.21567 +- 0.01858	0	0.21266 +- 0.05008	0.21852 +- 0.01963	1
Letter	0.76792 +- 0.05509	0.12192 +- 0.00457	*103	0.77045 +- 0.05412	0.16530 +- 0.00357	*95
Mush.	0.06278 +- 0.01293	0.00007 +- 0.00012	*41	0.06381 +- 0.01530	0.00010 +- 0.00013	*28
Opt.	0.17651 +- 0.02607	0.01630 +- 0.00157	*30	0.18552 +- 0.02737	0.03128 +- 0.00395	*26
Vehicle	0.67021 +- 0.04291	0.12222 +- 0.00965	*190	0.68156 +- 0.04213	0.29551 +- 0.01142	*71

TABLE VII

CLASSIFICATION ERROR AVERAGES, STANDARD DEVIATIONS AND f STATISTICS OF THE 5X2 CROSS-VALIDATION F TESTS [32] OF THE ENSEMBLES OF OFF-LINE MLPS TRAINED WITH NCL AND OF THE ENSEMBLES OF EFUNNS TRAINED WITH ON-LINE BAGGING NCL. VALUES f MARKED WITH THE SYMBOL "*" INDICATE STATISTICAL SIGNIFICANT DIFFERENCE WITH 95% OF CONFIDENCE.

	Train error			Test error		
	NCL with off-line MLPs Av +- SD	On-line bagging NCL with EFuNNs Av +- SD	f	NCL with off-line MLPs Av +- SD	On-line bagging NCL with EFuNNs Av +- SD	f
Adult	0.16193 +- 0.01438	0.21567 +- 0.01858	3	0.16247 +- 0.01222	0.21852 +- 0.01963	4
Letter	0.17184 +- 0.00693	0.12192 +- 0.00457	*38	0.18577 +- 0.00863	0.16530 +- 0.00357	3
Mush.	0.00000 +- 0.00000	0.00007 +- 0.00012	1	0.00020 +- 0.00062	0.00010 +- 0.00013	1
Opt.	0.01331 +- 0.00170	0.01630 +- 0.00157	*6	0.03274 +- 0.00428	0.03128 +- 0.00395	2
Vehicle	0.04043 +- 0.00682	0.12222 +- 0.00965	*26	0.17494 +- 0.02654	0.29551 +- 0.01142	*26

in relation to NCL.

Another important comparison to show the importance of the method proposed in this paper is the comparison between on-line bagging NCL with EFuNNs and NCL with off-line MLPs. Figure 1 shows the classification error averages obtained by both the approaches. Table VII shows the classification error averages, standard deviations and statistics f of the 5x2 cross-validation F tests. The statistical tests show that there is no statistical significant difference between the test classification errors obtained by on-line bagging NCL with EFuNNs and NCL with off-line MLPs, except for the Vehicle database, which is a short database. This is an impressive result, as in off-line learning the MLPs can process the whole training set a certain number of epochs, while EFuNNs process each training example only once. This analysis emphasizes even more the importance of having a wider range of choices for the base model, which is allowed by the proposed method.

V. Conclusions

This paper proposes a new method to on-line ensemble learning called on-line bagging NCL. Using this method, the training of an ensemble member is influenced by the training of the others, directly encouraging diversity. This is an advantage of the proposed method over the other on-line ensemble methods existent in the literature, except NCL. The advantage of the new method over NCL is that it sends a different sequence of training data to each one of the ensemble members, so that it is not necessary to build part of the diversity a priory. In this way, a wider range of base models can be used, including deterministic neural networks. So, on-line bagging NCL allows the choice of more adequate models to on-line learning, such as EFuNN.

The experiments show that NCL using on-line MLPs have high classification error due to the non suitability of on-line MLPs as base models. They also show that the proposed method using EFuNNs can outperform NCL using on-line MLPs in 4 out of 5 classification databases thanks to the wider range of base model choices, which allow the use of EFuNNs as the base model.

Moreover, on-line bagging NCL using EFuNNs manage to attain similar test classification error to NCL using off-line MLPs in 4 out of 5 databases. This is an impressive result, as in off-line learning the MLPs can process the whole training set a certain number of epochs, while EFuNNs process each training example only once. This analysis emphasizes even more the importance of having a wider range of choices for the base model, which is allowed by the proposed method.

References

[1] M. Tagscherer, L. Kindermann, A. Lewandowski, and P. Protzel, "Overcome neural limitations for real world applications by providing confidence values for network prediction," in *Proceedings of the Sixth International Conference on Neural Information Processing (ICONIP'99)*, vol. 2, Perth, Australia, 1999, pp. 520–525.

[2] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *Proceedings of the Third International IEEE Conference on Data Mining (ICDM'03)*. Los Alamitos, CA: IEEE Press, 2003, pp. 123–130.

[3] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning (ICML'95)*. San Francisco, CA: Morgan Kaufmann Publishers, 1995, pp. 331–339.

[4] A. Fern and R. Givan, "Online ensemble learning: An empirical study," *Machine Learning*, vol. 53, pp. 71–109, 2003.

[5] RoboCup Federation, "RoboCup official site," 2007. [Online]. Available: http://www.robocup.org/

[6] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'01)*. New York: ACM Press, 2001, pp. 359–364.

[7] C. Domingo and O. Watanabe, "MadaBoost: A modification of adaboost for the filtering framework," Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Tech. Rep. TR-C138, 1999.

[8] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 29, no. 6, pp. 716–725, 1999.

[9] ——, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, pp. 1399–1404, 1999.

[10] M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 820–834, 2003.

[11] Z. Wang, X. Yao, and Y. Xu, "An improved constructive neural network ensemble approach to medical diagnoses," in *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'04), Lecture Notes in Computer Science*, vol. 3177. Exeter, UK: Springer, 2004, pp. 572–577.

[12] A. Chandra, H. Chen, and X. Yao, *Multi-objective Machine Learning*. Springer-Verlag, 2006, ch. Trade-off Between Diversity and Accuracy in Ensemble Generation, pp. 429–464.

[13] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[14] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[15] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 31, no. 6, pp. 902–918, 2001.

[16] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[17] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, vol. 2, pp. 285–318, 1988.

[18] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, pp. 212–261, 1994.

[19] C. P. Lim and R. F. Harrison, "Online pattern classification with multiple neural network systems: An experimental study," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 33, no. 2, pp. 235–247, 2003.

[20] S. B. Kotsiantis and P. E. Pintelas, "An online ensemble of classifiers," in *Proceedings of the Fourth International Workshop on Pattern Recognition in Information Systems (PRIS'04)*. Porto, Portugal: INSTICC Press, 2004, pp. 59–68.

[21] A. Chandra and X. Yao, "Evolving hybrid ensembles of learning machines for better generalisation," *Neurocomputing*, vol. 69, pp. 686–700, 2006.

[22] C. Domingo and O. Watanabe, "Madaboost: A modification of adaboost," in *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*. San Francisco: Morgan Kaufmann Publishers Inc, 2000, pp. 180–189.

[23] T. G. Dietterich, "Machine learning research: Four current directions," *Artificial Intelligence*, vol. 18, no. 4, pp. 97–136, 1997.

[24] Y. Freund and R. E. Schapire, "Game theory, on-line prediction and boosting," in *Proceedings of the Ninth Annual Conference on Computational Learning Theory*. New York: ACM Press, 1996, pp. 325–332.

[25] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999. [Online]. Available: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume11/opitz99a-html/paper.html

[26] P. Sollich and A. Krogh, "Learning with ensembles: How over-fitting can be useful," *Advances in Neural Information Processing Systems*, vol. 8, pp. 190–196, 1996.

[27] F. L. Minku, H. Inoue, and X. Yao, "Negative correlation in incremental learning," *Natural Computing Journal - Special Issue on Nature-inspired Learning and Adaptive Systems*, p. 32p, 2008 (accepted).

[28] C. M. Bishop, *Neural Networks for Pattern Recognition*. United Kingdom: Oxford University Press, 2005.

[29] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller, "Efficient BackProp," in *Neural networks: tricks of the trade*. Berlin: Springer, 1998, p. 44p.

[30] N. C. Oza and S. Russell, "Online bagging and boosting," in *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3. New Jersey: Institute for Electrical and Electronics Engineers, 2005, pp. 2340– 2345.

[31] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10, pp. 1895–1923, 1998.

[32] E. Alpaydin, "Combined 5x2cv F test for comparing supervised classification learning algorithms," *Neural Computation*, vol. 11, pp. 1885–1892, 1999.

[33] G. Brown, J. L. Wyatt, and P. Tiňo, "Managing diversity in regression ensembles," *Journal of Machine Learning Research*, vol. 6, pp. 1621–1650, 2005.