Maximum Likelihood Topology Preserving Ensembles

Emilio Corchado[1], Bruno Baruque[1] and Bogdan Gabrys[2]

[1] Department of Civil Engineering. University of Burgos. Spain
escorchado@ubu.es, bbaruque@ubu.es
[2] Computational Intelligence Research Group. Bournemouth University. United Kingdom
bgabrys@bournemouth.ac.uk

Abstract. Statistical re-sampling techniques have been used extensively and successfully in the machine learning approaches for generations of classifier and predictor ensembles. It has been frequently shown that combining so called unstable predictors has a stabilizing effect on and improves the performance of the prediction system generated in this way. In this paper we use the re-sampling techniques in the context of a topology preserving map which can be used for scale invariant classification, taking into account the fact that it models the residual after feedback with a family of distributions and finds filters which make the residuals most likely under this model. This model is applied to artificial data sets and compared with a similar version based on the Self Organising Map (SOM).

1 Introduction

Topographic map formation is an organizing principle in the mammalian cerebral cortex. It consists of an orderly topographical arrangement of motor and sensory neurons with similar response properties across the cortical surface. There is an experimental evidence of the existence of cortical maps in the brain, and some examples have been identified in the visual cortex, somatosensory cortex and the auditory cortex.

The most typical example of an artificial topographic map formation is the Self-Organising Map (SOM) [1], [2], [3]. SOM is composed of a discrete array of L nodes arranged on an N-dimensional lattice and it maps these nodes into D-dimensional data space while maintaining their ordering. The dimensionality, N, of the lattice is normally less than that of the data. With the SOM, all data in a partition is quantised to a single point, and the combined effect of all of the vector-quantising nodes is to give a globally non-linear representation of the data set. Typically, the array of nodes is one or two-dimensional, with all nodes connected to the N inputs by an N-dimensional weight vector.

Another example of a topographic mapping algorithm is the Maximum Likelihood Scale Invariant Map (MLSIM) [4], [5], [6]. It is similar model to a Self-Organising Map (SOM) [3] but in this case training is based on the use of a particular Exploratory Projection Pursuit (EPP) model [7], [8] called Maximum Likelihood Hebbian Learning (MLHL) Network [6], [9]. The competitive learning and a neighbourhood function are then used in a similar way as with the SOM.

E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 1434 – 1442, 2006.

Competitive learning based networks are inherently instable, due to the nature of their learning algorithm. That means that even running the same algorithm, under the same learning conditions several times can give quite different results. To try to minimize the effect of this instability several methods are being studied. One of the most popular is the bagging technique [10]. This technique consists of constructing several different classifiers of the same type and combining their outputs. To train each one of the classifiers, a different subset of the training data is used, so a bit of diversity is included in the ensemble.

Tests on real and simulated datasets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy. The vital element is the instability of the classifying method. It has been observed that if perturbing the learning set can cause significant changes in the classifier decisions, the bagging can improve accuracy [11], [12], [13].

In this paper the instability of individual MLSIMs and potential for improvement of the performance using ensembles of classifiers are exploited by utilising bagging like combination approaches and MLSIM weight initialisation procedures described in the following sections.

2 Bagging

The term "bagging" refers to the union of two other words: “bootstrapping” and “aggregating” [10]. The first one refers to the way the inputs are extracted from the dataset used for training the predictor(s). The second refers to the fact that, instead of a unique one, a set or aggregation of predictors should be constructed. The aggregated predictor is potentially much more powerful than any individual predictor trained on the same data.

This technique, utilized in this study to improve the classifying capacity of certain topography preserving maps, is based on statistical re-sampling theory. The description of this "bootstrap aggregating" or "bagging" technique can be found in [10] and a version of bagging will be exploited in this paper in the context of topology preserving maps.

When dealing with classification trees for instance, this technique has been employed to generate n subsets of the main dataset under analysis through re-sampling with replacement and training individual decision trees on such re-sampled subsets. This permits to generate n classifiers which are most often combined by simple majority voting of their decisions [11], [12].

In our case, the idea is to employ the bagging like technique in combination with the MLSIM training carried out on several re-sampled subsets of the original training set. Once the multiple versions of MLSIM are generated the ensemble output is computed by simple voting procedure [14], [15].

3 Maximum Likelihood Scale Invariant Maps

Maximum Likelihood Scale Invariant Map (MLSIM) [4], [5] is an extension of the Scale Invariant Map (SIM) [16] based on the application of the Maximum Likelihood Hebbian Learning (MLHL) [6], [9].

As mentioned earlier an MLSIM is a regular array of nodes arranged on a lattice. Competitive learning and a neighbourhood function are used in a similar way as with the SOM. The input data (x) is fed forward to the outputs y_i in the usual way. After selection of a winner, the winner, c, is deemed to be firing (y_c=1) and all other outputs are suppressed ($y_i = 0, \forall i \neq c$).

The winner's activation is then fed back through its weights and this is subtracted from the inputs to calculate the error or residual $\mathbf{e}$ as shown in Eq. 1.

$$\mathbf{e} = x - W_c \cdot y_c, (y_c = 1) \tag{1}$$

Following this, the Maximum Likelihood Hebbian Learning is used to update the weights of all nodes in the neighbourhood of the winner which can be expressed as in Eq. 2.

$$\Delta W_i = h_{ci} \cdot \eta \cdot sign(\mathbf{e} - W_c) \,|\, \mathbf{e} - W_c \,|^{p-1}, \forall i \in N_c \quad \textit{with different values of } p \tag{2}$$

By giving different values to p [6], [9], the learning rule is optimal for different probability density functions of the residuals. h_{ci} is the neighbourhood function as in the case of the SOM and N_c is the number of output neurons. Finally, η represents the learning rate.

While training of a SOM relies on iteratively selecting a winner stimulated by the inputs, and updating the weights, in the case of the MLSIM, the weights of the winning node are fed back as inhibition at the inputs, and then, MLH learning is used to update the weights of all nodes in the neighbourhood of the winner as explained above.

4 MLSIM Ensembles

As explained before we intend to apply bagging in combination with MLSIM with the main objective of improving the classification performance of the ensemble of MLSIMs in comparison to individual MLSIM and some other topology preserving maps i.e. SOM.

4.1 Training the MLSIM Ensemble

When constructing MLSIM ensemble, first a subset of data is randomly drawn from the training dataset and used to train only one of the networks. For the next trained network the process is repeated. Thus, the networks of the ensemble are trained using slightly different datasets, giving as a result the desired diversity.

In our investigations we have conducted three distinctive procedures for constructing the ensemble of MLSIMs related to the initialisation of weights in the trained individual MLSIMs which have had a significant effect on the performance.

In the first approach we have trained several MLSIMs separately with completely random initialisation of the weights. However due to the random initialisation of the weights it was very difficult to compare the results of individual output neurons from different networks. In order to remedy this problem a more controlled way of setting the initial weights was required.

Our next step was to initialize the weights of several neurons of each network according to the first and second Principal Components of the training dataset in order to initiate all the networks within the constructed ensemble in a similar manner, but still maintaining a certain degree of independence (in a similar way it is done in [17]). To obtain these two first Principal Components (PCs), we have applied a PCA ensemble described in one of our previous papers [18]. Specifically, the weights of the first *(1)* and the middle *(n/2)* neurons were initiated to the values representing the first principal component, while the weights of the neurons situated in positions *(n/4)* and *((3*n)/4)* (with *n* being the number of neurons in the network) were initiated with the values representing the second principal component. This procedure has been applied to all the networks in the ensemble.

The first and second PCs are orthogonal, so the mentioned neurons (labelled as *1, n/2, n/4 and (3*n)/4*) are going to be initially located forming a kind of cross, along the two PCs as it can be seen in Fig. 1. This initialization has been performed deliberately in this way as it is known that MLSIM weights are commonly distributed forming a circular shape [4], [5], [16].

Additional benefit of such coordinated initialisation of the weights is the fact that the results of all individual networks are much easier to compare by visual inspection, as the networks tend to update their weights in a similar way.

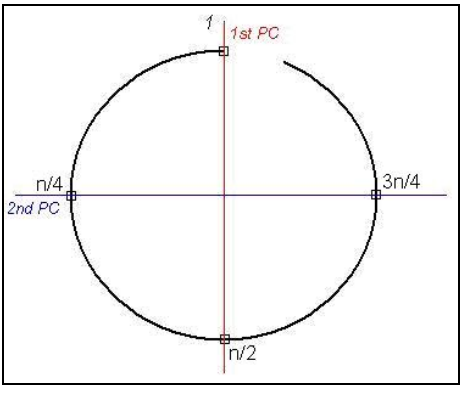

Fig. 1. Initialization of the MLSIM weights along the first two PCs

Although it is not critical in the examined context, in the third approach to weights initialisation we have tried to force even more of a commonality to the ensemble. This has been carried out mainly having in mind a possible future study of combination at the model rather than a decision level [13], [19] and also to make the networks even more easily comparable by visual inspection. In order to do so, we use the final weights obtained after training one network to initiate the next. The first trained network of the ensemble was initialized with the Principal Components as explained before, while the following ones were initiated with the final weights of their corresponding predecessors. In this way, the set of networks is quite more "compact", although it keeps its diversity element by using different bootstrap samples of the dataset used to train them.

The results showed below have been obtained by using this last initialisation and training procedure.

4.2 Testing the Classification

We have tested the effectiveness of the classification made by our proposed model using a similar semi-supervised technique as the one described in [17] and in combination with a classical ten-fold cross validation.

We randomly divide the input dataset into ten subsets and perform training and testing process ten times. Each time we iterate over this main algorithm we select a different part of the input dataset as a testing dataset, while the other nine parts are used for training. In this way all data were used to train and test the ensemble. At the end, we average the testing results obtained in the ten tests to achieve a final classification accuracy result.

Each of the ten times we perform the previously explained "outer loop", we take three steps.

1. **Training.** In this case the ensemble of MLSIMs is trained using the novel technique explained in section 4.1. We use the 9/10 of data considered as training data in the current iteration.
2. **Labelling of output neurons.** As MLSIMs use an unsupervised learning technique this step consists of presenting again the training dataset to the recently trained ensemble in order to label the output neurons with the most consistently recognised class label. As we know which class each of the training data belongs to through the given class label, by presenting the training data to the ensemble, we will consider that the output neuron is specialized in recognizing data from that class if it responded to training samples from that class as a winner in a majority of the cases.
3. **Testing.** In this step we present to the ensemble of MLSIMs the other 1/10 of data that was left out of the training process. The testing dataset is also labelled with its corresponding class labels, so we can compare the class the ensemble classifies a testing sample as with its real/given class label. In this way a measure of the accuracy of the ensemble can be obtained. To decide to which class an input belongs to, the MLSIM ensemble performs a majority voting among its composing MLSIMs [15]. The input is presented to each network individually, each one finds the winner neuron for that input and gives as an answer the class that winner neuron is supposed to recognize better (as determined in step 2). The ensemble collects the answers from all its composing networks and returns the answer that was repeated the largest number of times (i.e. in the majority of the cases).

5 Data Set and Results

In order to test the performance of our model in a dataset where it is supposed to do best [16] we have generated a radial dataset. It is composed of six normal distributions of 2 dimensions disposed in a radial way. Their centres are situated in points (3,2), (1,4), (-2,4), (-3,1), (-2,-4) and (1,-2) respectively. The number of samples corresponding to each distribution is as follows: 50, 100, 70, 50, 20 and 100.

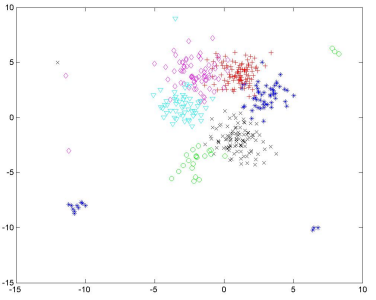

Fig. 2. Samples of the radial artificial data used in this study.

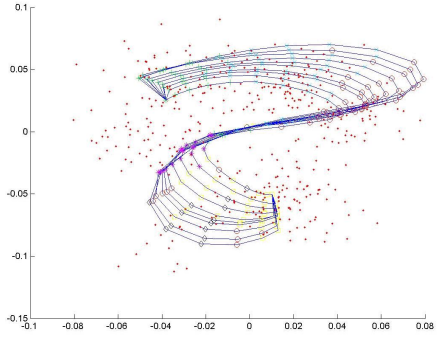

Fig. 3. Ten SOMs trained on the bagged data. The weights of the first one were initialized to the Principal Components of the dataset. The following ones were initialized to the final weights of its predecessor.

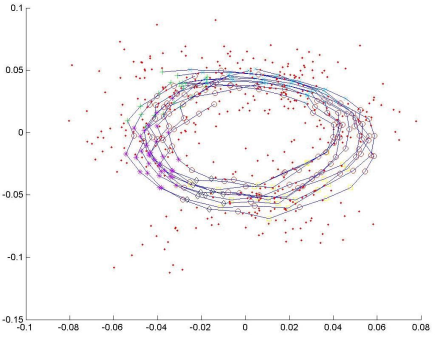

Fig. 4. Ten MLSIMs trained on the bagged data. The weights of the first one were initialized to the Principal Components of the dataset. The following ones were initialized to the final weights of its predecessor.

We have also included several outlier points to compare the results of different classification models when they are or are not present.

Fig. 3 and Fig. 4 show the results of training the ensembles of ten SOMs [17] and ten MLSIMs, respectively. As it can be seen the SOM ensemble tries to expand and cover the whole dataset range by forming a kind of inverted 'S', while the MLSIM ensemble does the same thing by using a circular form. In this particular case of a radial form dataset, the second approach should give better results, as the second form fits better the form of the considered dataset.

This difference in the form the two ensembles try to approximate to the data is mainly due to the training algorithm used (particularly the weights update) [3], [4], [5], [16] but also because of the initialization of the weights of the trained networks, as explained in section 4.1.

We have applied three classification models to the above described data set (including and without outliers). As expected the MLSIM ensemble model obtains better results than the single MLSIM and the SOM ensemble models, without and with outliers in the data set, as can be seen in Table 1 and Table 2. In the case of a single MLSIM, the variation between the best and worse accuracy results

Table 1. Classification accuracy of three different models applied to the data set from Fig. 2. The minimum, maximum and average accuracy from 10-fold cross validation testing runs are shown in the table. All the experiments were performed without the 20 outlier points.

	Accuracy of the Model (without outliers)		
	min	max	average
Single MLSIM	81.28%	86.15 %	83.58%
Ensemble (10 MLSIMs)	86.15%	88.2%	87.02%
Ensemble (10 SOM)	80.76 %	86.41 %	83.11%

Table 2. Classification accuracy of three different models applied to the data set from Fig. 2. The minimum, maximum and average accuracy from 10-fold cross validation testing runs are shown in the table. All the experiments were performed including 20 outlier points.

	Accuracy of the Model (including outliers)		
	min	max	average
Single MLSIM	75.36%	83.17 %	79.6 %
Ensemble (10 MLSIMs)	82.19%	86.34%	84.15%
Ensemble (10 SOM)	79.20 %	85.6 %	82.86%

(for dataset without outliers), is almost 5%, meaning that the model exhibit certain instability. This issue is even easier to be seen, in the case when outliers (i.e. mislabelled data) are present in the studied data set. The difference in this case is close to 8%. In the case of the model presented in this study, the MLSIM ensemble, we can see that the difference in both cases is smaller than in the case of a single one. It is around 2% for the case without outliers present, and around 4% when outliers are present.

For comparison purposes, we have applied an ensemble version of a SOM [17]**.** In the first case, when no outliers are present, the difference is less than 6%, and above 6% with outliers in the data set.

All these experiments have demonstrated how the MLSIM ensemble performs better than a single MLSIM model and an SOM ensemble version, in the case of radial data sets.

6 Conclusions

In this study we have applied a statistical re-sampling method for creating ensembles of classifiers based on a topology preserving model, MLSIM, trained in an unsupervised manner. We have studied and compared different ways to initialise the centres of individual MLSIMs and proposed an approach based on utilising Principal Components Analysis and sequential initialisation of subsequent MLSIMs within created ensemble.

We have compared the novel ensemble model with a SOM ensemble version and showed how our model improved the results obtained by the SOM when using radial data sets.

This study shows how the use of an ensemble version of an MLSIM improves the single model providing it with more stability and accuracy.

Future work will also investigate these ensemble methods on a range of artificial and real data sets, and the application of other viable classifier combining techniques such as the one known as bumping.

Acknowledgments

This research has been supported by the MCyT project TIN2004-07033 and the project BU008B05 of the JCyL.

References

1. Kohonen, T. Self-Organization and Associative Memory. Springer-Verlag, Heidelberg, Germany, 1984.
2. Kohonen, T. Barna, G and Chrisley R. Statistical Pattern Recognition with Neural Networks. In Proceeding of International Joint Conference of Neural Networks (pp. 61-88), IEEE Press, 1988.
3. Kohonen, T. The Self-Organizing Map. In Proceedings of the IEEE 78 (pp. 1464-1480), 1990.
4. Corchado, E. and Fyfe, C. Maximum Likelihood Topology Preserving Algorithms. In Proceedings of the U.K. Workshop on Computational Intelligence, Birmingham, UK, 2002.
5. Corchado, E. and Fyfe C. The Scale Invariant Map and Maximum Likelihood Hebbian Learning. International Conference on Knowledge-Based & Intelligent Information & Engineering System, IOS Press, 2002.
6. Corchado, E., MacDonald, D., Fyfe C., Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit. Data Mining Knowledge Discovery 8(3): 203-225 (2004).
7. Friedman J., Tukey. J.: A Projection Pursuit Algorithm for Exploratory Data Analysis. IEEE Transaction on Computers, Vol. 23 (1974) 881-890.
8. Hyvärinen A.: Complexity Pursuit: Separating Interesting Components from Time Series. Neural Computation, Vol. 13(4) (2001) 883-898.
9. Fyfe, C. and Corchado, E. Maximum likelihood Hebbian rules. ESANN (European Symposium on Artificial Neural Networks), ISBN 2-930307-02-1, 2002.
10. Breiman, L. Bagging Predictors. Machine Learning, 24 (pp. 123–140), 1996.
11. Ruta, D. and Gabrys, B. A Theoretical Analysis of the Limits of Majority Voting Errors for Multiple Classifier Systems, Pattern Analysis and Applications, vol. 5, pp. 333-350, 2002.
12. Schapire, R.E; Freud, Y; Bartlett, P. and Lee, W.S. Boosting the margin: a new explanation for the effectiveness of voting methods. The Annals of Statistics, 26(5):1651–1686, 1998.
13. Gabrys, B. Learning Hybrid Neuro-Fuzzy Classifier Models From Data: To Combine or not to Combine? *Fuzzy Sets and Systems,* vol. 147, pp. 39-56, 2004.
14. Kuncheva, L. Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, 2004.

15. Ruta, D. and Gabrys, B. Classifier Selection for Majority Voting, Special issue of the journal of information fusion on Diversity in Multiple Classifier Systems, vol. 6, issue 1, pp. 63-81, 1 March 2005.
16. Fyfe, C. A Scale Invariant Map. Network: Computation in Neural Systems, 7 (pp 269-275), 1996.
17. Petrakieva, L. and Fyfe, C. Bagging and Bumping Self-organising Maps. Computing and Information Systems, 2003.
18. Gabrys, B., Baruque, B. and Corchado, E. Outlier Resistant PCA Ensembles. To appear in the proceedings of the International Conference on Knowledge-Based & Intelligent Information & Engineering System, KES'2006, 2006.
19. Gabrys, B., Combining Neuro-Fuzzy Classifiers for Improved Generalisation and Reliability, Proceedings of the Int. Joint Conference on Neural Networks, (IJCNN'2002) a part of the WCCI'2002 Congress, ISBN: 0-7803-7278-6, pp. 2410-2415, Honolulu, USA, May 2002.