Modular Neural Network Task Decomposition Via Entropic Clustering

Jorge M. Santos
Instituto Superior de Engenharia do Porto
Instituto de Engenharia Biomédica
Porto, Portugal
jms@isep.ipp.pt

Luís A. Alexandre
IT - Networks and Multimedia Group
Covilhã, Portugal
lfbaa@di.ubi.pt

Joaquim Marques de Sá
Instituto de Engenharia Biomédica
Porto, Portugal
jmsa@fe.up.pt

Abstract

The use of monolithic neural networks (such as a multilayer perceptron) has some drawbacks: e.g. slow learning, weight coupling, the black box effect. These can be alleviated by the use of a modular neural network. The creation of a MNN has three steps: task decomposition, module creation and decision integration. In this paper we propose the use of an entropic clustering algorithm as a way of performing task decomposition. We present experiments on several real world classification problems that show the performance of this approach.

1 Introduction

The purpose of this work is to address the issue of using an entropic clustering algorithm to perform task decomposition in a modular neural network (MNN) approach, as opposite to the traditional methods. Task decomposition is one of the strategies used to simplify the learning process of any learning system. In neural networks it basically consists of the partition of the input space in several regions, this way decomposing the initial problem in different subproblems. This is done based on the assumption that these regions possess different characteristics and so they should be learned by specialized neural networks. By posterior integration of the learning results, we are able to hopefully achieve better solutions for the initial problem. Generally, task decomposition can be obtained in three different ways: explicit decomposition (the task is decomposed by the designer before training), class decomposition (the decomposition is made based on the classes of the problem) and automatic decomposition. When automatic decomposition is used, it can either be made during the learning stage or it can be made before training the modules using an unsupervised or clustering algorithm. We use this last approach performing the task decomposition with LEGClust: the Layered Entropic SubGraph Clustering Algorithm [1] that uses an entropic proximity measure to obtain the clusters.

The paper is organized as follows: the next section describes a general MNN, section 3 describes the clustering algorithm (LEGClust) used for task decomposition, section 4 presents the experiments and, in the last section, we present the conclusions.

2 Modular neural network

A modular neural network is an ensemble of learning machines. The idea behind this kind of learning structure is the divide-and-conquer paradigm: the problem should be divided into smaller subproblems that are solved by experts (modules) and their partial solutions should be integrated to produce a final solution. (Figure 1).

Ensembles of learning machines proved to give better results than the single learners that produces them. The proofs are mainly empirical [2, 3] but there are some theoretical results [4, 5, 6] that also support this assumption.

To use a MNN, three stages have to be considered:

- The task decomposition, where the problem is divided into smaller problems, each one to be given to one of the modules or expert networks. To better understand the task decomposition process let us take a look at the artificial data set depicted in Fig.2. This is a 3 class problem where the input space is clearly divided into 2 regions: one of the regions (upper right) contains samples from 2 classes (crosses and circles) and the other

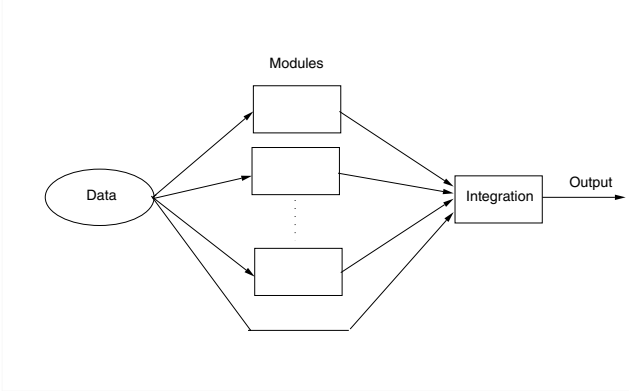


Figure 1. A modular neural network

contains samples from all 3 classes. Note that there are two classes with samples belonging to the 2 different regions. By having a classifier dedicated to each region we are able to transform this particular problem into two simpler ones.

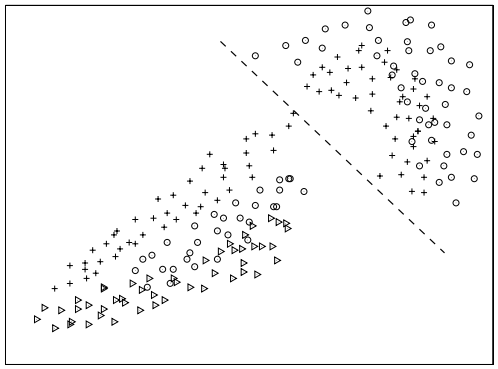


Figure 2. The partition of the input space for a 3 class problem.

- The training phase, where each individual expert (module) is trained until it learns to solve its particular subproblem.
- The decision integration. This strategy is used to combine the work of the experts and to produce a final network output. This can be done in several ways: using a gating network [7], make the modules vote [8] or through hierarchical integration (which can also use voting and/or gating networks) [9, 10]. In this paper we consider the use of a gating network. This network can be considered as an additional expert that is trained to recognized the region of the input space where each of the experts have their region of expertise, defined in the task decomposition phase.

After finishing the learning process, when a new pattern to be classified is presented to the network, the individual experts compute the class it might belong, but the gate network selects only a particular output that is given by the expert it considers to be 'competent' to solve the problem, taking into account the region of the input space to which the pattern belongs.

3 Task decomposition

As we mention before, the task decomposition is done before training the modules, using a clustering algorithm. Previous works like [11, 12, 13] and [6] already used this approach. There are several well known algorithms to perform clustering, being the most common ones those based on matrix theory and graph theory. However, it is also known that this kind of algorithms often have serious difficulties in identifying real clusters. The algorithms based on matrix theory build clusters according to some distance measure producing usually globular clusters and the algorithms based on graph theory, usually divisive algorithms, present serious difficulties in the process of graph partitioning to obtain plausible clusters.

In this paper we propose the use of a clustering algorithm developed by the authors to do the automatic task decomposition. This algorithm uses an entropic measure to build a proximity matrix used to construct several subgraphs. Based on these subgraphs we hierarchically obtain the clusters.

3.1 The LEGClust Algorithm

The LEGClust algorithm [1] stands for Layered Entropic subGraph Clustering algorithm. The basic foundations for this clustering algorithm are directed, maximally connected, unweighted subgraphs, built with an entropic measure. Using this entropic measure, we compute a proximity matrix and the related layered matrix used to construct subgraphs for each layer. In each subgraph each edge is the connection between each element and the corresponding element of that layer. The clusters are built hierarchically by joining together, following a set of established rules, the clusters that correspond to the layer subgraphs.

The main idea behind LEGClust is to make the connections follow the local structure of the data set. This is accomplish by using an entropic measure applied to the difference vectors between data samples.

Let $D = \{d_{ij}\}, i = 1, 2, .., M, j = 1, 2, .., M, i \neq j$, be the set of difference vectors (connections) associated with the M samples in a neighborhood of a certain data sample p. Let $H(D, p_i)$ be the entropy associated with connection p_i, the entropy of the set of all connections d_{ij} plus connection p_i, such that

$$H(D, p_i) = H(\{D\} \cup \{p_i\}), i = 1, 2, .., M \qquad (1)$$

Table 1. Pseudo-code for the LEGClust Algorithm.

Compute a dissimilarity matrix using any dissimilarity measure (to obtain the M nearest neighbors).
Compute the entropic dissimilarity matrix of the M nearest neighbors.
Compute the layered entropic proximity matrix of the M nearest neighbors.
Form the elementary clusters using the first layer.
While number-of-clusters > 1 do
Go to next layer (L)
Join clusters using the layer connections in L
End While

This entropy will be our dissimilarity measure. We will compute, for each element p, the M possible entropies and construct an entropic dissimilarity matrix and the correspondent entropic proximity matrix. The elements of the first column (first layer) of the proximity matrix are those correspondent to the samples having the smallest entropic dissimilarity value with each element.

The entropic measure used is the Renyi's Quadratic Entropy (H_{R2}). For a vector x ($x \in \mathbb{R}^m$), H_{R2} can be estimated as

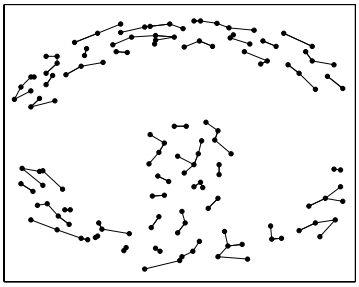
$$\begin{aligned}\hat{H}_{R2}(x) &= -\log\left[\int_{-\infty}^{+\infty}\left(\frac{1}{Nh^m}\sum_{i=1}^{N}G(\frac{x-x_i}{h},I)\right)^2 dx\right] \\ &= -\log\left[\frac{1}{N^2h^{2m-1}}\sum_{i=1}^{N}\sum_{j=1}^{N}G(\frac{x_i-x_j}{h},2I)\right]\end{aligned} \quad (2)$$

where N is the number of data samples, G is a radially symmetric Gaussian kernel, m is the dimension of x and h is the bandwidth parameter (also known as smoothing parameter or kernel size related to the Parzen Window method for probability density function estimation).

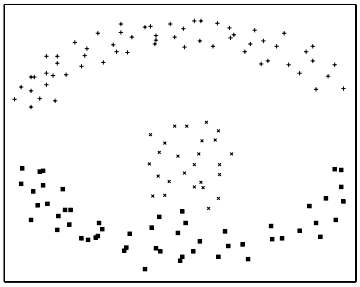
In LEGClust algorithm we use Renyi's Quadratic Entropy because of its simplicity; however one could use other entropic measures as well.

The pseudo-code for the LEGClust Algorithm is presented in Table 1

An example of the first layer subgraph obtained using LEGClust is depicted in Fig.3a. As we can see, these connections follow the local structure of the data set. For the same data set an example of a clustering solution produced using LEGClust is depicted in Fig.3b. We can see that clusters with different structures, combined in the same data set, are easily detected even if they are not well separated.



(a) First layer connections.



(b) Final clustering.

Figure 3. Clustering using LEGClust.

4 Experiments

We have performed a considerable number of experiments with several data sets using modular neural networks with task decomposition performed by three different clustering algorithms: one using k-means (K-MNN), other using Spectral Clustering [14] and the last using entropic clustering (EC-MNN). All the neural networks used in the experiments, both the modules and the gates of the MNN, were MLP's with one hidden-layer. The topologies of the MLPs were $[a : b : c]$, where a is the number of features, b is the number of neurons in the hidden layer and c is the number of classes treated by each expert and the number of experts for the gates. The neural networks were trained with backpropagation algorithm and early stopping. The experiments were made with the holdout method: half the data set was used for training and the other half for testing. Then the data sets were used with inverted roles (the original training set became the test set and the original test set became the training set). Each module is trained with the input data defined by the clustering algorithm (each module learns with the data from each cluster). The gate network is trained with all the data labelled by the clustering algorithm.

Regarding the clustering processes, since none of them give automatically the number of clusters, these must be defined by the user. Taking into account the data sets used in the experiments we only considered the possibilities of 2

and 3 clusters. Otherwise, the training set for each module would have an insufficient number of samples.

For the computation of the Renyi's Quadratic Entropy (2), one has to determine the value of the smoothing parameter. For this purpose we use a formula adapted from [15] to the specific characteristics of our entropic dissimilarity measure, that was proposed in [1]. The optimum value is given by

$$h_{op} = 2\,\sigma^* \left(\frac{4}{(m+2)N}\right)^{\frac{1}{m+4}} \tag{3}$$

where σ^* is the mean value of the standard deviations for each dimension of the vector x. The reason for using a single h_{op} for every dimension is related with the way we estimate Renyi's Entropy.

4.1 The data sets

We used in our experiments several real data sets and also the artificial one depicted in Fig.2 further designated as *ArtificialF2*. The real data sets are all publicly available. The Breast Tissue and the CTG data sets can be found in [16], Diabetes and Sonar in [17], 2VowelsPB in [7] and Olive in [18].

Table 2 contains a summary of the characteristics of these data sets. It shows the number of data samples, number of features and the number of classes for each data set.

Table 2. The data sets used in the experiments.

Data set	*# samples*	*# features*	*# classes*
ArtificialF2	222	2	3
Breast Tissue	106	9	6
CTG	2126	22	10
Diabetes	768	8	2
Olive	572	8	9
2VowelsPB	608	2	2
Sonar	208	60	2

4.2 Results

In Table 3 we present the parameters of each modular neural network for the results presented in Table 4. For each type of MNN we show the number of experts and, for each of them, we present the number of hidden neurons and the number of output neurons. The number of output neurons is defined by the number of classes in each cluster. The presented structures are corresponding to the best results in a large number of experiments with different combinations in the number of neurons in each module and in the gate.

Table 4. Errors and standard deviations for the performed experiments with MNN.

Data set	K-MNN	EC-MNN	S-MNN
ArtificialF2	16.40 (2.40)	14.70 (3.22)	15.32 (3.55)
Breast Tissue	58.95 (7.54)	32.79 (3.72)	33.53 (4.47)
CTG	22.90 (0.86)	20.67 (2.38)	23.91 (2.91)
Diabetes	24.45 (1.45)	23.89 (1.64)	23.96 (1.76)
Olive	49.11 (2.89)	4.74 (0.89)	5.20 (1.11)
2VowelsPB	7.23 (1.17)	7.25 (0.80)	7.28 (0.95)
Sonar	16.14 (3.43)	18.57 (3.40)	23.69 (4.57)

The results in Tables 4 and 5 are the average and standard deviations for 20 repetitions of each experiment.

4.3 Discussion

We must start by reminding that the MNN approach, with its associated task decomposition, will only be effective, giving better results than single neural networks, if the input space possesses some divisive proprieties, if different regions of the input space have different proprieties. This is the main reason why we just focus on the comparison between three different modular neural network task decomposition approaches and not the comparison with single neural networks. Just for information purpose we present in Table 5 the errors obtained using single neural networks(MLP's with one hidden layer. The number of neurons in the hidden layer is shown in column *Nh*). We can see that, there are data sets where we have a considerable reduction in the final error and others, instead, that present higher classification errors when compared with the MNN approaches.

So, what we want to compare in the performed experiments is the suitability of the entropic clustering to perform

Table 5. Errors and standard deviations for the performed experiments with single neural networks (SNN).

Data set	SNN	*Nh*
ArtificialF2	19.56 (3.95)	20
Breast Tissue	32.75 (3.26)	22
CTG	15.70 (0.60)	20
Diabetes	23.90 (1.69)	15
Olive	5.45 (0.62)	15
2VowelsPB	7.51 (0.37)	6
Sonar	21.90 (2.80)	14

Table 3. Structure of the modular neural networks used in the experiments corresponding to the results in Table 4.

Data set	K-MNN	EC-MNN	S-MNN
ArtificialF2	[2:18:3][2:18:2][2:18:2]	[2:20:3][2:12:2][2:14:2]	[2:18:3][2:18:2][2:16:2]
Breast Tissue	[9:10:2][9:12:2][9:12:2]	[9:12:2][9:12:2][9:3:2]	[9:4:2][9:14:2][9:6:2]
CTG	[22:22:10][22:18:9][22:18:9][22:26:3]	[22:20:6][22:18:2][22:26:2][22:26:3]	[22:18:10][22:22:10][22:22:10][22:22:3]
Diabetes	[8:10:2][8:12:2][8:10:2][8:12:3]	[8:14:2][8:18:2][8:12:2][8:16:3]	[8:18:2][8:12:2][8:10:3]
Olive	[8:6:6][8:12:4][8:12:8][8:12:3]	[8:10:4][8:4:3][8:12:2][8:8:3]	[8:12:4][8:12:4][8:4:3][8:6:3]
2VowelsPB	[2:6:2][2:6:2][2:2:2]	[2:4:2][2:5:2][2:2:2]	[2:5:2][2:5:2][2:2:2]
Sonar	[60:12:2][60:12:2][60:14:2]	[60:12:2][60:12:2][60:12:2]	[60:10:2][60:16:2][60:16:2]

task decomposition when compared with other methods, namely the k-means and spectral clusterings. To achieve that comparison we have two choices: use, for each data set, the same topology for the different kind of MNN's or, for each data set, try to find the best topology for each MNN. We think that the second hypothesis is the most reasonable because one should expect that the input regions obtained by the different task decomposition processes must have different proprieties and thus should be treated by different kinds of neural networks.

We can see that the average errors in almost every performed experiments were smaller for the EC-MNN than for the K-MNN and S-MNN. In some cases the differences in the performances are very substantial, specially when comparing EC-MNN and K-MNN, in the Breast Tissue and Olive data sets. The 2VowelsPB is a data set with 2 well separated globular clusters (one containing class 1 and 2 and the other classes 3 and 4); that is the reason for the equal results for the three different MNN's.

We used the Wilcoxon or ranksum non-parametric test to check the statistic significance of the differences between the similar results. The significance level used was $\alpha = 0.05$. The null hypothesis (medians are equal) can not be rejected for data set Diabetes between EC-MNN and K-MNN and for data sets Breast Tissue, Diabetes, ArtificialF2, and Olive between EC-MNN and S-MNN. Despite the results of the statistic tests the means and standard deviations of the performed experiments are smaller for the EC-MNN.

5 Conclusion

This paper introduced the EC-MNN (Entropic Clustering Task Decomposition for Modular Neural Networks). We used the LEGClust algorithm to choose the clusters in the task decomposition stage.

We tested the proposed approach in one artificial and six real world problems and compared the results with the K-MNN (k-means clustering) and the S-MNN (Spectral clustering) task decompositions.

It was shown empirically that there are several data sets that can benefit from the EC-MNN approach and that (generally) EC-MNN will yield better results (smaller classification error) than K-MNN and S-MNN.

Our experiments supported the idea that a MNN benefits from an entropic clustering task decomposition stage.

6 Acknowledgments

This work was supported by the Portuguese Fundação para a Ciência e Tecnologia (project POSC/EIA/56918/2004).

References

[1] Jorge M. Santos, Joaquim Marques de Sá, and Luis A. Alexandre. LEGClust - a clustering algorithm based on layered entropic subgraphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. submitted.

[2] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[3] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[4] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.

[5] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

[6] L. A. Alexandre, A. C. Campilho, and M. Kamel. Bounds for the average generalization error of the mixture of experts neural network. In *5th International Workshop on Statistical Techniques in Pattern Recognition*, LNCS 3138, pages 618–625, 2004.

[7] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, (3):pp.79–87, 1991.

[8] G. Auda and M. Kamel. Modular neural network classifiers: A comparative study. *Intel. Robotic Systems*, 21:117–129, 1998.

[9] M. Jordan and R. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, (6):181–214, 1994.

[10] R. Jacobs, F. Peng, and M. Tanner. A bayesian approach to model selection in hierarchical mixtures-of-experts architectures. *Neural Networks*, 10(2):231–241, 1997.

[11] R. Vilalta, M. K. Achari, and C. F. Eick. Class decomposition via clustering: a new framework for low-variance classifiers. In *Third IEEE International Conference on Data Mining*, pages 673–676, 2003.

[12] L. A. Alexandre, A. C. Campilho, and M. Kamel. A probabilistic model for the cooperative modular neural network. In *Proceedings of the IbPRIA 2003*, LNCS 2652, pages 11–18, 2003.

[13] Abdellatif Ennaji, Arnaud Ribert, and Yves Lecourtier. From data topology to a modular classifier. *International Journal on Document Analysis and Recognition*, 6(1):1–9, 2003.

[14] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[15] A. W. Bowman and A. Azzalini. *Applied Smooting Techniques for Data Analysis*. London:Oxford University Press, 1997.

[16] Joaquim Marques de Sá. *Applied statistics using SPSS, STATISTICA and MATLAB*. Springer, 2003.

[17] C. Blake, E. Keogh, and C. Merz. UCI repository of machine learning databases, 1998.

[18] M. Forina and C. Armanino. Eigenvector projection and simplified non-linear mapping of fatty acid content of italian olive oils. *Ann. Chim. (Rome)*, 72:127–155, 1981.