Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

Behavioral Partitioning in a Hierarchical Mixture of Experts using K-Best-Experts Algorithm

Mahdi Milani Fard
ECE department,
University of Tehran, Iran
m.milanifard@ece.ut.ac.ir

Amir-Hossein Bakhtiary
ECE department,
University of Tehran, Iran
a.bakhtiary@ece.ut.ac.ir

Abstract

In recent years methods for combining multiple experts (Multi Expert Systems, MES) have been used to solve different problems of classification and regression. In particular Hierarchical Mixture of Experts (HME) has been widely studied. This paper presents a novel method which divides the problem space into behaviorally portioned subsets using K-Best-Experts algorithm and then uses the HME structure to assign an expert to each subset. The gates used in the HME structure are Support Vector Machines which are trained to route each problem to the best fitting expert. The method is implemented and tested on the DELVE[1] framework and is compared with other similar methods.

Keywords: Multi Expert, Hierarchical Mixture of Experts, Behavioral Partitioning

1. Introduction

In the last decade there has been a great deal of study on different aspects of Multi Expert Systems (MES). MES is proved to produce much better results compared to single expert systems [1,2,4,6,7]. Many works have focused on the fusion mechanisms used to combine the results of different experts [10-17] while some others focused on different partitioning mechanisms [7,8,9]. The experts used in MES models can be of any type. Most studies, however, have focused on classifier combination methods.

There are two different views toward MES. Some methods try to use different experts with different structures and learning parameters and train all of them on the same domain. Then they use fusion mechanisms to combine the results of different experts and produce a more confident result. This method is usually useful with classifiers where one can calculate the confidence level of each expert. With enough data available, statistical methods can be used in fusion of the results [10,15,16].

Some other works, on the other hand, try to divide the problem space into smaller subsets and then use a different expert for each subset [1,7,8,9]. Here, the experts might be homogeneous or heterogeneous. This paper deals with this type of MES.

Hierarchical Mixtures of Experts (HME) are among the most studied types of MES structures [3,4,5]. HME consists of a tree structure system in which leaves are problem experts and internal nodes are combiners or gates (Figure 1). Most HME systems use simple models both for experts and gates. A common choice is a Generalized Linear Model (GLIM) which is a linear model with a single nonlinearity at the output:

$$y_i = f(U_i x) \qquad (1)$$

where U_i is a weight matrix and f is a fixed continuous nonlinearity [3].

A gating node in an HME could be either a classifier or a combiner which guides the selection or fusion mechanism respectively. Some methods use a simple weighted sum [1,3,12], and some others use linear classifiers [1].

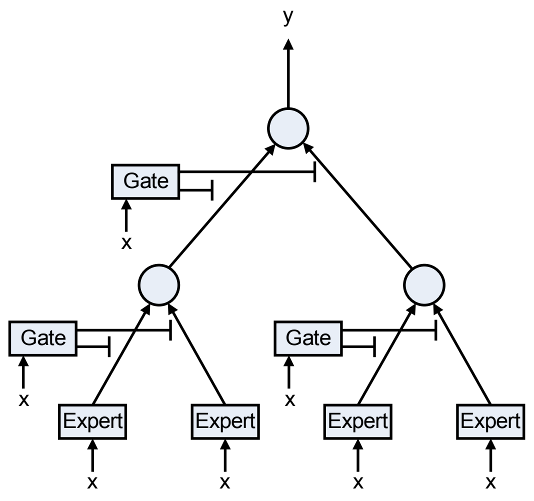

Figure 1. Hierarchical Mixture of Experts

[1] DELVE - Data for Evaluating Learning in Valid Experiments; developed at University of Toronto.

Both experts and gates are usually simple models to simplify the training methods. This is because in a conventional HME, the parameters of the experts and gates are correlated and thus cannot be adjusted separately. Even with simple GLIM experts and GLIM gates, the equations are so complicated that simple gradient descent is too slow for practical usage and might not provide good results [3]. Various methods, such as EM algorithm [3,4], are developed to overcome this problem for simple structures. However with the simple models, the structure has a limited capacity of generalization.

Even with simple learning models like GLIM's, HME has shown to be quite useful. The results have shown that HME is among the best ensemble method developed for learning problems considering both the response time and the quality of the results [1].

Here we focus on an HME structure in which gates are general selectors (classifiers) and experts are general learning models. We use complex models both for experts and gates and provide means to train them. The HME discussed here is used on regression problems, though it can be slightly altered to work on classification domain as well.

Unlike other methods used in HME structures, the method presented in this paper tries to separate different steps of training for a regression problem. First we use behavioral partitioning to divide the problem space and train the experts accordingly. Then we use a greedy algorithm to create the hierarchy structure and finally we use SVM to train the gates.

One might argue that these steps are inherently related to each other and their parameters cannot be set separately. Although it might be difficult to prove, behavioral partitioning seems to functionally separate these steps. This works around the assumption of an optimum partitioning, which lets us ignore the relations between different steps.

2. Behavioral Partitioning in HME

A. *Behavioral Partitioning*

Here we use a method for dividing the problem space into subsets each of which is handled by a separate expert. Behavioral partitioning is used here as the main approach, which uses the results produced by the experts as a means to divide the problem space [7,9].

Here the K-Best-Expert algorithm (KBE) is used. KBE slightly mimics a method used in Vector Quantization problem (VQ).

The algorithm is as follows:

```
1) Train each expert with a few random
   samples from the training data.

2) For each pair ($x$, $y$) in the training
   data:
     a) Feed $x$ to all experts and get
        the result $y_i$ of each expert.
     b) Find the $k$ nearest results to
        the target $y$ and train the
        corresponding experts with the
        pair ($x$, $y$).

3) Repeat step 2 for $n$ times.
```

Figure 2. K-Best-Experts algorithm

The algorithm starts by training the experts with random samples. This will result in a set of semi trained experts all of which can then compete to handle different parts of the input space. This prevents the problem of dead agents with behavioral partitioning [7,9]. The algorithm then simply uses each sample to train the k best fitting expert. With k bigger than 1, this introduces soft margins in the partitions.

As the process continues each expert gets trained in a subset of the problem space for which it is best fit. To prevent the problem of over-fitting we use early stopping method (not included in the above algorithm) by dividing the training data into two parts, one of which is used as the validation data for early stopping. This validation data is also used in the next steps of the training.

Although we have not trained a selector here, the problem space is implicitly divided into regions which are behaviorally separated. This method is used in a few other works and seems to produce good results [7,9].

In KBE, experts can be thought as learning agents, each trying to maximize its usage. The random bias at the beginning of the algorithm spreads these agents randomly in the problem space. The agents compete to produce the correct target value for each sample and the best k agents are awarded by letting them use the sample as the training data. If k in KBE is small, the agents are trained locally. However, if k is a large number most agents are trained with the same data and the resulting system is no better than a single agent one. Of course with heterogeneous agents and for classification purposes, this might be a good choice [10,11,12].

B. *Creating The Hierarchy*

The next step is to create a hierarchy for the HME structure. The idea used here is to create a hierarchy which will result in the minimum possibility of a miss-selection. For that, we need to create a structure in which wrong decisions at the gates will result in the selection of fairly similar experts to the optimum one. We do this by placing similar experts next to each other in the hierarchy. Consequently, even if the gating mechanism does not work properly, there is still a chance that a good expert is chosen.

The algorithm used here is in parts similar to the well-known Huffman compression algorithm. The algorithm is as follows:

```
1) For each expert allocate a node (a
   leaf).

2) For each pair (x, y) find the 2 best
   experts and create a relation link
   between the corresponding nodes.

3) While there are more than one
   unmarked node left:
   a) Find the pair of nodes with
      maximum number of links and
      combine them: create a parent
      internal node (a gate) and place
      the pair under this node. The
      links are then moved to the
      parent node and the child nodes
      are marked.
```

Figure 3. Hierarchy creation algorithm

The algorithm first creates *relation links* between experts trained in the last phase. These links are used as a measure of similarity between experts. Large number of links indicates that the experts work on similar domains. So the algorithm tries to place such nodes next to each other in the hierarchy. The last step of the algorithm finds the most related nodes and combines them into an internal node which is a selector gate.

This simple greedy algorithm results in a tree structure in which the distance between heavily linked experts is minimized. Although this does not give an optimum solution, it is sufficient for its application in HME structure.

C. Selection Mechanism

The next step is to train the gates at the internal nodes of the HME structure. As we separated the expert training phase, we can use any type of classifier here. Even complex models such as Support Vector Machines can be easily used and trained in such structure. Regardless of the type of the classifier we use, the algorithm to train the gates is fairly simple:

```
1) For each pair (x, y) find the best
   experts.

2) Identify the path from the root of
   the hierarchy to the best expert and
   train only the gates on the path to
   select the right choice.
```

Figure 4. Gate training algorithm

The samples here could be only the validation data from the first phase of training or the whole training data provided to the HME. By choosing the validation data as the input to this phase, we eliminate the chance of over-training of the selectors.

Here only the gates on the path from root to the best experts are trained. The gate at the root node is trained with the all samples; whereas as the nodes in other levels are only trained with small portion of the data. This is suitable as the selectors are trained only locally with the samples that their child experts are fit to work with. This works around the idea of local learning algorithms [7,9,21,22]. However, this causes a problem if the input to the internal nodes is sparse and the training might not be possible. This is a common problem with *divide and conquer* algorithms, which is due to the fact the dividing the samples usually increases the variance of the data [3]. In such cases, we might want to provide the sample to more than a single path, from the root to the *k* best experts.

If we use SVM as the selector gate, then we can gather the training data and train the SVM at the end with all the related samples. This would be a fast method as SVM is shown to be fast and reliable for selection mechanisms.

To ensure the performance of the algorithm we will do a brief study on the expected depth of an expert in the HME hierarchy. The depth of an expert will be the number of selectors on the path from the root to the expert node. If the number if proportional to *log(n)* where *n* is the number of experts, then we could ensure that the there is a small number of selectors on the way, and thus there is a small chance of a miss-selection.

To compute the expected depth of an expert, we shall make an assumption about the way the algorithm works. We assume that in each step of the merging, it is equally likely to choose any pair of unmarked nodes in the working set. We can especially use this assumption in a modification of the algorithm in which the links are weakened after a merge by reducing the number of links for the parent node.

Using the above assumption, we introduce a random variable D_i which is the depth of the expert E_i in the hierarchy. We want to find E[D_i]. We use an indicator random variable I_i which is defined as follows:

$$I_i = P\{\text{Expert } E_i \text{ is in one of trees merged at step } i\} \quad (2)$$

In step *i* there are *n-i* nodes in the working set. One of these nodes is the root to the sub-tree which contains expert E_i. Using the assumption mentioned above, we calculate the probability that this node (a particular node among the *n-i* unmarked nodes) is selected in step *i*:

$$I_i = \frac{\binom{n-i-1}{1}}{\binom{n-i}{2}} = \frac{n-i-1}{\frac{(n-i)(n-i-1)}{2}} = \frac{2}{n-i} \quad (3)$$

Each time the tree containing exert E_i is selected in the merging step, D_i is increased by one. So we could use the indicator random variable I_i to calculate D_i:

$$D_i = \sum_{i=0}^{n-1} I_i = \sum_{i=0}^{n-1} \frac{2}{n-i} = 2\sum_{i=1}^{n} \frac{1}{i} \quad (4)$$

Using an integral approximation we get:

$$D_i \approx 2\int_1^n \frac{1}{x}dx = 2\lg(n) \quad (5)$$

This indicates that the expected depth of an expert is proportional to the logarithm of the number of experts used in the HME structure, which further proves the efficiency of the algorithm.

D. HME usage

Now that we have trained both the experts and the selector gates, we can use the HME for the regression problem. Given and input *x*, we first feed it to the root element of the HME. As a selector, the root will choose one of its children as the desired one. Then we feed *x* to the chosen child and the process goes on, until we reach a leaf in HME which is a problem expert. Finally we use the selected expert to produce the final result.

It is important to notice that here only one of the experts is used for each problem. This is not the case for other models in which gates are combiners like the ones that use adding or voting mechanisms in the gates [1]. With complex models using combiner gates is not a good choice as it may need all the experts to produce the output, which might need a lot of computation or may require parallel computing.

3. Implementation Results

The HME structure proposed here was implemented in C++ using the *annie* [23] and *svmtl* [24] libraries with a 3 layer MLP neural networks for the experts and SVM$_c$ with RBF kernels as the selector gates. The implementation is a general purpose one and is intended to work well in the areas where there is a little knowledge about the problem domain. The work is tested on the DELVE [25] framework which provides a set of datasets and tools to test learning algorithms. DELVE is especially useful for comparison purposes and statistical analysis of results. For regression purpose, it provides both natural and synthetic test cases. For each test case, it provides a training set (a set of training pairs (x, y)) and a testing set, along which the method is tested to see how good it is to predict the target values. Well known learning methods have been developed and tested on DELVE and the results are easily available and can be used for comparison [1].

The work is tested on a few datasets of the DELVE framework. For different datasets, settings and parameters of the experts were chosen according to the number of training samples and the problem size (Table 1). To test different aspects of the proposed method, a variation of the method with an optimum selection mechanism (OSM) is also tested. OSM uses the actual target values in the selection mechanism and returns the nearest result of experts to the target. This is not a real regression system, as it uses the result, but it's a good means to study different parts of the algorithm separately.

The system was tested on both natural and synthetic test cases with different characteristics. The following datasets were used in the comparison [1]:

- **boston**: The Boston housing data contains information about housing in the Boston Massachusetts area collected by the United States Census Service.
- **kin**: The Kin family of data sets consists of different data sets which share the same model but which have different numbers of inputs and differing levels of noise and degrees of linearity. The model for the Kin family is a simulation of the forward dynamics of an 8 link all-revolute robot arm. The task in all data sets is to predict the proximity of the end-effectors of the arm from a target.

DELVE framework provides tools to find the standardized estimated expected loss of a regression mechanism. These values are calculated for different methods and can be used as a means to compare their performance.

TABLE I
SYSTEM PARAMETERS FOR DIFFERENT DLEVE TEST CASES

	INPUT OUTPUT	*SIZE OF TRAINING SET*	*NUMBER OF EXPERTS*	*K IN KBE*	*INTERNAL NODES IN MLP EXPERS*
boston	8/1	32	**4**	**2**	**5**
		64	**8**	**2**	**5**
		128	**10**	**2**	**5**
kin-fm linear	32/1	64	**14**	**3**	**8**
		128	**12**	**3**	**8**
		256	**10**	**3**	**8**
kin-nm nonlinear	32/1	64	**14**	**3**	**12**
		128	**12**	**3**	**12**
		256	**10**	**3**	**12**

There many different methods implemented on the DELVE framework for the test cases we chose for this paper. We chose a few methods among them that are closely related to the area we are working on [1]:

- **kbe**: K-Best-Expert Algorithm in HME structure
- **osm**: Optimal Selection Mechanism for KBE
- **knn-cv-1**: K-Nearest-Neighbor model selected by leave-one-out cross validation
- **hme-el-1**: A committee of hierarchical mixtures-of-experts trained by early stopping
- **mlp-ese-1**: A committee of multi-layer perceptions trained with early stopping
- **lin-2**: A generalized linear model for classification or regression in which the parameters were trained using the *macopt* [1] conjugate gradient

The results of the tests on the DELVE framework are summarized in Table I,II, III and IV. The values presented here are *standardized squared error loss* (see DELVE[25] for concrete definition and the significance levels) for each learning method which is relatively a good measure for comparison between different methods.

The results indicate that the system works well in problems with complex nonlinear behaviors. But for most test cases, it produces relatively poor results.

The OSM system, on the other hand, seems to highly outperform all the tested methods, even with small number of experts. This is not surprising as the method uses the target values to guide the selection. However, it also shows that the behavioral partitioning mechanism used to train the experts, is a good method and the problem lies down on the selection mechanism used in KBE.

TABLE II
RESULTS ON THE BOSTON DATA SET

Boston/Price	method	**32**	**64**	**128**
	kbe	0.405	0.239	0.206
	osm	0.124	0.061	0.027
	knn-cv-1	0.522	0.425	0.344
	mlp-ese-1	0.407	0.258	0.210
	hme-el-1	0.322	0.210	0.162
	lin-2	0.525	0.326	0.281

TABLE III
RESULTS ON THE KIN-32FM DATA SET

KIN-32fm/dist	method	**64**	**128**	**256**
	kbe	0.290	0.201	0.165
	osm	0.021	0.016	0.011
	knn-cv-1	0.705	0.618	0.544
	mlp-ese-1	0.187	0.124	0.105
	hme-el-1	0.272	0.181	0.136
	lin-2	0.182	0.124	0.107

TABLE IV
RESULTS ON THE KIN32-NM DATA SET

KIn-32NM/dist	method	**64**	**128**	**256**
	kbe	1.112	0.898	0.803
	osm	0.183	0.104	0.086
	knn-cv-1	0.965	0.926	0.910
	mlp-ese-1	0.936	0.886	0.813
	hme-el-1	1.115	0.919	0.846
	lin-2	1.444	1.001	0.847

KBE works poorly when the size of training data is small. As stated before, this is due to the problem of sparse input data to the selector gates. Results on the *kin-32fm* dataset (fairly linear problem with small noise) show that KBE also works poorly on a system with linear behavior. Using liner (or generalized linear) experts might help overcome this problem. Though, it further shows that the overall performance of the system is not as good as the ***hme-el-1*** method in terms of the errors on the datasets.

4. Conclusion

The method introduced in this paper tries to overcome the problems of HME learning algorithm by separating different learning steps. With the use of behavioral partitioning, there is no need to have the selection model trained along with the experts. This helps simplify the learning methods and yet creates the chance to use more complex learning models both for experts and gates.

Although there are many well established concrete mathematical solutions to the training problems of HME structures, ad-hoc methods such as the one proposed here might be useful in domains where simple mathematical models do not suffice or their assumptions are not valid. Complex experts can highly improve the learning capacity of the system which might be needed in complex highly nonlinear problems.

Although the implementation here uses homogeneous experts at the leaves of the hierarchy, the structure can hold totally heterogeneous experts. A mixture of linear experts and MLP neural networks might be useful if the problem shows non uniform behavior in different subsets of the problem space. This can be seen as an obvious future work on this paper which needs further considerations in the algorithms.

The behavioral partitioning method used here needs to be theoretically studied. This would generally be a game theory problem in which experts compete to handle different subsets of the problem space. It should also be checked to see if there is a Nash Equilibrium point where the ensemble gets to a stable state [9].

References

[1] Waterhouse, S.R., "Classification and regression using mixtures of experts", Ph.D., Thesis, Department of Engineering, Cambridge University, 1997.

[2] G. Valentini and F. Masulli, "Ensembles of learning machines", In M. Marinaro and R. Tagliaferri, editors, 13th Italian Workshop on Neural Nets, volume 2486 of Lecture Notes in Computer Science, pages 3--22. Springer-Verlag, 2002.

[3] Jordan, MI, & Jacobs, RA, "Hierarchical mixtures of experts and the EM algorithm", Neural Computation, 1994, 6, 181-214.

[4] Hinton, G. E., B. Sallans and Z. Ghahramani, "A hierarchical community of experts", In: Learning in Graphical Models (M. I. Jordan, Ed.). 1998, pp. 479-- 494. Kluwer Academic Publishers.

[5] M.I. Jordan and L. Xu, "Convergence results for the EM approach to mixtures of experts architectures", Neural Networks, 8:1409--1431, 1995.

[6] R. P. W. Duin, "The combining classifier: to train or not to train?", In Proc. of the 16th Intl. Conf. on Pattern Recognition - ICPR 2002.

[7] Mahdi Keramati, "Competitive Behavioral Partitioning of the Input Space for Local Experts", ECE Symp, University of Tehran, 2003

[8] R. Sun and T. Peterson, "Automatic partitioning for multi-agent reinforcement learning", From Animals to Animats: Proceedings of the International Conference of Simulation of Adaptive Behavior (SAB'2000). Paris, France. MIT Press, Cambridge, MA. 2000.

[9] Mahdi Milani Fard, "A Coevolutionary Competitive Multi-expert System for Image Compression with Neural Networks", In Proc of IEEE Intl. Conf. on Engineering of Intelligent Systems, Pakistan, 2006.

[10] L.I. Kuncheva, "A Theoretical Study on Six Classifier Fusion Strategies", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 2, pp. 281-286, Feb. 2002.

[11] Huang, Y.S., Suen, C.Y., "A Method of Combining Multiple Classifiers: A Neural Network Approach", ICPR94(418-420). BibRef 9400

[12] Giorgio Fumera, Fabio Roli, "A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems", IEEE Trans. Pattern Anal. Mach. Intell. 27(6): 942-956 (2005)

[13] Giacinto, Roli F. Dynamic classifier selection based on multiple classifier behavior. Pattern Recognition, 2001,34 (9)

[14] Marco F. Duarte and Yu-Hen Hu, "Decision Fusion in Collaborative Sensor Networks"

[15] Ludmila I. Kuncheva, "Switching between selection and fusion in combining classifiers: an experiment", IEEE Transactions on Systems, Man, and Cybernetics, Part B 32(2): 146-156 (2002)

[16] Kittler, J.V., Alkoot, F.M, "Sum versus vote fusion in multiple classifier systems", PAMI(25), No. 1, January 2003, pp. 110-115.

[17] Luigi P. et-al, "Optimizing the Error/Reject Trade-Off for a Multi-Expert System Using the Bayesian Combining Rule", SSPR/SPR 1998: 716-725

[18] Yea S. Huang, Ching Y. Suen, "A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals", IEEE Trans. Pattern Anal. Mach. Intell. 17(1): 90-94 (1995)

[19] V. Petridis, et-al, "A Bayesian Multiple Models Combination Method for Time Series Prediction", Journal of Intelligent and Robotic Systems, v.31 n.1-3, p.69-89, May -July 2001

[20] Matthias Rychetsky, et-al, "Application of Hierarchical Mixture of Experts Networks to Engine Knock Detection", 5th European Congress on Intelligent Techniques and Soft Computing, September 08. - 11, 1997

[21] Dietrich Wcttschcrcck and Thomas Dicttcrich, "Locally adaptive nearest neighbor algorithms", In Advances in Neural Information Processing Systems 6, pages 184-191, San Mateo, CA, 1994. Morgan Kaufmann.

[22] Bottou, L., & Vapnik, "Local learning algorithms", Neural computation, 4(6), 888900.

[23] ANNIE – Artificial Neural Network Library. http://annie.sourceforge.net/

[24] LIBSVMTL - a Support Vector Machine Template Library. http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/

[25] DELVE - Data for Evaluating Learning in Valid Experiments. Developed at University of Toronto. http://www.cs.toronto.edu/~delve/