Applying Parallel Learning Models of Artificial Neural Networks To Letters Recognition From Phonemes

Byoung Jik Lee
Department of Computer Science
University of Iowa
Iowa City, IA 52242
bjlee@cs.uiowa.edu

Abstract

A Parallel Artificial Neural Networks approach which converts strings of phonemes to strings of letters is described. Our approach employs several neural networks, each of which handles a different set of the input data with different weights. This approach is characterized by two important properties. First, this approach provides a way to handle the local minima problem without complex computations by generating multiple neural networks working in parallel. Neural networks are trained on examples iteratively which the previous neural networks could not produce proper results. Second, the test data is distributed across the learning networks according to the ability of each network to handle its assigned task. From among multiple outputs generated by trained neural networks, selection strategies are applied. Performance evaluated over 5,528 characters taken from the 934 most-used English words shows that our approach achieves more than 10% improvement over the general multilayered neural network (89.1% for our method, compared with 78.6% for the general network). In the training set, our approach achieves a correctness rate of 95.4%, while the general multilayered network has a correctness rate of 92.7%.

Introduction

The general backpropagation learning algorithm (Rumelhart et al. 1986) is a well-known training algorithm for a multilayered feedforward artificial neural network. This algorithm works successfully in most cases, but it can be stuck in local minima, since it uses a gradient descent method to reduce errors. Many efforts have been made to reduce the possibility of local minima, however many approaches require significant amounts of computation (Parker 1987)(Hertz et al. 1990). By providing more hidden units, the chances of local minima can be reduced. But, as the number of units in the neural network increases, overfitting problems may arise. This problem affects the generalization performance. To determine the size of the neural network optimally, there are two main approaches. First, network construction is that it starts with a minimal neural network which does not have a hidden unit, and adds new hidden units one by one (Fahlman 1990). Second, network pruning is that from a large amount of redundant information in the weights of a fully connected multilayered neural network, the size of a neural network is reduced by deleting weights selectively through second-derivative information (LeCun 1991). Our approach is a kind of a cascaded neural network. Instead of performing heavy computation, our approach employs several neural networks, each of which handles a different subset of the input data with different weights. A neural network in our approach determines the output only for a subset of the data, as opposed to a single neural network that handles all incoming data. Each neural network filters out only those cases which it can handle and passes on the other cases to the next sub network. Speech recognition has been one of the most studied subjects using neural networks (Lippman 1989). Our approach has been successfully applied to the speech processing problem.

Parallel Learning Artificial Neural Networks

Architecture

This approach creates a sub network when the main network has little confidence in its decision and thus needs another solving algorithm. The architecture consists of a main neural network and generated sub neural networks as shown in Figure 1. The main neural network and sub neural networks share the input, and each network has its own hidden and output layers with corresponding weights. The main neural network handles most of the cases and sub neural networks handle the remaining parts of incoming data. For this mapping of phonemes to letters, the main neural network and sub neural networks share an input layer which

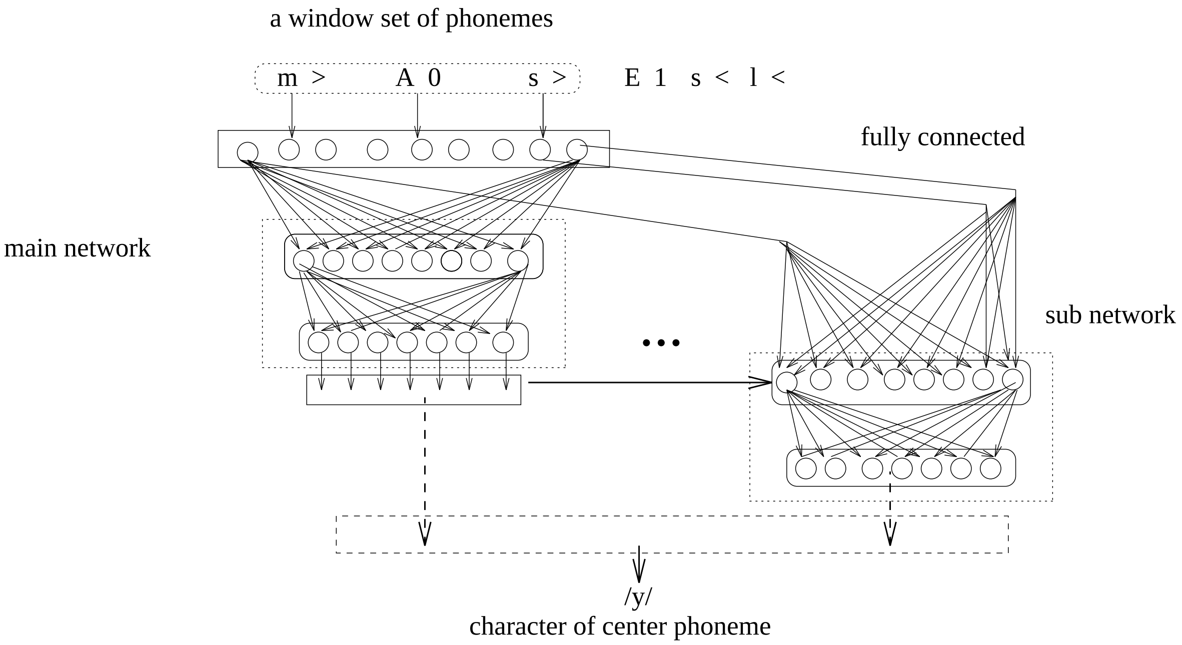

Figure 1: A hierarchical architecture. The input is the string of phonemes(`m > A 0 s >`) and the output is center letter y. This is an example of $myself$. Each circle represents a set of nodes.

consists of three groups, each with 57 units. Each group consists of a phoneme and a stress. Each network has 40 units in its hidden layer and 26 units in its output layer. The units are fully connected between adjacent layers. The phoneme and stress are represented locally in the input units. The window size is three phonemes, selected from a set of phoneme representations. Our goal is to recognize the center character from this three phonemes window.

Generating and Integrating Parallel Learning Models

Generating Parallel Learning Models The procedure for generating multiple networks in a hierarchical manner is described in Figure 2. We set the confidence level c to extract unsatisfactory data set from the current training data set and to construct a sub neural network. The pool size ($sizep$) is the minimum number of unsatisfactory data needed to build the next sub network. When $sizep$ is too small relative to the original training data set, there will be too many sub neural networks and this is not good because overfitting might occur. When $sizep$ is too big, it will be difficult to have an efficient sub network. The selection of appropriate values for $sizep$ and c is an interesting problem, which should be studied later. As an output function, the logistic function is used.

Integrating Parallel Learning Models Since there are several specially trained neural networks working in parallel, we need to choose an output among multiple outputs from the networks. We assign a preference vector $\mathbf{P} = (p_1, \ ..., \ p_n)$, where each p_i is the preference value for a network N_i and p_i = 0,1. The final output is chosen as the output with the best confidence among all the outputs with its preference vector of value 1. For example, suppose there is only one sub neural network with a main network with confidence level c of 0.7. When the confidence of output c_1^k is greater than 0.7 (where k is an instance of the test data set and 1 denotes the main neural network), p_1 is 1 and p_2 is zero. In this case, it is the same as a general multilayered artificial neural network. When the confidence of output c_1^k is less than or equal to 0.7, we have some choices. When p_1 is 0 and p_2 is 1, we choose the output from the sub network. The results are shown in Tables 2 and 4 as $ParallelNetwork_1$ with preference vector (0,1). When both p_1 and p_2 are 1, we treat the output equally and pick the result in which we have more confidence. The performance is shown in Tables 2 and 4 as $ParallelNetwork_2$ with

1. `Set arbitrary initial weights for the main network.`

2. `Train the main neural network using the training data set until it reaches an acceptance level (`$accept_1$`) for the main network.`

3. `Extract unsatisfactory data sets according to the confidence level `c`. Evaluate the confidence level of each training datum. If we have evaluated all of the training data set, go to 5. Otherwise, continue with step 4.`

4. `If the confidence of the training datum is satisfied, then go to 3. Otherwise, add the unsatisfactory training datum to the pool of data and continue with step 3.`

5. `Generate the sub neural network using a pool of the unsatisfactory training data set`

6. `Repeat steps 3 through 5 until the unsatisfactory data set reaches the minimum size, `$sizep$`, or until it reaches the limit of depth of generations, `$depth$`.`

Figure 2: The Sketch of Generating Multiple Networks Algorithm

preference vector (1,1). Because the size of sub training data set becomes smaller along with the depth of generations, overfitting problem might occur. One of the possible methods to handle overfitting is to reduce the number of parameters by considering the extracted unsatisfactory data size. The $ParallelNetwork_3$ with preference vector (0,1) and $ParallelNetwork_4$ with preference vector (1,1) have less number of parameters than $ParallelNetwork_1$ and $ParallelNetwork_2$. $ParallelNetwork_3$ and $ParallelNetwork_4$ have 20 hidden units.

Application and Results

Testing Procedure

For the problem of speech processing, Sejnowski proposed an efficient architecture called NetTalk which used a backpropagation algorithm to adjust weights in his architecture (Sejnowski 1987). NetTalk converts English text to speech. The reverse mapping, converting strings of phonemes to strings of letters, is used as a test bed for our proposed approach. The data set consists of the 934 most commonly used English words, selected from Sejnowski (Sejnowski 1987). Two cross over data (data set 1 and data set 2) are tested, where 748 words are used for the training data set and the remaining 186 words are used as a test data set. In data set 1, for a training data set, we picked 748 words in alphabetical order from the 934 words in the range $'able'$ to $'youth'$. In data set 2, for a test data set, we picked every fifth word from the set of 934 words. A simulator was written in C on an IBM RS/6000-530 system.

Results and Discussion

Performance The performance of our approach is better than a general multilayered neural network, as shown in Tables 1, 2, 3, and 4. In Table 1, Parallel Network$_2$ with preference vector $(1,1)$ and confidence level of 0.7 has a correctness rate of 95.4% in the training data set, compared with 92.7% for the general multilayered neural network. Table 2 compares the General Network, General Network$_h$, Parallel Network$_1$, Parallel Network$_2$, Parallel Network$_3$, and Parallel Network$_4$ for data set 1. General Network$_h$ has the same number of weights as the parallel networks, Parallel Network$_1$ and Parallel Network$_2$. In the Table 4, the comparison of General Network and Parallel Networks for data set 2 is shown. In both data sets, Parallel Network$_2$ has the best performance for the training data set.

For the test data set, Parallel Networks also show better performance than general multilayered neural networks. Our approach with Parallel Network$_2$ achieves a significantly higher correctness rate of 88.7%, compared with 78.6% for the general multilayered network. Parallel Network$_4$ with less number of parameters for the sub network has almost same performance as Parallel Network$_2$ in the training data set and its generalization performance on the test data set is better than Parallel Network$_2$, resulting in more than 10% improvement over the general multilayered network in the generalization.

Levels of Generations, Preference Vector, and Confidence Level In this approach, we can create as many levels as we like in the generation of hierarchical learning networks. In our investigation of the problem, we used a level of 2, and a p_i vector of (1,0), (0,1), (1,1). For this problem, (1,1) has the best result. The comparison of confidence level is shown in Table 3 and Figure 3. In Table 3, confidence level 0.9 shows the best performance in the generalization. In Figure 3, the *Error* contains the errors of the training

Epochs	*General Network*		*Parallel Multilayered Network*							
	Train	*Test*	*Train*				*Test*			
			0.3	0.6	0.7	0.8	0.3	0.6	0.7	0.8
30	4,094 chs	934 chs	3718	3651	3619(88.4%)	3590	788	806	771(82.5%)	783
60			3891	3825	3824(93.4%)	3796	805	816	806(86.3%)	804
100-1	3794(92.7%)	734(78.6%)	3913	3870	3855(94.1%)	3844	805	808	814(87.1%)	811
100-2			3912	3910	3904(95.4%)	3894	807	818	828(88.7%)	825

Table 1: The accuracy of character recognition for data-1. The training set is 4,094 characters and the test data set is 934 characters. The third row in the parallel multilayered network represents the level of confidence c. In 100 epochs of parallel multilayered network, 100-1 and 100-2 show the performance of Parallel Network$_1$ and Parallel Network$_2$ respectively.

Data Set 1	*General Network*	*General Network$_h$*	*Parallel Network$_1$*
train set: 4,094 chs	3794(92.7%)	3482(85.1%)	3855(94.1%)
test set: 934 chs	734(78.6%)	629(67.3%)	814(87.1%)
	Parallel Network$_2$	*Parallel Network$_3$*	*Parallel Network$_4$*
train set: 4,094 chs	3904(95.4%)	3856 (94.2%)	3892(95.1%)
test set: 934 chs	828(88.7%)	819 (87.7%)	832(89.1%)

Table 2: Comparison of General Network, General Network$_h$, $ParallelNetwork_1$, $ParallelNetwork_2$, $ParallelNetwork_3$, and $ParallelNetwork_4$ for the data set 1 when confidence level c is 0.7. For sub networks, $ParallelNetwork_3$ and $ParallelNetwork_4$ have less number of parameters than $ParallelNetwork_1$ and $ParallelNetwork_2$. The training data set is 4,094 characters (748 words) and the test data set is 934 characters.

Data Set 1	*Parallel Network$_2$*								
confidence c	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
train set: 4,094 chs	3919	3923	3912	3926	3919	3910	3904(95.4%)	3894	3894
test set: 934 chs	810	814	807	825	817	818	828(88.7%)	825	833(89.2%)

Table 3: The comparison of the confidence level c in the Parallel Network$_2$ for the data set 1.

Data Set 2			*General Network*		*$ParallelNetwork_1$*		*$ParallelNetwork_2$*	
	train	*test*	*train*	*test*	*train*	*test*	*train*	*test*
chs	3991	1037	3639(91.2%)	900(86.8%)	3724 (93.3%)	914(88.1%)	3776(94.6%)	932(89.9%)

Table 4: The comparison of performance for data set 2. The training data set is 3,991 characters (748 words) and the test data set is 1037 characters (186 words).

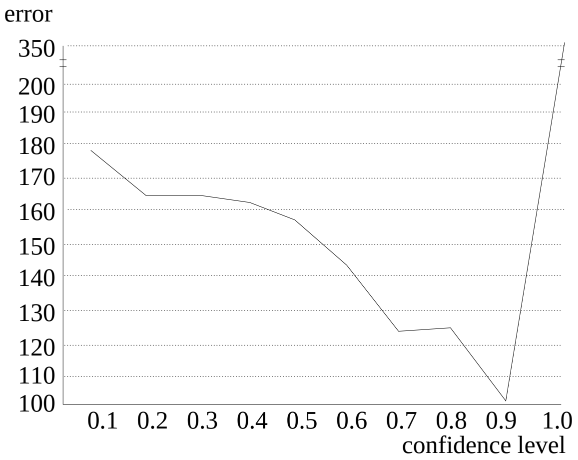

Figure 3: Comparison of Confidence Level. $Error = \frac{1}{2}\sum_{p,k}(target\ output_{p,k} - actual\ output_{p,k})^2$ where p refers pth incoming data vector and k refers kth output unit.

set and the test set. In this figure, the confidence level 0.9 shows a significant improvement compared with the confidence level 1.0 which is the same as a general multilayered neural network.

Vowel Recognition Particularly, most of the errors in the general multilayered neural network pertained to the vowel recognition. There are two kinds of errors. The overrated error with respect to a character e is that the occurrence of another character is identified as e. The underrated error with respect to a character e is that the occurrence of e is decided as another character. The combined errors of two cases is 27% in general network and our approach reduces the error to 15% in the vowel recognition. Our approach handles most of vowel recognition in the sub network, and increases the correctness rate of vowel recognition from 87% to 93%. This indicates that the rules for vowel recognition are quite different from those for consonant recognition. This also indicates the usefulness and effectiveness of this multiple hierarchical network approach.

Advantages of Our Approach

Our approach facilitates easy adjustment of the parameters. By selecting the confidence level, we can adjust the size of sub training data set and the depth of generations. Final output is decided by applying a preference to each of the network outputs. Each network works by filtering good instances and extracting some unsatisfactory instances to pass to the next sub network to handle. Our approach achieves better performance than a general multilayered neural network (Lee 94) for the speech recognition problem. In addition, our approach reduces training time when compared with a neural network which has the same number of hidden nodes.

Conclusions and Future Works

Our approach is characterized by the use of multiple specially trained neural networks working in parallel. Since each network explores a different search space, our approach avoids being stuck in a local minimum. More importantly, each network can be adapted to handle a *subset* of the problem rather than the whole problem, as seen in the speech recognition problem. [1] Our scheme of partitioning the input data according to the output confidence values provides a general way to create a subproblem and adapt each network more appropriately for the subproblem.

For the future works, the guidance of settings for important parameters confidence level and preference vector will be studied. Comparison of our algorithm with the boosting algorithm (Schapire 1990) is very interesting. The main difference between our algorithm and the boosting algorithm is how to filter data through the network and to generate a next data set for a next network. We plan to apply the boosting algorithm to the speech recognition problem and also apply our algorithm to the handwritten ZIP code problem to which the boosting algorithm was applied successfully (Drucker 1993). We hope that this experiment will give us the better understanding of the behaviors of the two algorithms.

We believe that our approach can be applied to many other areas.

Acknowledgements

I would like to thank to Terry Sejnowski for the data set.

References

Drucker H.; Schapire R.; and Simard P. 1993 Improving Performance in Neural Networks Using a Boosting Algorithm *Advances in Neural Information Processing Systems*, vol. 5, pp. 42-49.

Fahlman S. E. and Lebiere C. 1990 The Cascade-Correlation Learning Architecture *Advances in Neural Information Processing Systems* , vol. 2, pp. 524-532.

[1] Note that a sub network handles the vowel recognition problem better than the main network.

Hertz Z.; Krogh A.; and Palmer R. 1990 *Introduction to the theory of Neural Computation*, Santa Fe Institute, Addison Weisley, 1990.

LeCun Y. 1991 Optimal Brain damage *Advances in Neural Information Processing Systems*, vol. 3, pp. 598-605.

Lee B. J. 1994 Speech Recognition Using Neural Networks(HearNet) *Proceedings of the Third Golden West International Conference on Intelligent Systems*, vol. 1, pp. 525-530.

Lippman R. P. 1989 Review of Neural Networks for Speech Recognition *Neural Computation*, vol. l, pp. 1-38. MIT Press

Parker D. B. 1987 Optimal Algorithms for Adaptive Networks: second Order Back Propagation, Second Order Hebbian learning *Proceedings of the IEEE First Annual Conference on Neural Networks*, June, 1987.

Rumelhart D. E.; Hinton G. E. ; and Williams R. J. 1986 Learning Internal Representations by Backpropagation *Parallel Distributed Processing* , vol. 1 Cambridge, MA: MIT Press. pp 318-362.

Schapire R. 1990 The Strength of Weak Learnability *Machine Learning*, vol. 5, no.2 pp. pp 197-227.

Sejnowski T. J. and Rosenberg C. R. 1987 Parallel Networks that Learn to Pronounce English Text *Complex Systems*, vol. 1 no.1 pp. 145-168.