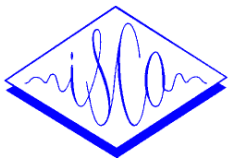

3rd International Conference on
Spoken Language Processing (ICSLP 94)
Yokohama, Japan
September 18-22, 1994

ADAPTATION OF NEURAL NETWORK MODEL: COMPARISON OF MULTILAYER PERCEPTRON AND LVQ

Dongxin Xu, Dao Wen Chen, Qian Ma, Bo Xu and Taiyi Huang

National Lab. of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences
P.O.Box 2728, Beijing 100080, P.R.China

ABSTRACT

This paper explores the adaptation features of Multilayer Perceptron and LVQ_3 under the speech phoneme recognition experiment. The result shows that Multilayer Perceptron can adapt to novel data easily but may lose the high performance on old data, and by contrast LVQ_3 can maintain what it has learnt in the past but is hard to adjust itself to fit the novel data if a relative great difference lies between the old and novel data. In order to gain a better adaptation algorithm, some modifications are added to LVQ_3 and better results are achieved under the same experiment. A method to add in new code vectors for LVQ_3 is also examined and a better result is also achieved. Finally, Some problems of the proposed algorithms are discussed.

1. INTRODUCTION

Neural Network approach has been applied to a variety of fields and has shown its power. In many cases, there exists the need to adapt a trained neural network model to a set of novel sample data so that it could improve its performance in the new case. It is highly hoped that a model can maintain what it has learnt in the past during its adaptation. To achieve the goal, one possible "Close Loop" method is to merge the original and novel data into a whole training set. But this is often impractical because of the too large size or even the unavailability of original data. Therefore, in such cases, "Progressive Training", an "Open Loop" method, is necessary which uses only novel data for its adaptation.

In speech recognition, "Progressive Training" is desirable for speaker adaptation in which original training data is often unavailable. In this circumstance, it is better that the recognition model can "accumulate its knowledge" during the adaptation, i.e. it does not lose its high performance on old speakers when it improves its performance on new speakers. If the recognition model can not keep its high performance on old speakers, it is, at least, expected to fit new speaker quickly by relative small amount of training data.

There has been many researches on Neural Network Speech Recognition, but few has been reported on their adaptation behavior. In this paper, two common Neural Network Model: Multilayer Perceptron and LVQ_3 are compared. Two successful phoneme recognizer: TDNN [1](Multilayer Perceptron Type) and Shift-Invariant LVQ [2] (LVQ_3 Type) are actually used in the phoneme recognition experiment. The experiment results are quite different but interesting. The analysis is made, and accordingly, some modifications are added into LVQ_3 algorithm. The new algorithms are tested under the same experiment. The following sections will give the details.

2. DATA BASE AND EXPERIMENT

The speech phoneme recognition task of this research is to recognize three very confusable Chinese initial consonant /b/, /d/, /g/. The 3 consonants are from 57 syllables in Chinese which cover almost all relevant syllables in Chinese regardless of syllable tone. There are ten speakers, 5 males and 5 females. Each speaker has four token for each of 57 syllables. The data of the male speaker "xdx" and the female speaker "gjl" are divided into two parts, one for training which is the set of one token of every 57 syllables and

marked by "(1)" in the following, and the other is for testing marked by "(2)". The data of other male and female speakers are just for testing. All speech data are hand-segmented into 15 frames.

TDNN model and Shift-Invariant LVQ_3 model are just as [1] and [2] except that LVQ_3 rather than LVQ_2 is used for the latter. The models are first trained by data "xdx(1)" and then by data "gjl(1)". The relative great different between the male speaker xdx's speech data and the female speaker gjl's speech data is thought to be suitable for testing the adaptation feature of the models. The recognition results are shown in Table 1 and Table 2. Figure 1 shows the performance trends during the models' adaptation.

Train	xdx (1	xdx (2)	other M	gjl (1)	gjl (2)	other F
xdx's	100%	92.4%	59.8%	68.4%	65.4%	62.1%
gjl's	87.7%	85.4%	63.9%	100%	82.4%	72.4%

Table 1: The Result of TDNN

Train	xdx (1	xdx (2)	other M	gjl (1)	gjl (2)	other F
xdx's	100%	98.8%	73.9%	71.9%	69.6%	72.8%
gjl's	100%	96.5%	76.6%	93.0%	88.9%	84.3%

Table 2: The Result of Shift-Invariant LVQ_3

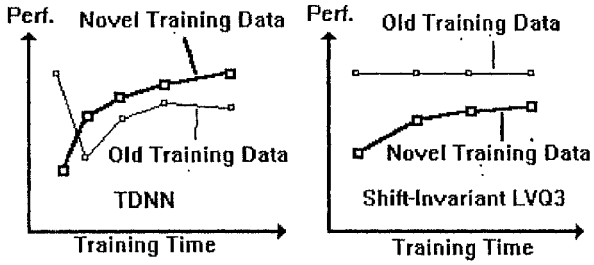

Figure 1: Trends of the performance during adaptation

3. ANALYSIS

From the above results, we can see that it is easy for TDNN, a kind of Multilayer Perceptron Model, both to adapt to new training data and to "forget" old data, while Shift-Invariant LVQ_3, one kind of LVQ_3, can keep the performance on old data but is hard to adjust itself to fit new data. Why this result?

As we all know, Back Propagation Algorithm, the basis of the training algorithm of TDNN, is a "Global Updating" one, in which all parameters of the model will be updated when error is back-propagated. This causes the great change of the parameters of the model. So, the effect of old training data is just to initialize the model. After such initialization, there is no any guarantee to keep the memory of the initial state. Therefore, Multilayer Perceptron is easy to "forget" old data, and at the same time, easy to adapt to new data.

By contrast, LVQ_3 algorithm, the basis of the Shift-Invariant LVQ_3 training algorithm, is a "Local Updating" one, in which only the nearest and the next nearest code vectors of the model have the chance to update. Such localized parameter updating gives some guarantee to keep the memory of the initial state of the model trained by old data.

To illustrate how "Global Updating" and "Local Updating" take their effects, a simple classification experiment is done. As shown in Figure 2, there are two classes of dots in 2-Dimensional Plane, one is Class A and the other is Class B. The dots of both classes are evenly distributed in Circle A and Circle B. The dots in Circle A' belongs to Class A and is also evenly distributed in Circle A'. A Multilayer Perceptron model with two hidden nodes and a LVQ_3 model with 3 code vectors are used to classify these two classes. First the dots in Circle A and Circle B are used to trained the models. And then the dots in Circle A' and Circle B are used as novel data for adaptation. The lines in Figure 2 are boundaries of two classes. Figure 2 gives the image to show how models change under "Global Updating" and "Local Updating" training algorithm.

Besides the "Local Updating", another point related to the adaptation behavior of LVQ_3 Model is the "Fine Tuning" nature of its training algorithm. In general, there are two steps in LVQ_3 training algorithm: 1. Model Initialization with Clustering Method; 2. Parameter Tuning under LVQ_3 algorithm. The first step gives the relative well-prepared parameters and the second one finely tune the parameters to achieve the optimum performance. In step 2, only those

training data which locate in some specific areas of Feature Space can be used to update the parameters of the model ,and further, such updating must be weighted by a decreasing factor. This "Fine Tuning" nature of LVQ_3, we think, is another guarantee to keep the memory of the initial state of its model.

With the "Global Updating" and "Fine Tuning" nature, LVQ_3 can maintain what it has learned, and just for these two points, it is hard to adapt to new data. In the above phoneme recognition example, even the performance on novel training data can not reach the above 95%.

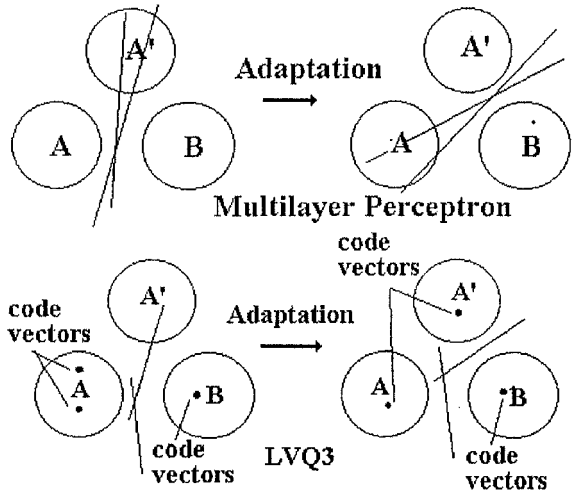

Figure 2: A Simple Classification Task.

4. MODIFICATION TO LVQ_3

As stated above, the "Fine Tuning" of LVQ_3 can drive the model to approach the optimum performance when the initial state of the model is well-prepared. But under "Progressive Training", it sometimes can not work well because there exist very different data in novel training set and "Fine Tuning" is not enough. In order to overcome this defect, based on LVQ_1 and LVQ_3, some modifications are added into the original LVQ_3 algorithm which can make more adjustment for the parameters and even add new code vectors for LVQ_3 Model when necessary. There are three kind of modifications. The central notion of the modification is to utilize those training data which may have no chance to participate in Parameter Updating because of the "Fine Tuning" nature of LVQ_3. Following will give the details:

Modification 1:

When input vector X disagrees with its two closest code vector M(i), M(j) then update code vectors according to following:

$M(i) = M(i) - a(t) (X - M(i))$

$M(j) = M(j) - a(t) (X - M(j))$

where a(t) is a decreasing factor.

In other cases, go according to original LVQ_3 algorithm.

Modification 2:

When input vector X disagrees with its two closest code vector M(i), M(j) then update code vectors according to following:

$M(i) = M(i) - a(t) (X - M(i))$

$M(j) = M(j) - a(t) (X - M(j))$

$M(k) = M(k) + a(t) (X - M(k))$

where a(t) is a decreasing factor. M(k) is the closest code vector of the same class as X.

In other cases, go according to original LVQ_3 algorithm.

Modification 3:

This is the method to add new code vectors in LVQ_3 Model. There are two steps:

1. Select X from novel training data set when following conditions are satisfied:
   X disagrees with its two closest code vector M(i), M(j), or X agrees with M(i) but disagree with M(j) and D(X,M(i)), the distance between X and M(i), is less than D(X,M(j)) and D(X,M(j)) / D(X,M(i)) > thr1. where "thr1" is a threshold and more than 1.
2. Use the selected data for clustering. Initially, give a small number of ,code vectors, then do clustering, if the average distortion is more than "thr2", then increase the number of code vectors and continue clustering until the average

distortion is less than "thr2". Where "thr2" is a threshold which can be a value comparable to the average distortion of the original model. The produced code vectors are new code vectors for the model

These three modified algorithms are tested by the same experiment task. Table 3, Table 4 and Table 5 show the results. Table 3 and Table 4 are all the results after 60 epochs' training. Table 5 is the result after 30 epochs' training. In Table 5 the model is first trained by the original LVQ_3 using "xdx(1)" as training data, and then added new code vectors by Modification 3 using "gjl(1)" as training data, and finally trained by the original LVQ_3 and data "gjl(1)". The number of code vectors of the three original models are all 30 for each class.

Train	xdx (1	xdx (2)	other M	gjl (1)	gjl (2)	other F
xdx's	100%	98.8%	73.3%	70.2%	69.0%	73.0%
gjl's	100%	97.1%	76.2%	98.2%	90.6%	83.7%

Table 3: The Result of Modification 1.

Train	xdx (1	xdx (2)	other M	gjl (1)	gjl (2)	other F
xdx's	100%	98.8%	73.5%	70.2%	69.0%	73.1%
gjl's	100%	95.3%	77.0%	96.5%	90.9%	82.6%

Table 4: The Result of Modification 2.

Train	xdx (1	xdx (2)	other M	gjl (1)	gjl (2)	other F
xdx's	100%	98.8%	73.9%	71.9%	69.6%	72.8%
gjl's	100%	98.8%	75.9%	98.2%	92.4%	77.0%

Table 5: The Result of Modification 3 and the original LVQ_3 algorithm.

Train	xdx (1	xdx (2)	other M	gjl (1)	gjl (2)	other F
xdx's	100%	97.1%	70.4%	63.2%	59.1%	67.0%
gjl's	98.2%	95.9%	76.8%	82.5%	77.8%	73.0%
*	94.7%	87.1%	73.5%	100%	92.4%	72.3%

Table 6: The Result of the model with only 7 code vectors for each class in the beginning.

* using Modification 3 and gjl(1) to add new code vectors, then using Modification 2 and gjl(1) to train the model

The experiment results show the effectiveness of the proposed modification. However, if the number of code vectors of the model is not large enough, then the conflict between keeping the high performance on old data and adapting to new data will be greater. Table 6 is an example.

5. CONCLUSION AND DISCUSSION

By comparing and analyzing the experiment results, we can get the following conclusion:

1. There exist some conflict between keeping high performance on old data and adapting to new data, especially when great difference lies between old data and new data. If the number of the parameters in a model is not large enough, such conflict may increase.

2. TDNN (Multilayer Perceptron) is easy to adapt to new data but may lose high performance on old data. Shift-Invariant LVQ_3 can maintain what it has learned in the past but is hard to adapt to new data. Modification 1 and 2 are effective in improving the LVQ_3's adaptation ability to new data. But when the number of code vectors is not enough, such improvement is limited. The generalization ability (performance on other data) of the above LVQ algorithms are greater than that of TDNN.

3. Modification 3 is not a very good method for adding new code vectors for LVQ_3, especially its generalization ability is not good. The reason may be its interference with the distribution of original code vectors and old data. So, taking the distribution of original code vectors and old data may improve this method.

REFERENCE

[1] Alexander Waibel, etal "Phoneme Recognition Using Time-Delay Neural Networks". IEEE Trans. ASSP Vol 37

[2] Erik McDennott, etal "Shift-Invariant Multi-Category Phoneme Recognition Using Kohonen's LVQ2" ICASSP 89

[3] Qian Ma "Chinese Initial Consonant and Final Vowel Recognition Based on Shift-Tolerant LVQ_3", Master's Thesis, Institute of Automation, Chinese Academy of Sciences.