4th European Conference on
Speech Communication and Technology
EUROSPEECH '95
Madrid, Spain, September 18-21, 1995

A COMPETITIVE ALGORITHM FOR TRAINING HMM FOR SPEECH RECOGNITION.

Pedro L. GALINDO
e-mail : plgr@cain.uca.es
Universidad de Cádiz
C/ Chile S/N
11003 Cádiz
SPAIN

ABSTRACT

In this paper we describe an alternative estimation procedure for training Hidden Markov Models (HMM) called Competitive Forward-Backward(CFB) algorithm, which is aimed at minimizing the number of recognition errors. CFB integrates the LVQ3 neural network classification technique into the Baum-Welch training algorithm, and improves significantly the performance of HMM systems over previously reported procedures, such as Baum-Welch and Corrective Training.

1. INTRODUCTION

Bahl et Al.[1] introduced the Corrective Training algorithm and compared it to an error-corrective training procedure for linear classifiers[2]. The results were a better recognition performance than either maximum-likelihood estimation via the Baum-Welch algorithm, or maximum mutual information estimation.

Learning Vector Quantization (LVQ) algorithm has shown[3] to be statistically superior to other classical (K-nearest neighbor, Parametric Bayes) or neural-like (Backpropagation network, Boltzmann machine) systems when applied to the classification of artificial or natural patterns. Several variants of LVQ have been proposed, but LVQ3 has shown to give consistently higher accuracy, when classifying speech spectra, than LVQ1 or LVQ2.[4]

Katagiri and Lee[5] used an hybrid algorithm between HMM and LVQ to improve significantly the performance of an original HMM-based speech recognizer. However, the LVQ module was considered independent of the HMM training process. What we are proposing is a complete integration of LVQ learning process into the training phase of HMM.

2. LEARNING VECTOR QUANTIZATION

2.1 LVQ1 Algorithm

LVQ algorithm is basically a supervised learning algorithm trained with "target" responses. The centroid with minimum distance from the pattern x(t) is assumed to classify the pattern. If the class matches that of the pattern, then the weight vector is moved closer to the pattern, otherwise it is moved further away. This is summarized as follows. Let C the nearest class to x(t), apply :

$$\text{if } x(t) \in \text{class } C,$$
$$m_c(t+1) = m_c(t) - \alpha(t)\cdot[x(t) - m_c(t)]$$

$$\text{if } x(t) \notin \text{class } C,$$
$$m_c(t+1) = m_c(t) + \alpha(t)\cdot[x(t) - m_c(t)]$$

$\alpha(t)$ is a decreasing monotonically function which represents the learning rate. One should start with a small value, say 0.01 or 0.02 and let it decrease to zero.

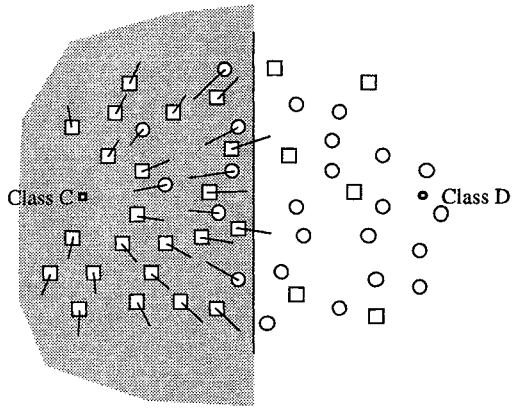

Figure 1. Complete set of values affecting to centroid of class C when LVQ1 algorithm is applied.

As shown in [4], the decision surface seems to be near-optimal, although piecewise linear. In Figure 1, we represent all the values

$$\Delta m_c(t) = \pm\alpha(t)\cdot[x(t) - m_c(t)]$$

that affects to centroid of class C. Squares represent patterns from class C and circles represent patterns from class D.

2.2 LVQ2 Algorithm

LVQ algorithm can easily be modified to better comply with Bayes' philosophy. If the classifying centroid is incorrect and the next nearest centroid is correct, and the pattern falls into a small window, both centroids are

modified. This process allows that decision boundaries asymptotically approach those expected from Bayes rule. This can be stated easily. If the following conditions are satisfied :

a) The nearest class to x(t) is incorrect (let c this class)

b) The next-nearest class to x(t) is correct (let d this class)

c) x(t) falls into a small, symmetric window defined around the midpoint between m_c and m_d, centroids for class c and d respectively.

then apply the reestimation transformations :

$$m_c(t+1) = m_c(t) - \alpha(t) \cdot [x(t) - m_c(t)]$$

$$m_d(t+1) = m_d(t) + \alpha(t) \cdot [x(t) - m_d(t)]$$

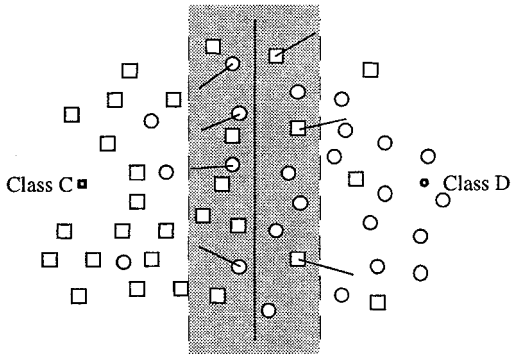

Figure 2. *Complete set of values affecting to centroid of class C when LVQ2 algorithm is applied.*

In Figure 2, we show all the values $\Delta m_c(t)$ that affects to centroid of class C. Those modifications correspond to all the patterns incorrectly classified, where the next nearest centroid is correct, falling into the window, and tend to modify the decision surface to approximate the Bayes frontier.

2.3 LVQ3 Algorithm

LVQ3 is the last modification proposed to LVQ algorithm. It was introduced to optimize LVQ2, and compensate the effects produced when long runs of the LVQ2 process are applied. In fact, the classification accuracy of the LVQ2 is improved at the beginning, when the decision surface approximates the Bayes limit, but drifts away after that. Therefore, the number of iterations of LVQ2 should be limited to a relatively short value. To compensate this effect, all the vectors from the window are accepted for training. The only condition for training is that one of the two closest codebooks belong to the correct class. These ideas are implemented easily. If the following conditions are satisfied :

a) x(t) belongs to class c,

b) c and d are the nearest classes to x(t)

c) x(t) falls into a small, symmetric window defined around the midpoint between m_c and m_d, centroids for class c and d respectively.

then apply the reestimation transformations :

$$m_c(t+1) = m_c(t) - \alpha(t) \cdot [x(t) - m_c(t)]$$

$$m_d(t+1) = m_d(t) + \alpha(t) \cdot [x(t) - m_d(t)]$$

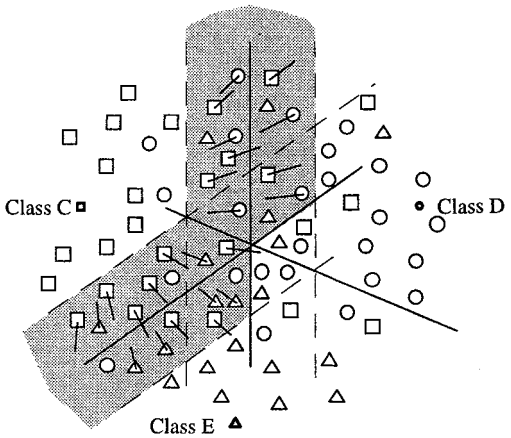

Figure 3. *Complete set of values affecting to centroid of class C when LVQ3 algorithm is applied.*

In Figure 3, we show all the values $\Delta m_c(t)$ that affects to centroid of class C. In this case, we have selected a three-class problem, to exactly represent what happens with those patterns located at critical zones.

3. THE CFB ALGORITHM

The Competitive Forward-Backward (CFB) algorithm introduces the LVQ classification technique into the Forward-Backward training process allowing the minimization of the misclassification rate[6].

If we consider each input sequence of data as a single pattern, and each class is defined to contain all sequences of data that belong to a given set of labeled data, the HMM recognition phase can be considered as a Pattern Classification problem. From this point of view, the HMM training phase can be considered as a Pattern Classification training problem. Given that LVQ3 has proven to be an excellent algorithm for Pattern Classification, we have adapted this algorithm, and applied it to HMM training.

First of all, since LVQ3 assumes a good initial state, the Baum-Welch algorithm can be used to obtain an estimate of the parameters of each model.

The CFB algorithm considers the value $-P(O,\lambda^i_j)$ as a distance measure between the observation sequence $O=(O_1,O_2,O_3,...O_T)$ and the model of phone *i* at iteration *j*. Therefore, each model can be considered as a *centroid*, and each observation sequence as a *pattern*. According to these statements, the CFB algorithm can be stated as follows :

1. Obtain an initial estimate of HMM model parameters for each phone by using the Baum-Welch algorithm on labeled data.

$$\lambda^i_0 = (A^i_0, B^i_0, \pi^i_0) \quad , \quad i=1,2....N$$

2. For each input sequence, (O), apply the Forward-Backward algorithm, to obtain a value of the distance of the *input pattern* (O) to each *centroid* at iteration *j*.

$$d(O,\lambda^i_j) = -\log P(O,\lambda^i_j) \qquad i = 1,\ 2,\ \ldots\ N$$

3. Obtain the two Hidden Markov Models with lowest values of $d(O,\lambda^i_j)$, i.e., the two *centroids* ($m_r=\lambda^r_j$ and $m_s=\lambda^s_j$) nearest to the *input pattern* O.

4. If the following conditions are satisfied :

A) O and m_r belong to the same class,

B) O and m_s belong to different classes

C) $\left|d(O,\lambda^r_j) - d(O,\lambda^s_j)\right| < \delta(t)$

(i.e. into a given "window")

then apply the reestimation transformations.

5. If the decrease in the overall error rate relative to the value at the previous iteration is below a selected threshold, STOP; otherwise go to Step 2.

The reestimation transformations can be easily derived, as shown in [6] from the Baum-Welch algorithm, and are simply an extension of those formulae.

As said above, suppose that the three conditions are met. Let ρ^r_{j-1} denote the model obtained after the application of the Baum-Welch algorithm to λ^r_{j-1}, given the observation sequence O. Let Let ρ^s_{j-1} denote the model obtained after the application of the Baum-Welch algorithm to λ^s_{j-1}, given the observation sequence O. Then, the reestimation formulae are applied :

$$\lambda^r_j = \lambda^r_{j-1} - \alpha(t)\cdot[\lambda^r_{j-1} - \rho^r_{j-1}]$$

$$\lambda^s_j = \lambda^s_{j-1} + \alpha(t)\cdot[\lambda^s_{j-1} - \rho^s_{j-1}]$$

The proposed equations reestimate the values of the models to be trained by observing which patterns are in the neighborhood of the frontiers defined by the nearest and next-nearest models. The value $\delta(t)$ defines the size of the window where patterns are considered confusable. $\alpha(t)$ is a monotonically decreasing scalar function, and defines the percentage of reestimation on each iteration. The initial values for $\alpha(t)$ and $\delta(t)$ must be determined experimentally.

4. EXPERIMENTAL RESULTS

Experiments were made on the DARPA TIMIT database, on the phonetic recognition of phonemes / ah - eh - ih /. We used a discrete HMM, one model per phone, with 7 states per model with tied transitions. This model was used with great success in [7].

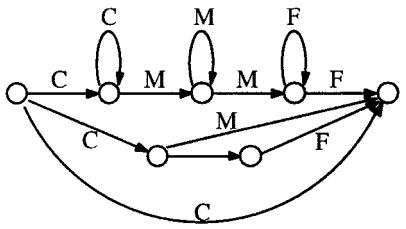

***Figure 4** : Phone model used in the experiments.*

All classes were vector quantized using all the segments of the phones / ah - eh - ih / included in the training subset. We have started with $\alpha=0.02$ and letting it to decrease to zero lineally in 10 iterations over the whole training set. An initial value of $\delta=2.5$ was selected, decreasing it lineally to 1.25. The results obtained for different VQ sizes are shown in Table 1.

	VQ Size = 32			*VQ Size = 64*			*VQ Size = 128*		
	B-W	***CT***	***CFB***	***B-W***	***CT***	***CFB***	***B-W***	***CT***	***CFB***
open	60.97	63.01	63.35	62.41	63.31	65.02	62.50	63.87	65.45
close	60.17	62.95	63.23	61.48	63.05	63.47	61.92	63.22	63.79

***Table 1.** Phonetic recognition rates for /ah-eh-ih/ phones with different VQ sizes with Baum-Welch, Corrective Training and Competitive Forward-Backward algorithms.*

As shown above, a comparative analysis between the classical Baum-Welch algorithm, Corrective Training procedure and CFB algorithm shows that CFB algorithm improved significantly the Baum-Welch one, and is slightly better than Corrective Training, both in close (over the training samples) and open (over the test samples) tests.

Moreover, we performed a similar experiment on 10 phones, namely */ ah - eh - ow - s - f - dh - m - n - l - r /* extracted from TIMIT database, to compare the evolution of recognition rate for each algorithm, i.e.

Baum-Welch, Corrective Training and Competitive Forward-Backward Training. We observed, as shown in Figure 5, that Baum-Welch algorithm improves only slightly the recognition rate. On the other side, just one iteration of CFB algorithm was enough to obtain better results than Corrective Training or Baum-Welch did in the eight first iterations.

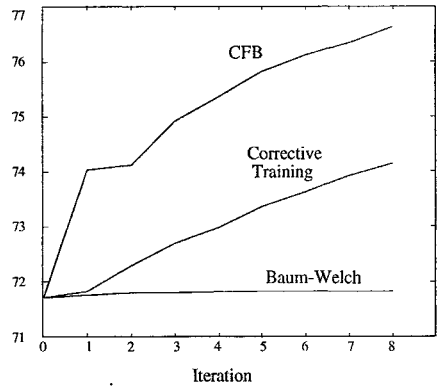

Figure 5. *HMM phonetic recognition rates for /ah-eh-ow-s-f-dh-m-n-l-r/ phones with B-W, CT and CFB algorithms for TRAINING set.*

If we analyze what happens (see Figure 6) when the test dataset is evaluated after each training iteration, Baum-Welch algorithm is absolutely inefficient in this task. Corrective training performs quite well, but CFB greatly outperforms the other two, after just one iteration.

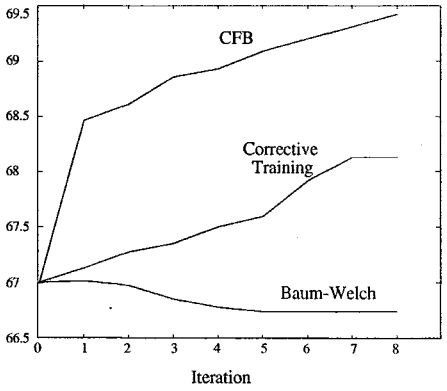

Figure 6. *HMM phonetic recognition rates for /ah-eh-ow-s-f-dh-m-n-l-r/ phones with B-W, CT and CFB algorithms for TEST set.*

The convergence of CFB algorithm is not proved but experimental evidence suggest that it does. In fact, convergence has been observed over the training set, but when the number of iterations is too high, the recognition rate obtained in the test set reaches a maximum, after which, it starts to decrease slowly. This effect is caused by an excessive adaptation of HMM models to the training patterns, and has been shown to be attenuated by increasing the size of the training dataset.

5. CONCLUSIONS

The solution proposed can be considered a variant of the Corrective Training procedure for HMM, where the error-corrective training for linear classifiers has been replaced by LVQ3. Given the optimal results obtained when LVQ3 is applied to pattern classification over other classical algorithms, is easy to understand why CFB improves greatly Corrective Training performance.

In summary, CFB algorithm estimates the parameter values of HMM, improving significantly the recognition rates, both in training and test datasets over previously reported training procedures.

6. REFERENCES

[1] L.R. Bahl, P.F. Brown, P. Souza and R.L. Mercer. *Estimating Hidden Markov Model Parameters so as to Maximize Speech Recognition Accuracy.* IEEE Trans. on Audio Processing, Vol.1, No.1, pp.77-83. January, 1993.

[2] N.J. Nilsson. *Learning Machines.* New York. McGraw-Hill, chap. 4-5, pp.65-94, 1965.

[3] T. Kohonen, G. Barna and R. Chrisley. *Statistical Pattern Recognition With Neural Networks: Benchmarking Studies.* Proc. of the IEEE Conference on Neural Networks, pp.61-68. San Diego, 1988.

[4] T. Kohonen. *The Self-Organizing Map.* Proceedings of the IEEE, Vol.78, No.9, September, 1990.

[5] S. Katagiri and C.-H. Lee. *A New Hybrid Algorithm for Speech Recognition Based on HMM Segmentation and Learning Vector Quantization.* IEEE Trans. on Speech and Audio Processing, Vol. 1, No. 4, October, 1993.

[6] P.L. Galindo. *The Competitive Forward-Backward Algorithm.* Fourth International Conference on Artificial Neural Networks. Cambridge, UK. June, 1995.

[7] K.F. Lee. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers. Boston 1989.