Learning Vector Quantization, Multi Layer Perceptron and Dynamic Programming: Comparison and Cooperation

Xavier Driancourt, Léon Bottou, Patrick Gallinari

<xd@lri.lri.fr> or <xd@FRLRI61.BITNET>

L.RI. Bat 490
Université Paris Sud
94450 Orsay
FRANCE

Abstract

This paper presents the comparison of several methods (DP, MLP, TDNN, Shift-Tolerant LVQ, K-Means) on a multi-speaker isolated word small vocabulary problem. A sub-optimal cooperation between TDNN and other algorithms is proposed and successfully tested on the problem. The combination of TDNN and DP performs especially well. An optimal cooperation method between DP and some other algorithms is proposed.

1 Introduction

Speech can be considered as a sequence of local phonetic events. Consequently, speech recognition should integrate both sequence and phonetic events detection aspects. From this point of view, many already popularized speech recognition methods are not satisfying. Many methods use global recognition of whole patterns (e.g. classical MLP, LVQ, K-Means ...), some methods perform local events detection (e.g. TDNNs, Shift-Tolerant LVQ or K-Means ...). Dynamic Programming performs sequence detection by using some simple distance to match frames. Two popular methods integrate partially the two aspects: Hidden Markov Models which try to reduce the weakness of local detection performed by DP and Recurrent Networks which learn to perform some kind of sequence detection. Cooperation between HMM or DTW for time alignement and Neural Networks for local events detection seems especially interesting. Such cooperative models have been recently introduced [Bourlard 90, Sakoe et al. 88].

2 A Multi-Speaker Isolated Word Recognition Problem

The French database used for the experiment reported here is made of **thirty words** uttered **ten times** by each of **ten speakers**. It was sampled at 10 KHz. In all the experiments reported here, we have used a bark-scale **filterbank** signal preprocessing. The result is a condensed spectrogram, made of 100 frames of 16 coefficients per second; the overlap between two consecutive frames is 1/200 second.

The database was recorded in an **office environment**. The words selected for the database are the ten digits and twenty text editor command words.

Half of the database was used for training and the other half for testing. The test set had about 1500 words, so for example the 95% confidence interval at 96% correct classification is 1.1%.

0-7803-0164-1/91/0000-0815$01.00©1991 IEEE

3 Comparison of Several Methods

We give herein some details about the versions of the different methods we have employed, as well as the recognition rates they obtained on the test base.

Dynamic Programming (DP)

Our Dynamic Progamming algorithm is quite simple and uses as references the examples from the training base. This DP obtained 86.7% recognition.

K-Means

K-Means is a **stochastic** algorithm: we adapt references after presentation of each example and the database is presented several times. K-Means can be used as an efficient classification method. To do this, we **supervise** the algorithm by applying a K-Means independantly to the different classes. Furthermore, we use a **shift-tolerant** version, based on the same principle as the shift-tolerant LVQ algorithm [McDermott]: during training, K-Means is applied to temporal windows over the input samples. For classification, each word is scanned and the decision is obtained through the vote of all temporal windows. This sophisticated version obtained 82% success.

Learning Vector Quantization (LVQ)

We used a **shift-tolerant** LVQ [McDermott] (see above *K-Means* for more details on shift-tolerance). The algorithm we used is a modification of **LVQ2** [Kohonen] (we prefer to consider the **nearest reference of the good class** instead of the second closest reference). We initialized LVQ with K-Means. The best LVQ of this kind had 11 references for each class, and obtained **83% success; with** 16 references per class, the learning rate was better, but the test was not as good: 79%.

Adaline

Adaline gave a baseline performance of 74%.

Time Delay Neural Network (TDNN)

Our version of TDNN is stochastic. Figure 1 shows our most efficient architecture, called IWR2.

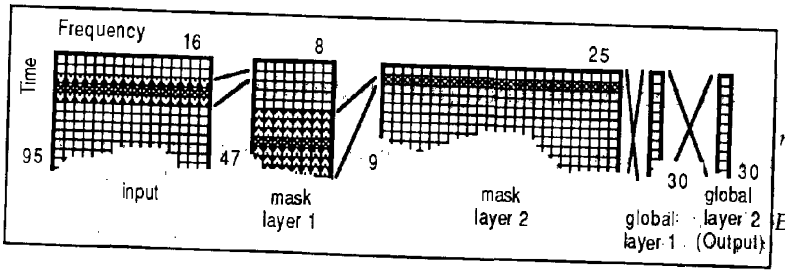

Figure 1 : IWR2

This figure shows the most efficient classical TDNN architecture. Each square stands for a neuron

Many architectures were tested. We mention here two modifications of IWR2: IWR1, where the layer *"global layer 1"* has been deleted and IWR1-16, where *global layer 1* has been deleted and where *mask layer 1* uses 16 hidden units per frame (instead of 8 in IWR2). IWR2 gave a 96,5% recognition rate, IWR1-16 96.5%, and IWR1 94.5%.

Frequentialy Connected TDNN (F-TDNN)

In common TDNN, masks (i.e. quasi-linear filters) are local along the temporal dimension, but global in the frequential dimension. It is possible to use masks which use local connections in both dimensions. Then "expert" knowledge can be introduced in the design of these connections, which can improve performance and speed. A first attempt with this method, gave 97% recognition. This F-TDNN is a restriction of the architecture of TDNN IWR1.

Recurrent Neural Networks (RNN)

We used a special kind of Recurrent Neural Network where the recurrent part of the RNN is implemented with shared weights, by unfolding the recurrent network along time. Consequently, the RNN has a finite temporal extension.

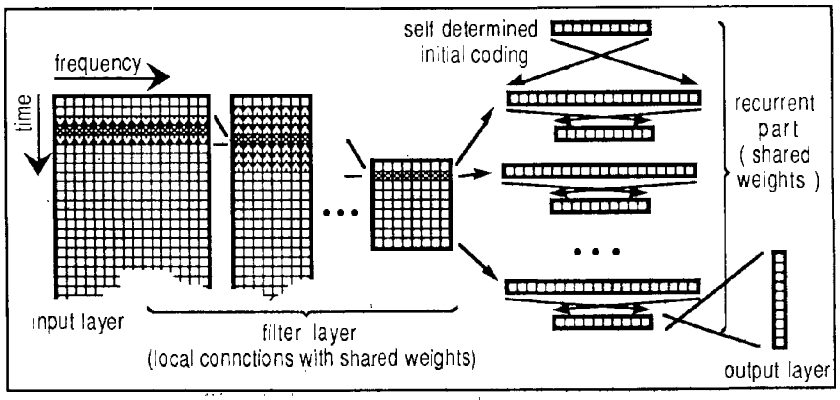

Figure 2: Recurrent Network

This recurrent network has a finite temporal extension; the iterative part is implemented with shared weights

We faced problems in conditionning the algorithm. Our best recurrent network obtained 96% success.

The following table sums up all the results mentionned in §3:

DP	Shift K-Means	Shift LVQ-16	Shift LVQ-32	Adaline	TDNN IWR1	TDNN IWR2	TDNN IWR1-16	F-IWR1	RNN
86.7	82	83	79	74	94.5	96.5	96.5	97	96

4 A Simple Sub Optimal Cooperation Method: Feature Extraction

The method we propose here consists in using the lower layers of a TDNN to provide a preprocessing, then any classifier can be used on this preprocessed data.

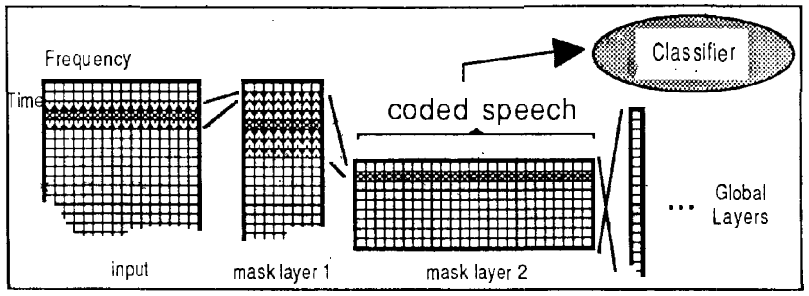

Figure 3: Feature Extraction

Propagation through the network is redirected after last mask layer towards another classifier.

Figure 3 illustrates feature extraction. In a first step, we train a TDNN on our IWR problem. Then we code the data with the TDNN mask layers, resulting in numerical *extracted features*. Then we use any other classifier on these *features* to solve our problem.
This cooperation is said *sub-optimal* because nothing guaranties that the TDNN trained in a first step on the IWR problem will provide a preprocessing which will code the data in a way which fits the classifier used afterwards.

We have tested various combinations of TDNN architectures and classifiers. The results are given below:

Top classifier / Feat. extractor	DP	MLP1	MLP2	LVQ-LBG	LVQ KM16	LVQ KM32	RNN
TDNN IWR1	98.8	94.5	96.5	97.75	97.12	97.25	96.5
TDNN IWR2	92	86	95	KM stands for K-Means; LBG is another			
TDNN IWR1 16	94			clustering algorithm used to initialize LVQ;			
F-TDNN IWR1	98.8			MLP1 and 2 are MLPs with 0 and 1 hidden layer.			

Further experiments should be made for a complete understanding of these first results. We can already make some remarks:

- the best scores are clearly obtained by combining DP and TDNN IWR1 or F-TDNN IWR1,
- the training of the lower layers of a TDNN is influenced by the upper layers; a two-layers top used during training influences the training of the lower layers, so that the extracted features are more complex (and harder to understand by top classifiers used after derivation) than extracted features obtained by training the same lower layers with a single-layer top,
- these cooperative systems are clearly sub-optimal: the code obtained by feature extraction may be not especially adapted to the top classifier. It is especially critical for DP, which is not adaptive and uses simple distances (Euclidean, Manhattan...) to match frames.
- a further advantage of feature extraction is the reduction of dimensionality which allows to spare a large amount of time for training adaptative top classifiers (LVQ, ...) or for using matching algorithms (DP); in the case of DP, there is a ratio of about 8 to 1.

5 Optimal Cooperation With Dynamic Programming

Encouraged by the good results given by the sub-optimal cooperation methods, we began working on a familly of optimal cooperation algorithms, based on the frame to frame matching operated by DP. For purpose of simplicity, we introduce and detail these algorithms for isolated word recognition only.

Dynamic Programming computes a numerical mismatch between two sequences of frames. By the way, a path is found in the grid of frame to frame matching. This path is a sequence of frame to frame correspondance. We can then build a *K-Means-DP*: given a training word, the closest DP reference W among the set of references Ω can be adapted by applying a gravity attraction (like k-means) from each frame of the training word on each corresponding frame of W.

Actually, this system is an original application of gradient back-propagation. We call Q the cost function (error) back propagated applied on a training example $X = x_1,\ldots, x_{P_X}$; $W = w_1,\ldots, w_{P_W}$ a reference and $c_1 \ldots c_M$ the frames of all reference words together: $c_{W1} = w_1, \ldots, c_{WP} = w_P$ are the frames of word W. A path which matches X to W is written: $p_{XW} = (l_{XW}, \sigma_{XW}, \hat{\sigma}_{WX})$, where l_{XW} is the length of the path and $\sigma_{XW}(i)$ (resp. $\hat{\sigma}_{WX}(i)$) the coordinates of X (resp. W) for the i^{th} step of the path. We can write Q:

$$Q(X,c_1,c_2,\ldots,c_M) = \underset{\substack{\text{words} W \\ \text{in set } \Omega}}{\text{Min}} \left(\underset{\substack{\text{paths } p_{XW}\,(l_{XW}\,,\,\sigma_{XW}\,,\,\hat{\sigma}_{WX}) \\ \text{which match } X \text{ to } W}}{\text{Min}} \sum_{i=1}^{l_{XW}} d(x_{\sigma_{XW}(i)}\,,\,w_{\hat{\sigma}_{WX}(i)}) \right) \quad [1]$$

Given the closest reference **W** and the best path p_{XW} which matches **W** to **X**, we can write the gradient of Q, using twice the property: $[\nabla_y \text{Min}\, Z = \nabla_y Z_{min}]$. We obtain:

$$\nabla_{cj} Q = \sum_{i \text{ such that: } \hat{\sigma}_{WX}(i) = w_j} \nabla_w \hat{\sigma}_{XW(i)}\, d(x_{\sigma_{XW}(i)}\,,\,w_{\hat{\sigma}_{WX}(i)}) \;;\; \text{reminding that for any j: } w_{w_j} = c_j \quad [2]$$

We can remark, that consequently:

$$\nabla_{cj} Q = 0 \quad \text{for} \quad j \notin \{w1,\ldots,wp\} \quad [3]$$

The same method works to build a LVQ2-DP; compared to K-Means-DP, we must make two modifications. First, nothing happens when the classification is good. Second, when a classification is bad, the frames of closest reference word **W** are repulsed, while the frames of closest good reference **V** are attracted by the corresponding frames of training exemple **X**.

The equation [2] gives the possibility of initializing the gradient of the top of TDNN lower layers. The resulting adaptative systems are then TDNN-K-Means-DP and TDNN-LVQ2-DP.

6 Conclusion

We have compared many popular methods of speech recognition on a real-size simple problem. We have tested a sub-optimal cooperation method between TDNN and other classifiers, and found that cooperation between Dynamic Programming and Time Delay Neural Network was especially efficient. We have introduced a familly of optimal algorithms, based on error back-propagation and which makes DP cooperate with K-Means, LVQ and TDNN.

Acknowledgement

Part of this work was developped for ESPRIT project nb 2059, named Pygmalion. Two authors were supported by DRET grants (YLB 87/808/19 and XD 89/1591).

References

Bourlard H. 1990: *How Connectionist Models could Improve Markov Models for Speech Recognition* - Proceedings of the International Symposium on Neural Networks for Sensory and Motor systems, March 1990, Dusseldorf, F.R.G., Ed R. Eckmiller

Mc Dermott E., Katagiri S. 1989: *Shift-Invariant, Multi-Category Phonem Rcognition using Kohonen's LVQ2* - Proceedings of ICASSP-89, Glasgow, U.K., 1989, pp 81-84

Sakoe H., Isotani R., Yoshida K., Iso K., Watanabe T. 1988: *Speaker Independant Word Recognition Using Dynamic Programming Neural Networks* : Proceedings of ICASSP-88, pp 107-110, New-York, 1988