IDIAP RESEARCH REPORT

IMPROVED PHONE POSTERIOR ESTIMATION THROUGH K-NN AND MLP-BASED SIMILARITY

Benjamin Picart

Idiap-RR-18-2009

Version of AUGUST 19, 2009

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

POLYTECH.MONS

idiap RESEARCH INSTITUTE

Faculté Polytechnique de Mons
&
Idiap Research Institute

Telecoms and Multimedia Master's Thesis report presented by

Benjamin PICART

IMPROVED PHONE POSTERIOR ESTIMATION THROUGH k-NN AND MLP-BASED SIMILARITY

Academic year 2008 - 2009

I would like to thank Professor Hervé Bourlard, my master's thesis supervisor, who allowed me to do it at the Idiap Research Institute and who gave me a very interesting work subject: Improved Phone Posterior Estimation Through k-NN And MLP-based Similarity. I thank him, and also Dr Mathew Magimai Doss and Mrs Afsaneh Asaei for their kindness, their guidance, their availability and their support all along this thesis.

I am also thankful to Professor Thierry Dutoit, my supervisor at Faculté Polytechnique de Mons in Belgium, for giving me the opportunity to realize my master's thesis within the international scientific and cultural context of the Idiap Research Institute.

I would like to thank also particularly Mrs Nadine Rousseau and Mrs Sylvie Millius for their kindness, their advices and their helps in all the administrative procedures and in the research of an apartment in Martigny.

Finally, I would like to thank all the people working in Idiap Research Institute for their warm welcome, when I came there for the first time and throughout my master's thesis, and particularly: Afsaneh, Petr, Giulia, Alfred, Hari, Joel, Fabio, Mathew, John, Phil, Danil, Niklas, Chris, Deepu, Joan, Paco, Francesco, Roger, Cosmin, Hui, Lakshmi, Daira, Oya, Stefan, Perttu, Jordi, Hugo, Majid, Radu, Valérie, Stéphanie, Elisa, Tatiana, Kate, Hayley, Silèye, Xiangxin, Gokul, Nikhil, David, ….

This work was performed in the context of a European ERASMUS exchange program, complemented by the AMIDA training program. AMIDA (Augmented Multi-party Interaction with Distance Access) is a European Union Integrated Project, contract number IST-033812. The authors gratefully thank the EU for their financial support, and all project partners for a fruitful collaboration. More information about AMIDA is available from the project web site www.amiproject.org.

Abstract

In this work, we investigate the possible use of k-nearest neighbour (kNN) classifiers to perform frame-based acoustic phonetic classification, hence replacing Gaussian Mixture Models (GMM) or MultiLayer Perceptrons (MLP) used in standard Hidden Markov Models (HMMs). The driving motivation behind this idea is the fact that kNN is known to be an "optimal" classifier if a very large amount of training data is available (replacing the training of functional parameters by plain memorization of the training examples) and the correct distance metric is found.

Nowadays, amount of training data is no longer an issue. In the current work, we thus specifically focused on the "correct" distance metric, mainly using an MLP to compute the probability that two input feature vectors are part of the same phonetic class or not. This MLP output can thus be used as a distance metric for kNN. While providing a "universal" distance metric, this work also enabled us to consider the speech recognition problem under a different angle, simply formulated in terms of hypothesis tests: "Given two feature vectors, what is the probability that these belong to the same (phonetic) class or not?". Actually, one of the main goals of the present thesis finally boils down to one interesting question: "***Is it easier to classify feature vectors into C phonetic classes or to tell whether or not two feature vectors belong to the same class?***".

This work was done with standard acoustic features as inputs (PLP) and with posterior features (resulting of another pre-training MLP). Both feature sets indeed exhibit different properties and metric spaces. For example, while the use of posteriors as input is motivated by the fact that they are speaker and environment independent (so they capture much of the phonetic information contained in the signal), they are also no longer Gaussian distributed.

When showing mathematically that using the MLP as a similarity measure makes sense, we discovered that this measure was equivalent to a very simple metric that can be analytically computed without needing the use of an MLP. This new type of measure is in fact the scalar product between two posterior feature vectors.

Experiments have been conducted on hypothesis tests and on kNN classification. Results of the hypothesis tests show that posterior feature vectors achieve better performance than acoustic feature vectors. Moreover, the use of the scalar product leads to better performance than the use of all other metrics (including the MLP-based distance metric), whatever the input features.

Keywords: Automatic Speech Recognition (ASR), posterior-based speech features, short-term spectrum-based speech features, Multi-Layer Perceptron (MLP), hypothesis tests, k-Nearest-Neighbors classification rule (kNN), Euclidian distance, Mahalanobis distance, Bhattacharyya distance, Kullback-Leibler divergence, Scalar Product similarity

Contents

List of Figures

List of Tables

Notations and conventions

- $x_n = (x_{n1}, x_{n2}, \ldots, x_{nD})^T$: acoustic vector at time n
- D : dimension of acoustic vectors
- T : transpose operation
- ω_k : a general class
- C : number of classes
- $X = \{x_1, \ldots, x_n, \ldots, x_N\}$: acoustic vector sequence of length N; each $x_n \in \mathrm{X}$
- $X_{n-c}^{n+c} = \{x_{n\text{-}c}, \ldots, x_n, \ldots, x_{n+c}\}$: a subsequence of X of length $2c + 1$
- c is the context window
- $\mathrm{X} = \{X_1, \ldots, X_M\}$: set of acoustic vectors
- Acoustic (feature) vector: vector whose components are the short-term spectrum coefficients like MFCC, PLP, …
- Posterior (feature) vector: vector whose components are the phonetic classes a posteriori probabilities, given an acoustic feature vector
- Feature vector: will be used when applicable to both Acoustic feature vector and Posterior feature vector

1. Introduction

Typical Automatic Speech Recognition (ASR) systems use features obtained from short-term spectrum, like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction (PLP). Phoneme *a posteriori probabilities* (more commonly called *posterior*) can also be used as features, knowing that they are more stable and more robust (this will be discussed in Section 2.2). Posterior probabilities are currently often used as powerful features to improve automatic speech recognition (ASR) systems. The interesting ideas behind posterior probabilities are that they could be provided by discriminant training while accommodating acoustic context. This idea was first used in the development of the successful hybrid HMM/ANN system which initiated extensive use of posteriors in speech recognition systems. In this approach, emission probabilities required in HMM system are provided by a posteriori probabilities computed by an Artificial Neural Network (ANN), and more specifically by MLP [7]. Hence, in HMM/ANN the posterior probabilities are used as local classifiers. This application of posteriors as local measures was later explored in several other speech recognition purposes such as word lattice rescoring [32], beam search pruning [1] and confidence measures estimation [4]. On the other hand, posterior probabilities could be used as acoustic features. This approach was proposed and implemented in the state-of-the-art Tandem speech recognition system where posterior probabilities are used as the most discriminant and informative features.

There are several types of non-parametric methods of interest in pattern recognition [21]. Some of them estimate the density functions $p(x_n|\omega_j)$ - the class-conditional probability density function (probability density function for x_n given that the state of nature is ω_j) - from sample patterns. Some other alternatives directly estimate the a posteriori probabilities $P(\omega_j|x_n)$ - the probability of the state of nature being ω_j given that feature value x_n has been measured. This is closely related to non-parametric design procedures, such as the nearest-neighbor rule, which bypasses explicit probability estimation and goes directly to decision functions.

The (k)-Nearest Neighbor (kNN) rule is amongst the most popular and successful pattern classification techniques. Despite the simplicity of the algorithm, it performs very well and is an important benchmark method. The kNN classifier, as described by [14], requires a distance metric d, a positive integer k, and the reference templates X of M labelled patterns.
Generally, Euclidian or Mahalanobis distances have been used as local distance between feature vectors. However, the notion of a metric is far more general, and we now turn to the use of alternative measures of distances to address key problems in classification.

In the current work, we thus specifically focused on the "correct" distance metric, mainly using an MLP to compute the probability that two input feature vectors are part of the same phonetic class or not. This MLP output can thus be used as a distance metric for kNN. While providing a "universal" distance metric, this work also enabled us to consider the speech recognition problem under a different angle, simply formulated in terms of hypothesis tests: "Given two feature vectors, what is the probability that these belong to the same (phonetic) class or not?".

We will assess the performance of the new MLP-based distance metric against the more conventional Euclidian, Mahalanobis, Kullback-Leibler and Bhattacharyya metrics.

When showing mathematically that the MLP as similarity measure is working well, we discovered that this measure was equivalent to a very simple metric that can be analytically computed without needing the use of an MLP. This new type of measure is in fact the scalar product between two posterior feature vectors.
The analysis will be done with standard acoustic features as inputs (PLP) and with posterior features (resulting of another pre-training MLP).

This work must be considered as an experiment to evaluate the potential usefulness of the MLP and the scalar product as a (non-linear) similarity measure between feature vectors, to improve phone posterior estimation through k-NN.

This master's thesis is organized as follows. We begin by a theoretical part (Section 2), where we explain the basic concepts governing this work, in particular: Automatic Speech Recognition (Section 2.1), the state of the art, the motivations to use posterior feature vectors, a reminder of what is the kNN classification rule and the definition the distances used in this work (Section 2.2) and finally a reminder on the Multi-Layer Perceptron and the estimation of its parameters (Section 2.3).
Section 3 presents in more details the concepts of acoustic features extraction (Section 3.1) and of a posteriori probabilities estimation (Section 3.2).
Section 4 shows mathematically how an MLP can be used as similarity estimator (Section 4.1). After that we show that the output of the MLP is an estimation of the scalar product of the 2 (actual) posterior feature vectors associated to the 2 input feature vectors (Section 4.2).
Section 5 explains the principles of pairs of feature vectors creation, which will be useful throughout this work.
Section 6 will explains the main ideas of the hypothesis test, based on a histogram drawing approach. In this section, the objective of hypothesis tests is explained (Section 6.1). Then we show how to choose the feature vectors necessary for that experiment (Section 6.2). The optimal decision point determination is explained in Section 6.3, which conditioned the training and cross-validation accuracy. And finally, the experimental setup is briefly described (Section 6.4).
Section 7 shows the general bloc diagram of our classification system based using kNN rule, and explains its different parts.
Experiment results are explained in Section 8. The database used is described in Section 8.1, the parameters selected for features extraction in Section 8.2, the results of the posteriors estimates in Section 8.3, how the pairs of feature vectors were created in Section 8.4, results of the hypothesis tests in Section 8.5 and of kNN classification rule in Section 8.6.
Finally, Section 9 gives some ideas of future works, to go further with the results obtained in this thesis, and Section 10 concludes this work.

2. Background

2.1. What is Automatic Speech Recognition (ASR)?

The goal of Automatic Speech Recognition (ASR) is to recognize automatically (i.e. by the machine) the message expressed by a spoken utterance independently of the speaker and the environment. An ASR system follows the structure of a pattern classification task [14]. As shown in Figure 1, the speech signal is first processed to extract the features that are necessary to recognize the linguistic message. Then a distance score is computed between the speech features and each reference class (Acoustic Modeling) and a classification decision is finally made according to the distance scores (Decoder).

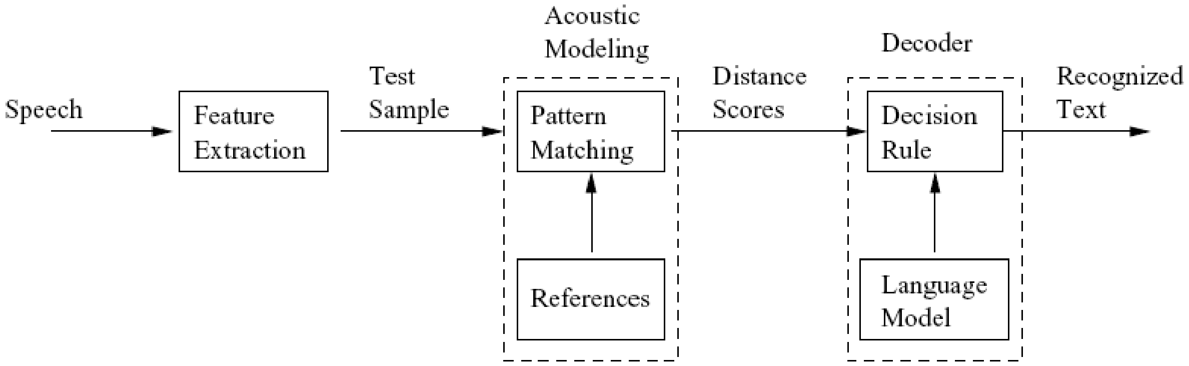

Figure 1: General block diagram of an ASR system [2]

This thesis focuses on the acoustic model and the distance scores definition. As already explained briefly in the introduction, we will use an MLP as similarity measurer between feature vectors.

The input of an ASR system is a digital speech signal sampled typically at 8 or 16 kHz. It contains the various features of the source, which are often redundant or irrelevant for the recognition task. Only the lexical information is useful in this case. The role of the feature extraction is thus to remove this useless information, which is not related to the linguistic message W represented by the signal. A vector of acoustic features is computed on a fixed-length window, shifted typically by 10 ms. Therefore, a sequence of acoustic feature vectors is obtained at the end of the feature extraction module.

Within the Bayesian framework, the ASR problem can be formulated as follow. Considering a sequence of speech features $X = \{x_1, x_2, \ldots, x_n, \ldots, x_N\}$, where $x_n = (x_{n1}, x_{n2}, \ldots, x_{nD})^T$, the most probable linguistic message (sequence of words, or sequence of phonemes in this work) $\hat{W}$ is then [34]:

$$
\begin{aligned}
\hat{W} &= \arg\max_{W \in p} p(W|X) \\
&= \arg\max_{W \in p} \frac{p(X|W)\,p(W)}{p(X)} \\
&= \arg\max_{W \in p} \left[\log p(X|W) + \log p(W)\right]
\end{aligned}
$$

where:

- π represents the set of all possible word sequences
- $p(X|W)$ is called the acoustic model (it depends on the sequence of speech features X)
- $p(W)$ is called the language model (it represents the prior knowledge about the sequence of words W)
- $p(X)$ can be ignored because it does not affect the maximization solution

2.2. State of the art and general ideas

The entry of the ASR system is the speech signal recorded by a microphone. This speech signal contains the various features of the source (lexical information, speaker, noise, signal reflection, …). The ultimate goal of the acoustic front end is to transform the input signal into robust feature vectors so as to make it (the most possible) independent of the features of the source, except of course of the lexical information. Typically, acoustic features obtained from short-term spectrum, like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction (PLP) are used.

However, another kind of feature can be used, which are more stable and more robust: the a posteriori probabilities (commonly called posteriors). In this work, phones posterior probabilities will be used.

2.2.1. Posterior-based speech features

Posterior features were initially motivated as a simple scheme to take the advantage of both HMM/ANN and HMM/GMM speech recognition frameworks [25]. These features are extracted by an MLP using spectral-based features such as MFCC or PLP as input. In this approach, each output unit of the MLP is associated with a particular class (phoneme in this work) of the set of all possible classes and it is trained to generate a posteriori probabilities of the classes conditioned on the input acoustic observation sequence X, i.e. $p(\omega_i|X)$. While allowing for discriminant training, such an approach also has the advantage of possibly accommodating acoustic context by providing several frames at the MLP input, thus estimating $p(\omega_i| X_{n-c}^{n+c})$, where $X_{n-c}^{n+c} = \{x_{n-c}, \ldots, x_n, \ldots, x_{n+c}\}$ (the context window c is typically equal to 4) [10]. However, context up to $c = 50$ has also been successfully used [26].

These MLP-generated phoneme posterior probabilities could be fed (after some transformation) as input posterior feature vector into the standard HMM recognizer. Tandem has been the most successful system which made this scheme possible [27]. In this approach, the MLP posterior probability estimates are roughly gaussianized by computing logarithm of the MLP output (a static nonlinearity) and whitened by the Karhunen-Loeve transform (KLT) derived from the training data [10]. Such gaussianized and whitened posterior probabilities form the feature vector for the subsequent HMM/GMM recognizer. Thus, the conventional features derived from a spectral density vector representing the spectral envelope are replaced by the transformed posteriors of acoustic events (in the original concept the events were context-independent phonemes) [10].

Input to Tandem [10] can be any data that are believed to provide a relevant evidence for the classification. In its simplest form, Tandem takes as an input a superframe of typical conventional speech features resulting of the concatenation of 9 frames composed of PLP static and dynamic features. Usually, Tandem inputs are concatenated outputs from other sub-band classifiers (TRAP [26] or HATS [11]). TRAP has been also reported to be efficient in combining different features and for alleviating irrelevant information [30] [38].

In both main applications of posterior probabilities, either as local classifiers or as features, the system efficiency strongly depends on the quality of the estimated posteriors, and the compatibility of the models and similarity measures used. To boost the quality of the posteriors, another classifier is often used, as a hierarchy, after the initial MLP in order to capture more phonetic and contextual information of the speech signal; whereas for model compatibility, posteriors are gaussianized and decorrelated to form the Tandem features and fed into the standard HMM/GMM or in KL-HMM, their distribution is directly used in HMM model where Kullback-Leibler divergence is used as similarity measure for better realization of posterior characteristics [3].

2.2.2. Motivations for using posterior-based feature vectors

The use of posterior-based speech features instead of spectral-based speech features (e.g. PLP coefficients) is motivated by the following advantages [2]:

- Trained features:
    - They are generated by an MLP; this contrasts with the extraction process of standard spectral-based features, which is based on a transformation mainly inspired from perceptual models;
    - A context window of generally 9 frames can be used in the MLP;
    - They are speaker and environment independent (if the MLP is trained on a rich enough database, in terms of speakers and vocabulary), so they provide a robust representation of the speech signal;
    - They are “detectors” that minimize the error probability in a Bayesian classifier (as explained in paragraph 3.2). So they can be seen as the optimal (phonetic) representation;
- Discriminant features:
    - The MLP is trained using a discriminative criterion;
    - Because of the non-linear discriminant analysis of the MLP in the input feature space, a transformation that projects the input features onto a sub-space of maximum class discriminatory information is learned [29]. This projection is able to suppress the noise related variability, while keeping the speech discriminatory information intact. Therefore, posterior features capture much of the phonetic information contained in the signal;
- Relax some of the independence / correlation assumptions, like the stationarity assumption to extract short term spectral-based features and so makes posteriors highly rich in contextual and phonetic information since this information is usually spanned in a long temporal interval [37];
- Each component of the posterior feature vector corresponds to a specific phoneme and contains a linguistically meaningful value;
- Since posterior features can be seen as discrete distributions, measures from the information theory field can be applied (e.g. Kullback Leibler (KL) divergence).

These appealing characteristics make posterior probabilities powerful features for ASR systems. However, the distribution of posteriors over the feature space is not easy to model using for example GMMs due to the sharp shape of the distribution (it is not Gaussian at all). In this work, we will use kNN classifier to do phone classification using posterior features. Since kNN is a non-parametric classifier, there is no need to assume any knowledge about the underlying statistical distribution. Moreover, given enough training data and a proper metric, the a posteriori distribution given the nearest-neighbor to the acoustic vector *x* also converges to the a posteriori distribution given *x* [13]. This makes kNN classifier a good candidate to deal with posterior features.

2.2.3. kNN classification rule

While being very simple, the (k)-Nearest Neighbor (kNN) rule is amongst the most popular and successful pattern classification techniques. Despite the simplicity of the algorithm, it performs very well and is an important benchmark method. The kNN classifier, as described by [14], requires a distance metric *d*, a positive integer *k*, and the reference templates X of *M* labelled patterns. The algorithm is summarized as follows:

- Out of the *M* training vectors, identify the *k* nearest neighbors of the test vector, irrespective of the class label
- Out of these *k* samples, identify the number k_i of vectors belonging to the class *i*, *i* = 1, …, *C*. Obviously, we have $\sum_{i=1}^{C} k_i = k$
- Assign to the test vector the class containing the maximum number k_i of samples

This method was first introduced by Fix and Hodges [17] [18] and later studied by Cover and Hart [12]. Cover and Hart have statistically justified that kNN approaches the optimal Bayes classifier as the number *M* of samples and *k* both tend to infinity in such a way that k/M à 0, which also states that the density estimates will converge to the true densities. The error in that case is the Bayes error, the smallest achievable error given the underlying distribution. Beyond this remarkable property, the kNN owes much of its popularity in the Pattern Recognition community due to its good performance in practical applications where it can be very competitive with the state-of-the-art classification methods [16] [24].

kNN is attractive in several ways:

- No need for a priori knowledge about the probability distribution of the classification problem
- No need of training, which is necessary for other methods like MLP for estimating the posteriors
- Can optimally estimate a posteriori probabilities when a large number of correctly labeled patterns is available

Furthermore, nonlinear transformation performed by MLP which converts PLP to posterior features is a kind of discriminant projection which makes posteriors more stable [25] and more robust to noise [29]. This transformation could also increase the efficiency of kNN for classifying phonemes. Thus, it is important to evaluate the possibility of using kNN with posterior features to perform local phonetic classification. In this case, we have to address the kNN main issues in posterior space.

Since kNN is a non-parametric classifier, posteriors could be used directly without any a priori assumption about their distribution (which is not Gaussian, as already said). On the other hand, according to the nearest neighbor rules, the samples which fall close together in feature space are likely either to belong to the same class or to have the same a posteriori distributions of their respective classes [13]. The few theoretical restrictions that we have to impose are merely intended to guarantee the convergence of the nearest neighbor to the true density as the number of training samples becomes arbitrarily large. This convergence for the finite-sample considerations in a d-dimensional Euclidean space is guaranteed under assumptions regarding the distance metric. Here after, several distance metrics are defined. The number *k* should also be small in order that all the kNN to the test sample will be contained in a small neighborhood. Furthermore, it is shown that the optimal value of *k* is case specific and depends on the observation to be classified (when using Geometric Nearest-Neighbor GNN classifier for example, as explained in the following paragraph) [22]. We have addressed these issues by proposing a new approach, the MLP-based similarity, for investigation of the posterior feature space.

2.2.4. Distance metrics definitions

From our discussion above, using a metric which respects the inherent characteristics and boundaries of the features space is a key to the kNN performance. Thus, we have explored different distance functions that are already used in posterior feature space.

The Euclidean distance function between feature vectors x_i and x_j, $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$ and $x_j = (x_{j1}, x_{j2}, \ldots, x_{jD})^T$, is probably the most commonly used in any distance-based algorithm. It is defined as:

$$d_e(x_i, x_j) = \sqrt{\sum_{k=1}^{D} (x_{ik} - x_{jk})^2}$$

The Mahalanobis distance takes into account the covariance among the variables in calculating distances [39]. With this measure, the problems of scale and correlation inherent in the Euclidean distance are no longer an issue. To understand how this works, consider that, when using Euclidean distance, the set of points equidistant from a given location is an hyper-sphere. The Mahalanobis distance stretches this hyper-sphere to correct for the respective scales of the different variables, and to account for correlation among variables. It is defined as:

$$d_m(x_i, x_j) = \sqrt{(x_i - x_j)^T S (x_i - x_j)}$$

where *S* is the covariance matrix of the data.

Previous studies have shown that Kullback-Leibler (KL) divergence is an appropriate measure of similarity in posterior feature space considering the boundaries and inherent characteristics of the posterior probabilities [2]. We have used a symmetric version of KL divergence which satisfies the triangular inequality and is defined as:

$$d_{KL}(x_i, x_j) = 0.5 \sum_{k=1}^{D} x_{ik} \log \frac{x_{ik}}{x_{jk}} + 0.5 \sum_{k=1}^{D} x_{jk} \log \frac{x_{jk}}{x_{ik}}$$

Bhattacharyya distance has been also used as a measure of similarity of two discrete probability distributions [19]. This distance function is defined as:

$$d_b(x_i, x_j) = -\log \sum_{k=1}^{D} \sqrt{x_{ik}\, x_{jk}}$$

In this thesis, we investigate a new type of metric exploiting an MLP to estimate the “distance” between two feature vectors x_i and x_j presented at its inputs. Actually, given (x_i, x_j) at the inputs of the MLP, the output is trained to estimate the probability that these two feature vectors belong to the same class or not. We will assess the potential further improvements of the kNN classifier performance using this metric.

2.3. The Multi-Layer Perceptron (MLP)

2.3.1. General architecture of the MLP

The most common method to estimate posterior probabilities of sub-word units given the cepstral-based features, such as phonemes is through an MLP [2] because it scales well with large amount of training data and it can easily incorporate contextual information. The general architecture of the one hidden layered MLP that will be used is illustrated in Figure 2. It consists of an input layer, composed of *D* input nodes (if we do not use contextual information, *D* represents the dimension of the acoustic vectors; if we use contextual information, *D* represents the dimension of the acoustic vectors times the length of the context window), an hidden layer containing *H* hidden units and an output layer containing *C* nodes (where *C* is the number of classes). Each input is connected to each hidden unit and each hidden unit is connected to each output. The values on each node of the hidden / output layer are computed through non-linear function (sigmoid / softmax respectively, in our case) of the input / hidden values. The non-linearity on the hidden layer is used to generate higher order momentum of the input vectors [23]. The non-linearity in the output layer is mandatory because it allows simulating a binary decision, which minimizes the classification error rate [23]. If we want to model a posteriori probabilities, it should be nice if all the output values sum to one, to respect the probability definition. This is the reason why we use the softmax function on the output layer, which is defined as [7]:

$$j\,(u_i) = \frac{e^{u_i}}{\sum_{k=1}^{H} e^{u_k}}$$

where:

- φ is the activation function of a neuron
- u_i is the value of the output node just before the non-linear function
- H is the number of hidden units

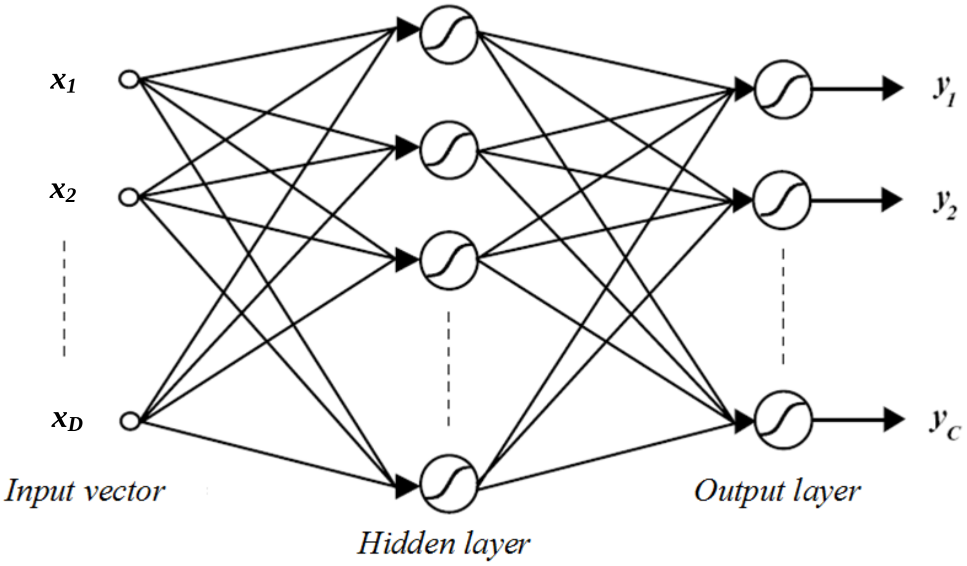

Figure 2 : General architecture of a one hidden layered MLP [23]

2.3.2. Estimation of the MLP parameters

The parameters of the MLP, i.e. the inter-layer weights, can be estimated using a supervised training (as explained in Section 3.2), based on pre-classified data (the training data). This MLP is trained[1] using the standard (feed-forward) back propagation algorithm with the cross entropy error criterion (also know as the relative entropy criterion) as cost function. This cost function is minimized using a gradient descent algorithm.

In practice, two main cost functions can be used: the least mean squared error criterion or the cross entropy error criterion.

The least mean squared error criterion is defined as follow:

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\left(d_k(n) - g_k(n)\right)^2$$

where:

- N is the number learning vectors
- C is the number of classes
- $d_k(n)$ is the desired k-th output for input x_n
- $g_k(n)$ is the observed k-th output for input x_n
- $d(n) = (d_1(n), \ldots, d_C(n))^T$ is the desired output vector
- $g(n) = (g_1(n), \ldots, g_C(n))^T$ is the observed output vector

[1] In this work, all the neural networks are trained using the ICSI tool QUICKNET V3.20 [28]

The cross entropy error criterion is defined as follow:

$$E_e = \sum_{n=1}^{N} \sum_{k=1}^{C} \left[g_k(n) \ln\left(\frac{g_k(n)}{d_k(n)} \right) + (1 - g_k(n)) \ln\left(\frac{1 - g_k(n)}{1 - d_k(n)} \right) \right]$$

The use of the entropy criterion, instead of the least mean squared error criterion, allows the learning not to slow down because of a saturation of the output units. It is shown in [7] that the use of the entropy criterion allows correcting the weights of the output layer with an amplitude proportional to the gap between the real and the desired output values (i.e. if the real output value is very far from the desired output value, then the weights correction will be very big, and conversely).

The learning rate and stopping criterion are controlled by the frame classification rate on the cross validation data. Indeed, during the training, measuring on one side the recognition rate evolution for the objects that participate in the training of the model (training objects), and on the other side the recognition rate evolution for the objects that do not participate in the training (validation objects), we obtain two curves similar to those presented in Figure 3.

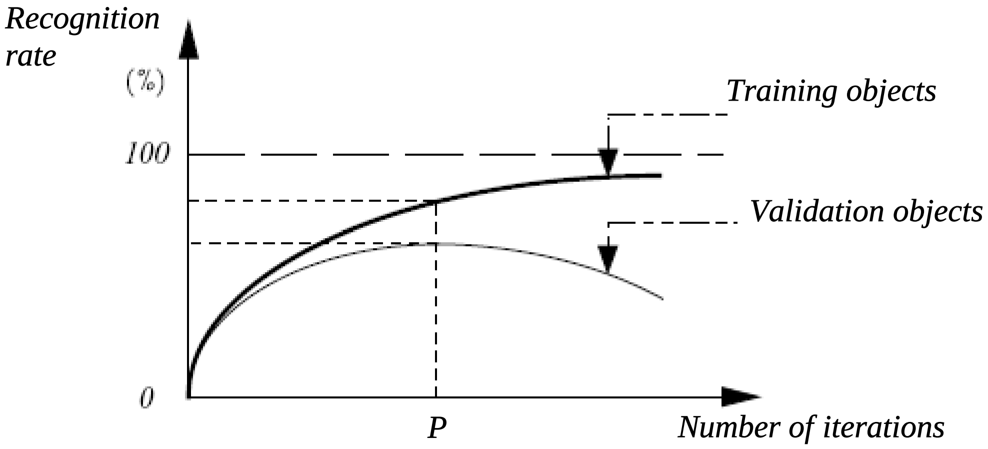

Figure 3 : Cross-validation principle (adapted from [23])

While the recognition rate always increases on the training objects, it decreases (from point *P*) on the validation objects when the training lasts too long. This is explained by the fact that the model try to memorize the objects themselves instead of the discriminant features [23]. Therefore, we have to stop the training at point *P*.

The name of the strategy used for updating the learning rate in successive training epochs is “newbob”. It means using a constant learning rate until the error reduction drops below a given threshold (that can be set manually), and then decreasing it exponentially.

There are two main methods for training an MLP: online and offline. In the first case, the weights are updated after each training object presentation (the instantaneous gradient of the cost function is used). The latter case consists in the accumulation of the instantaneous gradients of the cost function and in adapting the weights when all the training object have been presented to the MLP. Actually, in our work, the training is quasi-online, i.e. that the weights are updated after each presentation of a bunch of 256 training objects.

3. Acoustic front-end

3.1. Acoustic features extraction

The goal of the acoustic front end is to transform the input signal into robust feature vectors so as to make it (the most possible) independent of the features of the source, except of the lexical information.

Generally, automatic speech recognition systems proceed in two main steps [9]: the first one is a pre-processing step (microphone, pre-amplifier, anti-aliasing filtering, A/D convertor, telephonic line …), and the second one is the feature extraction. As the speech is highly non-stationary, its analysis must be performed inside successive elementary frames which we suppose to be stationary. Typically, an analysis is performed every 10 ms on 30 ms-long frames (by shifting and overlapping of the analysis frames, in order to improve the smoothness properties of the analysis frames), on which a Hamming window is applied, to generate an acoustic vector representative of the frame being analyzed. This is illustrated in Figure 4 (in our case, the analysis frame length is 25 ms). This acoustic vector contains 13 Perceptual Linear Prediction (PLP) coefficients (12 cepstral coefficients + the energy value, i.e. cepstral C_0 coefficient), which are continuous values. These coefficients code progressively the perceptual spectral envelop of the signal (perceptual power spectral density). A detailed description of how these coefficients are extracted can be found in [36].

At the end of this process, a sentence or a word is represented by a sequence of acoustic vectors $X = \{x_1, \ldots, x_N\}$, where x_i is the acoustic vector computed at time i*10 ms over a 30 (or 25) ms-long frame. In this work, an acoustic vector will not represent a sentence or a word but a phoneme. Ideally, this module should minimize the effects of the non-linguistic sources.

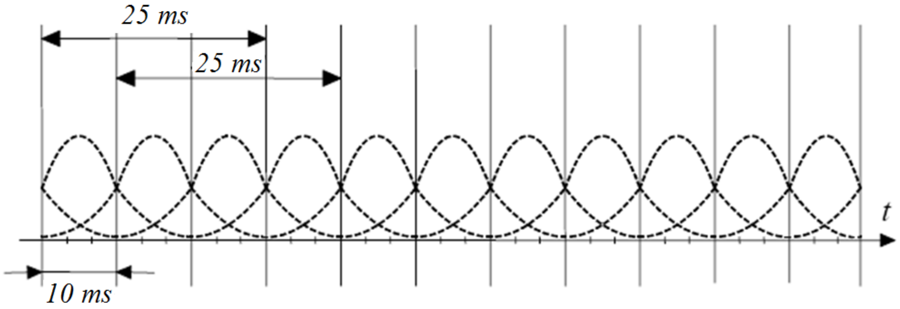

Figure 4 : Speech signal analysis by shifting 10 ms by 10 ms a Hamming window of length 25 ms [15]

Moreover, a mean and variance speaker normalization is applied to these cepstral coefficients. We proceeded as follow:

- Compute the mean and the variance over the cepstral features vectors belonging to each speaker;
- For each speaker, subtraction of the mean from his cepstral features vectors and division of the results by the variance.

More recently, it has been shown [9] that the performance of state-of-the-art automatic speech recognition systems were significantly improved when using the dynamic properties of the sequence of short-term spectrum acoustic vectors discussed above. Indeed, there is a strong correlation between adjacent frames due to the continuous nature of the speech signal and the overlapping shifting between time windows. This can be achieved by extending the acoustic vectors with their first and second order temporal derivatives [20]. These derivatives are estimated as the slope of a linear regression among a context of typically 5 (the current input + the 2 previous acoustic vectors + the 2 following acoustic vectors) frames. These parameters are usually called delta and double delta respectively. Hence, typical acoustic vectors contain 39 dimensions (13 static features + 13 delta features + 13 double delta features), as shown in Figure 8.

Although the speech processing described above reduces the information related to the speaker and environment, spectral-based speech features still suffer from a high variance in the feature space of a sound [2]. This is the reason why we will use another type of speech features, the posterior-based speech features.

3.2. A posteriori probability estimation

In our case, these features have been extracted by an MLP using the PLP features as input. The use of an MLP for estimating the a posteriori probabilities is motivated as follow [23]. It has been show [8] that if a one hidden layered MLP, containing neurons with non-linear continuous activation function, is trained under the following conditions:

- The neural network contains one neural output per class, and is trained to produce a "1" on the output associated with the class of the input vector, and "0" on all the other outputs
- The optimization criterion is the least mean squared error (LMSE) or the entropy criterion
- The number of hidden units is large enough
- The training does not converge to local minimum

then, the outputs of the MLP can be seen as good approximations of the classes a posteriori probabilities.

Therefore, in that case, the MLP allows us to approximate the optimal discriminatory Bayes decision rule, defined as:

$$x_n \in w_i \Leftrightarrow p(w_i|x_n) > p(w_j|x_n) \quad \forall j = 1,2,...,C, \; j \neq i$$

where:

- x_n is the input acoustic vector (39 dimensions)
- ω_i, (i = 1, …, C) are the output classes
- C is the total number of classes (40 in our case)
- $p(\omega_i|x_n)$ is the a posteriori probability that the correct class is ω_i when x_n is observed

Hence, this MLP can be used as a classifier by simply selecting which output is maximum (Maximum A Posteriori probability, or MAP) and assigning the corresponding class to the input pattern.

Let us illustrate this with a two-class problem (ω_1 and ω_2), in a one-dimensional space.

$$\begin{aligned} x_n \in w_1 &\Leftrightarrow p(w_1|x_n) > p(w_2|x_n) \\ &\Leftrightarrow \frac{p(x_n|w_1)p(w_1)}{p(x_n)} > \frac{p(x_n|w_2)p(w_2)}{p(x_n)} \\ &\Leftrightarrow p(x_n|w_1)p(w_1) > p(x_n|w_2)p(w_2) \end{aligned}$$

Figure 5 represents these functions [14]. The continuous lines correspond to the actual a posteriori probabilities, the dashed lines correspond to the a posteriori probabilities estimated by the MLP. The intersection of the two estimated distributions provides a possibly non-optimal decision point x*, which divides the space into two regions R_1 and R_2.

There are two ways in which a classification error can occur; either an observation x_n falls in R_2 and the true state of nature is ω_1, or x_n falls in R_1 and the true state of nature is ω_2. Since these events are mutually exclusive and exhaustive, the probability of error is [14]:

$$\begin{aligned} p(error) &= p(x_n \in R_2, w_1) + p(x_n \in R_1, w_2) \\ &= p(x_n \in R_2|w_1)p(w_1) + p(x_n \in R_1|w_2)p(w_2) \\ &= \int_{R_2} p(x_n|w_1)p(w_1)\,dx_n + \int_{R_1} p(x_n|w_2)p(w_2)\,dx_n \end{aligned}$$

The two integrals in the last relation represent the areas in the tails of the functions $p(x_n|w_i)p(w_i)$. Because the decision point x* was derived from an approximation of the a posteriori probabilities (intersection of the dashed lines), the probability of error is not as small as it might be. By moving the decision boundary to the left, we could eliminate the triangular dark red area and reduce the probability of error. The lowest possible error is obtained for the (unknown) decision point x, defined by the intersection of the continuous lines corresponding to the actual a posteriori probabilities. This could only be reach by an ideal Bayesian classifier.

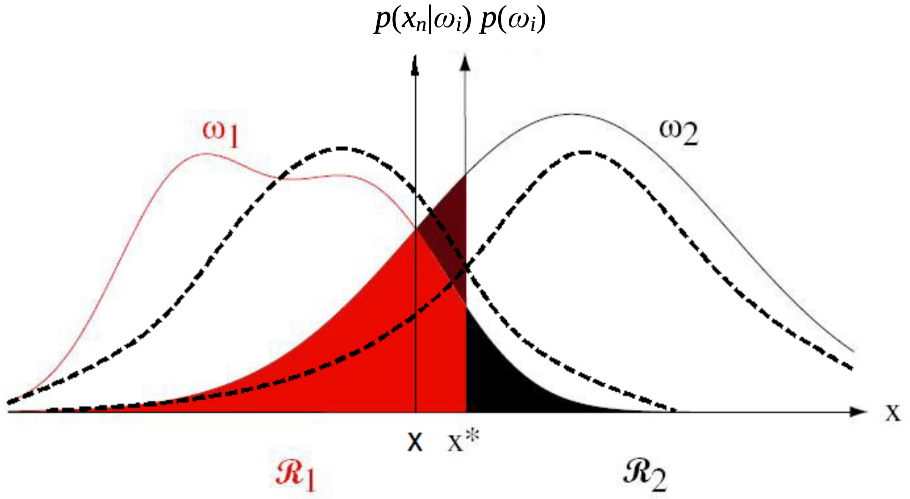

Figure 5 : Components of the probability of error for equal a priori probability classifiers and (non-optimal) decision point x*. The "optimal" decision boundaries corresponds to score values x of equal actual a posteriori probabilities

The MAP selection from the posteriors estimated by the MLP trained in the conditions described above is thus an approximation of the Bayes optimal classification.

4. MLP based similarity measure (MLP-s)

A distance metric such that higher distance values correspond to more distant objects is called a measure of dissimilarity between these objects. The Euclidian metric, for instance, is a dissimilarity measure. Conversely, a distance metric such that higher distance values correspond to closer objects is called a measure of similarity between these objects.

In this section, we consider using an MLP to compute the probability that two input feature vectors are part of the same phonetic class or not. This MLP output can thus be used as a distance metric for kNN. More precisely it provides a measure of similarity between the two input feature vectors. In the following, this MLP will be called MLP-s (for MLP-similarity)

Its inputs are the components of the two acoustic (PLP) or posterior vectors. Its output provides the similarity measure between these vectors. It is trained over a set of training vectors pairs. For each input pair, the target output of the MLP-s is fixed to 1 when the two vectors in the pair belong to the same class, and to 0 when they belong to different classes.

The structure of the MLP-s is illustrated at Figure 6.

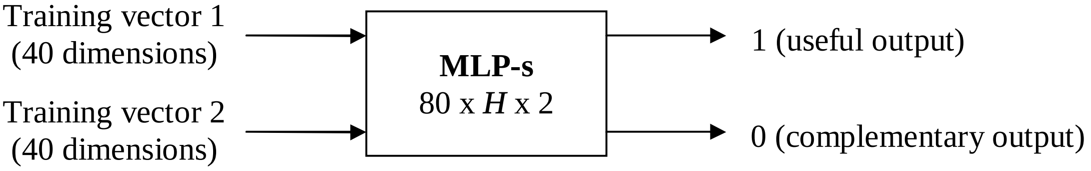

Figure 6: The MLP-s is trained to produce a 1 at its output if the two input vectors belong to the same class and 0 in the opposite case.

In this section, we will give an interpretation of the MLP based similarity measure, leading to an equivalent and very simple metric which can be analytically computed without needing the use of a neural network. This new metric is simply the scalar product of the (estimated) posterior vectors associated to the two input feature vectors (whatever they are, i.e. PLP, MFCC, and even posteriors features …). The scalar product of their posterior vectors is a measure of similarity between two feature vectors.

4.1. MLP based similarity measure

Let $X = \{x_1,\ldots,x_n,\ldots,x_N\}$ be a sample set of N feature vectors in a D dimensions feature space, drawn independently and identically distributed (i.i.d) according to a probability law $p(x)$. Each vector belongs to one of C possible classes ω_c ($c = 1, \ldots, C$).

The probability law of the population can be written $p(x) = \sum_{c=1}^{C} p(x|w_c)p(w_c)$,

where:

- $p(x|w_c)$ is the class-conditional probability density function for x
- $p(w_c)$ is the class a priori probability

Let $\{(x_i,x_j)\}$ be a set of M pairs of feature vectors, made from X.

Each pair (x_i,x_j) belongs to one of 2 possible classes:

- The "same class" pairs $\Omega_s = \{(x_i,x_j)|x_i \in w_k, x_j \in w_m, k = m\}$
- The "different class" pairs $\Omega_d = \{(x_i,x_j)|x_i \in w_k, x_j \in w_m, k \neq m\}$

The MLP-s has

- $2D$ input
- 1 output q

For a given input pattern (x_i,x_j) the observed output is noted $g(x_i,x_j)$, and $t(x_i,x_j)$ is the corresponding target output (used for training the MLP), as already explained in Section 2.3.2.

In the present case: $t(x_i,x_j) = d_{km}$ if $\begin{cases} x_i \in w_m \\ x_j \in w_k \end{cases}$ (4.1)

Let us consider training of the MLP parameters based on the minimization of the Mean Squared-Error (MSE) over all the training patterns $\{(x_i,x_j)\}$:

$$E = \sum_{i,j} \left(g(x_i,x_j) - t(x_i,x_j)\right)^2 \cdot p(x_i,x_j) \qquad (4.2)$$

Since x_i and x_j are drawn independently, $p(x_i,x_j) = p(x_i).p(x_j)$ (4.3)

But $p(x) = \sum_{c=1}^{C} p(x|w_c) \cdot p(w_c)$ with C the number of classes

and $p(x|w_c) \cdot p(w_c) = p(w_c|x) \cdot p(x)$ (Bayes relation)

thus $p(x) = \sum_{c=1}^{C} p(w_c|x) \cdot p(x)$ (4.4)

Taking into account (4.1), (4.3) and (4.4), the relation (4.2) becomes:

$$E = \sum_{i,j} p(x_i) \cdot p(x_j) \cdot \sum_{k=1}^{C} p(w_k|x_i) \cdot \sum_{m=1}^{C} p(w_m|x_j) \cdot (g(x_i,x_j) - d_{km})^2$$

$$E = \sum_{i,j} p(x_i) \cdot p(x_j) \cdot \sum_{k=1}^{C} p(w_k|x_i) \cdot \sum_{m=1}^{C} p(w_m|x_j) \cdot (g^2(x_i,x_j) - 2g(x_i,x_j)d_{km} + d_{km}{}^2)$$

$$E=\sum_{i,j} p(x_i)\cdot p(x_j)\cdot \sum_{k=1}^{C} p(w_k|x_i)\cdot$$

$$\underbrace{\left(g^2(x_i,x_j)\sum_{m=1}^{C} p(w_m|x_j)-2g(x_i,x_j)\sum_{m=1}^{C} d_{km}\,p(w_m|x_j)+\sum_{m=1}^{C} d_{km}^{\ 2}\,p(w_m|x_j)\right)}_{(4.5)}$$

Since
$$\sum_{m=1}^{C} p(w_m|x_j)=1$$
$$\sum_{m=1}^{C} d_{km}\,p(w_m|x_j)=p(w_k|x_j)$$
$$d_{km}^{\ 2}=d_{km}$$

the factor (4.5) becomes:

$$\begin{aligned}
&g^2(x_i,x_j)-2g(x_i,x_j)\cdot p(w_k|x_j)+p(w_k|x_j)\\
=\quad &g^2(x_i,x_j)-2g(x_i,x_j)\cdot p(w_k|x_j)+p(w_k|x_j)^2-p(w_k|x_j)^2+p(w_k|x_j)\\
=\quad &\left(g(x_i,x_j)-p(w_k|x_j)\right)^2+p(w_k|x_j)\cdot\left(1-p(w_k|x_j)\right)
\end{aligned}$$

Thus:

$$\begin{aligned}
E=&\sum_{i,j} p(x_i)\cdot p(x_j)\cdot\sum_{k=1}^{C} p(w_k|x_i)\cdot\left(g(x_i,x_j)-p(w_k|x_j)\right)^2\\
&+\sum_{i,j} p(x_i)\cdot p(x_j)\cdot\sum_{k=1}^{C} p(w_k|x_i)\cdot p(w_k|x_j)\cdot\left(1-p(w_k|x_j)\right)
\end{aligned}$$

Since the second term is independent of the MLP output, minimization of the mean squared-error cost function is achieved by choosing MLP parameters to minimize the first expectation term:

$$\begin{aligned}
\frac{\partial E}{\partial g(x_i,x_j)}&=\frac{\partial}{\partial g(x_i,x_j)}\left(\sum_{i,j} p(x_i)\cdot p(x_j)\cdot\sum_{k=1}^{C} p(w_k|x_i)\cdot\left(g(x_i,x_j)-p(w_k|x_j)\right)^2\right)\\
&=p(x_i)\cdot p(x_j)\cdot\sum_{k=1}^{C} 2p(w_k|x_i)\left(g(x_i,x_j)-p(w_k|x_j)\right)\\
&=0
\end{aligned}$$

$$\Rightarrow \sum_{k=1}^{C} p(w_k|x_i)\cdot g^{opt}(x_i,x_j)=\sum_{k=1}^{C} p(w_k|x_i)\cdot p(w_k|x_j)\qquad \text{if } p(x_i)\cdot p(x_j)\neq 0$$

$$\Rightarrow g^{opt}(x_i,x_j)\cdot\underbrace{\sum_{k=1}^{C} p(w_k|x_i)}_{=1}=\sum_{k=1}^{C} p(w_k|x_i)\cdot p(w_k|x_j)$$

$$\Rightarrow\qquad \boxed{g^{opt}(x_i,x_j)=\sum_{k=1}^{C} p(w_k|x_i)\cdot p(w_k|x_j)}\qquad (4.6)$$

Of course, this optimal output value can only be reached if the MLP has enough parameters, does not get stuck in a local minimum during the training, and is trained long enough to reach the global minimum [7]. We also note that a balanced training set, in which we have selected equal numbers of examples from both “same class” pairs Ω_s and “different class” Ω_d pairs, is required to achieve an accurate MLP-s training. This compensation for Ω_s and Ω_d class priors will be analyzed in Section 5.

4.2. Interpretation of the MLP-s output as a similarity measure

Let us now consider the posterior vector $\overrightarrow{Px_n} = \left(p(w_1|x_n), p(w_2|x_n),..., p(w_C|x_n)\right)$ associated to the feature vector x_n. The ith component of $\overrightarrow{Px_n}$ is the (actual) a posteriori probability $p(w_i|x_n)$ that the state of nature be ω_i, given x_n. $\overrightarrow{Px_n}$ is the representation of x_n in the posterior space (dimension = C). Its extremity belongs to an hyperplane (dimension = C-1) defined by $\sum_{k=1}^{C} p(w_k|x_n) = 1$.

With this notation, (4.5) may be rewritten as

$$\boxed{g^{opt}(x_i, x_j) = \overrightarrow{Px_i} \cdot \overrightarrow{Px_j}} \qquad (4.7)$$

and interpreted as follows: **the output of the MLP, trained in the conditions described above, is an estimation of the scalar product of the 2 (actual) posterior vectors associated with the 2 input feature vectors**.

These conclusions are valid independently of the type of input features (MFCC, PLP, posteriors) used at the input of the MLP.

4.3. The Posterior Scalar Product (PSP) metric

The interpretation given in the previous section leads us to the definition of an equivalent and very simple metric that can be analytically computed without the need of a neural network.

Considering the posterior vector $\overrightarrow{Px_n} = \left(p(w_1|x_n), p(w_2|x_n),..., p(w_C|x_n)\right)$ associated with the feature vector x_n, we have:

$$\overrightarrow{Px_i} \cdot \overrightarrow{Px_j} = \sum_{k=1}^{C} p(w_k|x_i) \cdot p(w_k|x_j)$$

by definition of the scalar product.

Actually, $p(w_k|x_i)\cdot p(w_k|x_j)$ represents the probability that both x_i *and* x_j belong to the same class ω_k. As a consequence, $\sum_{k=1}^{C} p(w_k|x_i)\cdot p(w_k|x_j)$ is the probability that x_i and x_j belong to the same class, whatever the class is.

In other words, **the probability that two feature vectors belong to the same class is simply given by the scalar product of their associated posterior vectors.**

Defined in this way, the scalar product can be considered as a new distance metric, which will be referred to as **Posterior Scalar Product (PSP)** metric in the sequel of this document.

5. Compensating for class priors

If we use all the possible pairs made from the set *X* of sample vectors to train our model, we could run into severe difficulties due to the small proportion of the "same class" pairs in our training set.

For instance, if *X* contains *n* vectors of each class, this proportion is $\frac{Cn^2}{Cn^2+C(C-1)n^2}=\frac{1}{C}$ and

- Only one in every *C* pairs corresponds to the "same class" Ω_s
- *C*-1 in every *C* pairs correspond to the "different class" Ω_d

For *C* = 40, the proportion of "same class" pairs is equal to 2.5%, versus 97.5% of "different-class" pairs. In this case, the learning algorithm will not be exposed to a broad range of examples of "same class" pairs and hence is not likely to generalize well. A classifier that assigns every pair to the "different class" would already achieve 97.5% accuracy and it would be difficult to avoid this trivial solution. A balanced pairs set in which we have selected equal numbers of examples from both classes Ω_s and Ω_d would allow us to find a more accurate model [5].

5.1. Pairs creation / selection

In this paragraph, we analyze the way to achieve the Ω_s and Ω_d priors compensation, using the notation introduced in Section 4.1.

Consider

- The subset X_s of *X*, made of the firsts s_k vectors from each class ω_k (k = 1 … *C*)
- The subset X_d of X_s, made of the firsts d_k vectors from each class ω_k (k = 1 … *C*)

with the constraint $\frac{s_k}{d_k}=a=\text{constant}\ \forall k$.

$N_s=\sum_{k=1}^{C} s_k$ is the number of elements in subset X_s.

$p_k = \frac{s_k}{N_s}$ is the prior of ω_k in subset X_s (k = 1 … C). We see immediately that $\sum_{k=1}^{C} p_k = 1$.

The number of all possible “same class” pairs that can be built up from X_s is:

$$M_s = \sum_{k=1}^{C} s_k^{\,2} = N_s^{\,2} \sum_{k=1}^{C} p_k^{\,2} \tag{5.1}$$

The number of all possible “different class” pairs that can be built up from X_d is:

$$M_d = \sum_{k=1}^{C} \sum_{\substack{m=1 \\ m \neq k}}^{C} d_k d_m = \sum_{k=1}^{C} d_k \sum_{\substack{m=1 \\ m \neq k}}^{C} d_m = \sum_{k=1}^{C} d_k \left(\sum_{m=1}^{C} d_m - d_k \right) = \left(\sum_{k=1}^{C} d_k \right)^2 - \sum_{k=1}^{C} d_k^{\,2}$$

$$M_d = \frac{1}{a^2} \left(\sum_{k=1}^{C} s_k \right)^2 - \frac{1}{a^2} \sum_{k=1}^{C} s_k^{\,2}$$

$$M_d = \frac{N_s^{\,2}}{a^2} \left(\sum_{k=1}^{C} p_k \right)^2 - \frac{N_s^{\,2}}{a^2} \sum_{k=1}^{C} p_k^{\,2}$$

$$M_d = \frac{N_s^{\,2}}{a^2} \left(1 - \sum_{k=1}^{C} p_k^{\,2} \right) \tag{5.2}$$

The priors compensation $p(\Omega_s) = p(\Omega_d) = \frac{1}{2}$ implies $\frac{M_s}{M} = \frac{M_d}{M}$, hence $M_s = M_d$

$$\Rightarrow \quad \sum_{k=1}^{C} p_k^{\,2} = \frac{1}{a^2} \left(1 - \sum_{k=1}^{C} p_k^{\,2} \right)$$

$$\Rightarrow \quad a = \sqrt{\frac{\left(1 - \sum_{k=1}^{C} p_k^{\,2} \right)}{\sum_{k=1}^{C} p_k^{\,2}}} \tag{5.3}$$

The total number of pairs is $M = M_s + M_d = 2M_s = 2N_s^{\,2} \sum_{k=1}^{C} p_k^{\,2}$

$$\Rightarrow \quad N_s = \sqrt{\frac{M}{2 \sum_{k=1}^{C} p_k^{\,2}}} \tag{5.4}$$

Finally, the creation of a set of M balanced pairs requires:

$$s_k = N_s p_k = \sqrt{\frac{M}{2 \sum_{k=1}^{C} p_k^{\,2}}}\, p_k \qquad \text{and} \qquad d_k = \frac{s_k}{a} = \sqrt{\frac{M}{2 \left(1 - \sum_{k=1}^{C} p_k^{\,2} \right)}}\, p_k \tag{5.5}$$

Approach 1: $p_k = \frac{1}{C}$

$$\Rightarrow\ s_k = \sqrt{\frac{M}{2C}} \quad \text{and} \quad d_k = \sqrt{\frac{M}{2C(C-1)}} \tag{5.6}$$

This approach represents the easiest way to create a set of balanced pairs, as we just have to take the same number of vectors from each class ω_k (s_k and d_k are independent of k). But in this case, the priors p_k of subset X_s and X_d are not representative of the actual priors $p(\omega_k)$ of the population.

In this approach, we have neutralized the priors $p(\Omega)$, but also the priors $p(w)$.

Approach 2: $p_k = p(w_k)$

$$\Rightarrow\ s_k = \sqrt{\frac{M}{2\sum_{k=1}^{C} p(w_k)^2} p(w_k)} \quad \text{and} \quad d_k = \sqrt{\frac{M}{2\left(1-\sum_{k=1}^{C} p(w_k)^2\right)} p(w_k)} \tag{5.7}$$

In this approach, we have neutralized the priors $p(\Omega)$ only. The priors p_k of subset X_s and X_d are representative of the actual priors $p(\omega_k)$ of the population.

Remembering that X, hence X_s and X_d, are sets of feature vectors drawn independently and identically distributed (i.i.d) according to a probability law of the population $p(x) = \sum_{c=1}^{C} p(x|w_c)p(w_c)$, both approach respect the class-conditional probability densities $p(x|w_c)$.

5.2. Effect of pair priors compensation on MLP-s output

The MLP-s optimal output (4.5) only depends on the a posteriori probabilities $p(w_c|x)$, hence only on the class-conditional densities $p(x|w_c)$ and on the priors $p(w_c)$. The pair priors compensation performed according to Approach 2 has thus no effect on the MLP-s optimal output, and should be preferred to Approach 1 for MLP-s training.

6. Histogram-based hypothesis tests

In this section, we will focus on the following problem: "Given two feature vectors, what is the probability that these belong to the same (phonetic) class or not, whatever the class?". In order to achieve this goal, we will classify pairs (of feature vectors) in two classes (“same class” and “different classes”). Again, we will use the notations introduced in Sections 4.1 and 5.1.

6.1. Hypothesis test

Each pair (x_i,x_j) belongs to one of two possible classes: Ω_s (the "same class" pairs) and Ω_d (the "different class" pairs).

One interesting feature that can be extracted from these pairs, is the distance $\mathbf{l}$ between their two components. In the following, each pair will be represented by this single feature, in a one dimensional feature space. The objective is to classify the *M* pairs on the basis of this single feature.

Let $L = \{\mathbf{l}_1,...,\mathbf{l}_m,...,\mathbf{l}_M\}$ be the set of *M* scalar features representing the *M* pairs.
The "optimal" decision point $\mathbf{l}^*$ is the point of equal a posteriori probabilities (Bayes decision), that is:

$$p(\Omega_s|\mathbf{l}^*) = p(\Omega_d|\mathbf{l}^*) \qquad (6.1)$$

$$\Rightarrow \quad \frac{p(\mathbf{l}^*|\Omega_s)P(\Omega_s)}{p(\mathbf{l}^*)} = \frac{p(\mathbf{l}^*|\Omega_d)p(\Omega_d)}{p(\mathbf{l}^*)}$$

$$\Rightarrow \quad p(\mathbf{l}^*|\Omega_s)p(\Omega_s) = p(\mathbf{l}^*|\Omega_d)p(\Omega_d) \qquad (6.2)$$

The hypothesis test is performed in two main steps:

- In the training phase, $\mathbf{l}^*$ is assessed using the training pairs made from a set of training vectors
- In the test phase, the classification accuracy is assessed using the test pairs made from a set of test vectors independently drawn from the same population

6.2. Compensating for class priors

For the raisons explained in section 5, an accurate model for $\mathbf{l}^*$ assessment requires a balanced pairs set in which

$$p(\Omega_s) = p(\Omega_d) = \frac{1}{2} \qquad (6.3)$$

$$\Rightarrow \quad M_s = M_d \qquad (6.4)$$

This can be achieved by selecting the *M* pairs (x_i,x_j) in the way described in section 5.1.

In this case:

$$p(\mathbf{l}^*|\Omega_s) = p(\mathbf{l}^*|\Omega_d) \qquad (6.5)$$

In the frame of the hypothesis test, the effect of this priors compensation is automatically compensated if the same balancing is applied to both the training and the test pairs sets, provided that the probability densities $p(\mathbf{l}|\Omega)$ remain unchanged under this priors modification.

In order to verify this latter condition, let us consider the scalar random variable *L*, function of the two vectorial random variables X_i and X_j : $L = d(X_i , X_j)$. Realized values of *L* are related to realized values of X_i and X_j as follows: $\lambda = d(x_i, x_j)$. The function $d(x_i , x_j)$ represents a distance between vectors x_i and x_j .x_i and x_j are drawn independently according to the same probability law $p(x) = \sum_{c=1}^{C} p(x|w_c)p(w_c)$, hence $p(x_i, x_j) = p(x_i|x_j)p(x_j) = p(x_i)p(x_j)$

By definition, the distribution function of the variable *L* is:

$$\begin{aligned} P_L(\lambda) &= P(L \le \lambda) \\ &= P\big(d(X_i, X_j) \le \lambda\big) \\ &= \int_{R^D}\left[\int_{\Theta(x_i,\lambda)} p(x_i)p(x_j)dV_j\right]dV_i \end{aligned}$$

where:

- *D* is the dimension of x_i
- Θ is the region of R^D such that $d(x_i, x_j) \le \lambda$. It is the set of vectors x_j whose distance to a given x_i is less or equal to the given λ. Here, $\Theta = \Theta(x_i, \lambda)$ is a function of $\mathrm{x_i}$ and λ

$$P_L(\lambda) = \int_{R^D} p(x_i)\left[\int_{\Theta(x_i,\lambda)} p(x_j)dV_j\right]dV_i$$

And, since $p(x) = \sum_{c=1}^{C} p(x|w_c)p(w_c)$,

$$P_L(\lambda) = \int_{R^D}\sum_{k=1}^{C} p(x_i|w_k)p(w_k)\left[\int_{\Theta(x_i,\lambda)}\sum_{m=1}^{C} p(x_j|w_m)p(w_m)dV_j\right]dV_i$$

$$P_L(\lambda) = \sum_{k=1}^{C} p(w_k)\sum_{m=1}^{C} p(w_m)\int_{R^D} p(x_i|w_k)\left[\int_{\Theta(x_i,\lambda)} p(x_j|w_m)dV_j\right]dV_i$$

The probability density function is the derivative of the probability distribution function:

$$p_L(\lambda) = \frac{\partial P_L(\lambda)}{\partial \lambda}$$

$$p_L(\lambda) = \sum_{k=1}^{C} p(w_k)\sum_{m=1}^{C} p(w_m)\underbrace{\int_{R^D} p(x_i|w_k)\left[\frac{\partial}{\partial \lambda}\int_{\Theta(x_i,\lambda)} p(x_j|w_m)dV_j\right]dV_i}_{(6.6)} \qquad (6.7)$$

The factor (6.6) is a function of $p(x|w_k)$, $p(x|w_m)$, λ and of the distance *d*.

The relation (6.7) may be rewritten:

$$p_L(\lambda) = \sum_{k=1}^{C} p(w_k)\sum_{m=1}^{C} p(w_m)H\big(p(x|w_k), p(x|w_m), \lambda, d\big)$$

Finally,

$$p(\mathbf{l}|\Omega_s) = \sum_{k=1}^{C} p(w_k)^2 H\big(p(x|w_k), p(x|w_k), \mathbf{l}, d\big)$$

and $$p(\mathbf{l}|\Omega_d) = \sum_{k=1}^{C} p(w_k) \sum_{\substack{m=1\\ m\neq k}}^{C} p(w_m) H\big(p(x|w_k), p(x|w_m), \mathbf{l}, d\big)$$

For a given metric *d*, the probability densities $p(\mathbf{l}|\Omega)$ only depend on the class-conditional densities $p(x|w_c)$ and on the priors $p(w_c)$. The pair priors compensation performed according to Approach 2 has thus no effect on the densities $p(\mathbf{l}|\Omega)$, and should be preferred to Approach 1 for MLP-s training.

6.3. Experimental decision point evaluation

A probability distribution *p*(*x*) may be modelled by a standard histogram, simply by partitioning *x* into distinct bins of width Δ_i and then by counting the number n_i of observations of *x* falling in bin *i*. In order to turn this count into a normalized probability density, we simply divide by the total number *M* of observations and by the width Δ_i of the bins to obtain probability values for each bin, given by:

$$p_i = \frac{n_i}{M\Delta_i}$$

This gives a model for the density *p*(*x*) that is constant over the width of each bin. Generally, the bins are chosen to have the same width $\Delta_i = \Delta$.
This is one of the non parametric approaches to density estimation. These approaches make less assumption about the form of the distribution and thus have less limitation than the parametric approaches [5].

Applied to our case, the density $p(\mathbf{l}|\Omega_s)$ can be modelled by a histogram, by partitioning $\mathbf{l}$ into distinct bins of width Δ and by counting the number n_{si} of observations of $\mathbf{l} \in \Omega_s$ falling in bin *i*. Thus:

$$p(\mathbf{l}|\Omega_s) \cong \left\{\frac{n_{si}}{M_s\Delta}\right\}$$

Similarly: $$p(\mathbf{l}|\Omega_d) \cong \left\{\frac{n_{di}}{M_d\Delta}\right\}$$

This is illustrated in Figure 7, showing the histograms approximation of the continuous distributions $p(\mathbf{l}|\Omega_s)$ (green) and $p(\mathbf{l}|\Omega_d)$ (red).

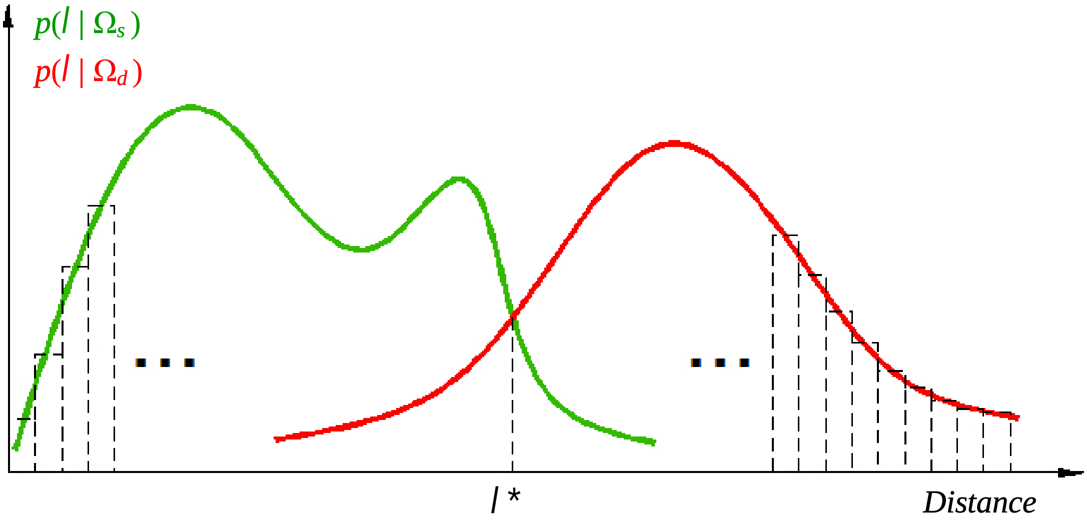

Figure 7 : Histogram of $p(\mathbf{l}|\Omega_s)$ (green) and $p(\mathbf{l}|\Omega_d)$ (red) versus the distance l

From (6.5), $\mathbf{l}*$ is determined by:

$$p(\mathbf{l}*|\Omega_s) = p(\mathbf{l}*|\Omega_d)$$

$$\Rightarrow \quad \frac{n_{si}}{M_s\Delta} = \frac{n_{di}}{M_d\Delta}$$

$$\Rightarrow \quad \frac{n_{si}}{M_s} = \frac{n_{di}}{M_d} \quad \text{and, since } M_s = M_d :$$

$$\Rightarrow \quad n_{si} = n_{di} \qquad (6.8)$$

$\mathbf{l}*$ is thus determined by the intersection of the two histograms obtained by counting the number n_{si} (respectively n_{di}) of observations of $\mathbf{l} \in \Omega_s$ (respectively $\mathbf{l} \in \Omega_d$) falling in bin *i*.

6.4. Experimental set up

In this work, a comparative analysis will be conducted for:

- Different types of feature vectors: posterior vectors, PLP vectors
- The different types of "distances" previously mentioned: Euclidian, Mahalanobis, Kullback-Leiber, Bhattacharyya, MLP similarity and Scalar Product

We will adopt the following principles:

- $\mathbf{l}*$ is determined using the training pairs (made from the training vectors)
- The classification accuracy is performed using the cross-validation pairs (made from the cross-validation vectors)
- Pairs of both sets are selected in the same way
- The same sets are used for the comparative assessment of the different types of distances
- For the MLP similarity, the training pairs are used in a first step for training the network and, in a second step, for $\mathbf{l}*$ determination

7. Improved acoustic vectors classification

Despite the simplicity of the algorithm, the (k)-Nearest Neighbor (kNN) rule performs very well and is an important classification benchmark method. The kNN classifier, as described by [14], requires a distance metric *d*, a positive integer *k*, and the reference templates of *N* labelled patterns.

Generally, Euclidian or Mahalanobis distances have been used as local distance between feature vectors. However, the notion of a metric is far more general.
In this work, we conducted experiments to assess the potential usefulness of alternative measures of distances, the MLP similarity and the scalar product, to improve phone posterior estimation through k-NN.

Figure 8 represents the general bloc diagram of the system used to perform this analysis. It is composed of the 3 main parts: the acoustic front end, the metric component, and the classifier itself.

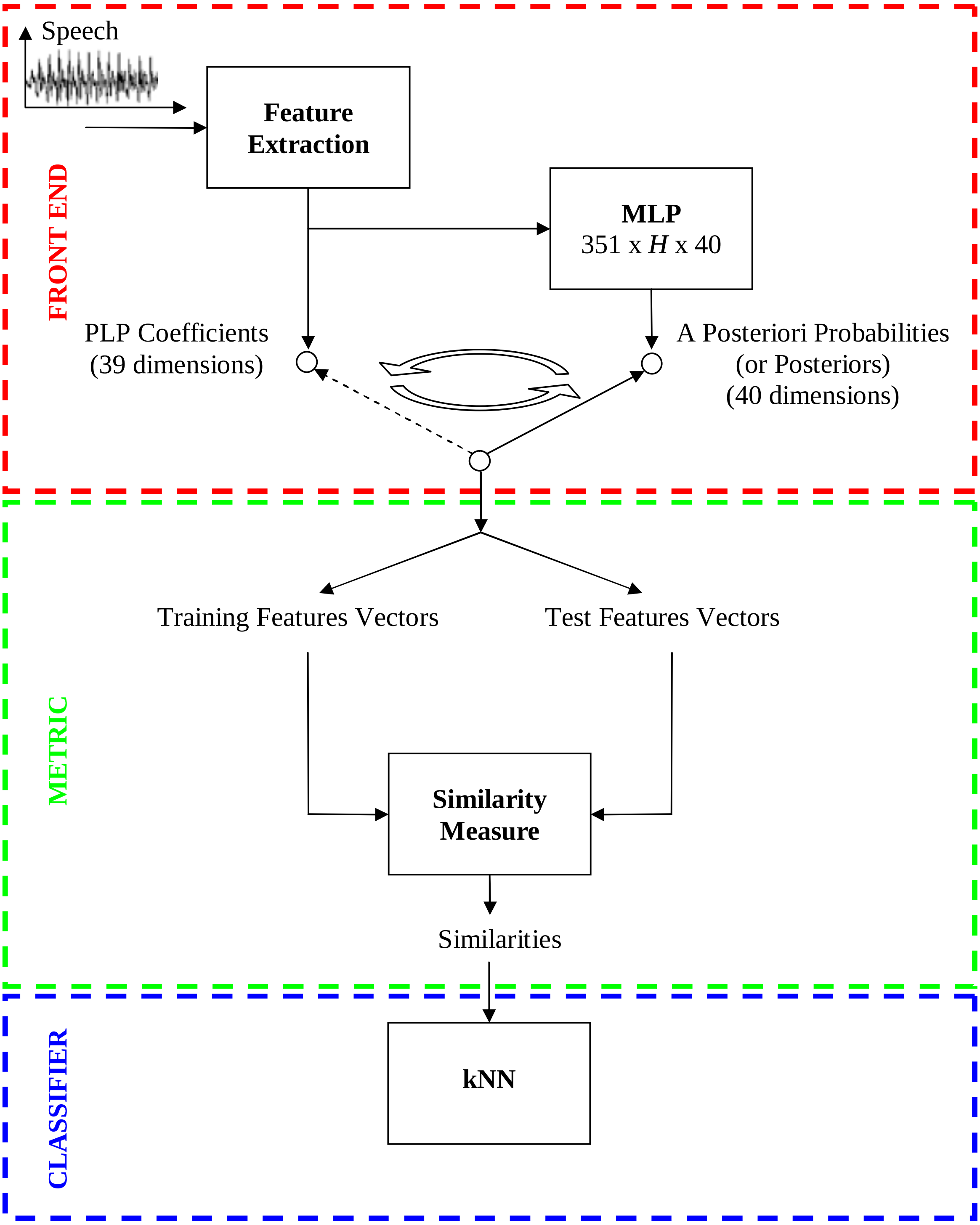

Figure 8 : General block diagram

7.1. Acoustic front end

The acoustic front end transforms the input speech signal into robust feature vectors. It provides a set of labelled PLP feature vectors and a set of labelled Posterior feature vectors

- The PLP feature vectors are extracted from the speech signal as described in section 3.1
- The Posterior feature vectors are estimated from the PLP coefficients, by an MLP trained in the conditions given in section 3.2

Both sets are split in three subsets: the training vectors, the cross-validation vectors and the tests vectors

7.2. Distance metric

The second component measures the distance between training (considered as reference templates for the kNN classifier) and test or cross-validation vectors.
We focus our investigations on the MLP-based similarity (described in section 4) and the scalar product similarity (see section 4.2), between Posterior or PLP feature vectors.

The MLP-s requires a preliminary training, based on a balanced set of training vectors pairs, selected according to section 5.

Our results will be compared with those obtained with more traditional distance measures (Euclidian, Mahalanobis, Bhattacharyya, Kullback-Leibler)[2].

7.3. (k)-Nearest Neighbor

Using the selected metric (scalar product or MLP-s), and the selected type of feature vectors (Posterior or PLP), the kNN method is able to provide a (new) estimation of the a posteriori probabilities from the set of *n* labelled samples.

Indeed, suppose that we place a cell of volume *V* around *x* and capture *k* samples, k_i of which turn out to be labelled ω_i [14]. Then the obvious estimate for the joint probability $p(x, \omega_i)$ is

$$p_n(x, w_i) = \frac{k_i / n}{V}$$

thus, a reasonable estimate for $P(\omega_i|x)$ is

$$P_n(w_i \mid x) = \frac{p_n(x, w_i)}{\sum_{j=1}^{C} p_n(x, w_j)} = \frac{k_i}{k}$$

[2] Thank you to Mrs Afsaneh Asaei for contributing to this part of the work.

That is, the estimate of the a posteriori probability that ω_i is the state of nature is merely the fraction of the samples within the cell that are labelled ω_i. For minimum error rate, we select the category most frequently represented within the cell. If there are enough samples and if the cell is sufficiently small, it can be shown that this will yield performance approaching the best possible [14].

Another point of view, the kNN rule allows to go directly to the decision by assigning to the vector to be classified (tests vector), the label most frequently represented among the k nearest samples (training vectors).

8. Experiments

In this section, we present all the experimental results related to the previous sections. We first describe the database we used (Section 8.1). After that, the configuration of the features extraction process is given (Section 8.2), and the parameters of the MLP that estimates the posteriors are shown in Section 8.3. Then, we explain how we created our training and test data, consisting of vectors pairs (Section 8.4). Section 8.5 is dedicated to hypothesis tests. We conducted our experiments on two types of features (posteriors and PLP) and on different types of distances / similarities (Euclidian, Mahalanobis, Kullback-Leibler, Bhattacharyya, MLP-based, Scalar Product). Finally, Section 8.6 is dedicated to kNN classification, using the same kind of features and distances / similarities as for the hypothesis tests.

8.1. TIMIT database

The database used throughout this master's thesis is called TIMIT [35]. It consists of a 16-bit, 16 kHz speech waveform file for each sentence utterance. A more detailed description of the whole TIMIT database is given in Appendix 1.

Experiments were performed on a modified version of TIMIT, excluding the 'sa' dialect sentences [33]. The training data consists of 3,000 utterances from 375 speakers, the cross-validation data set consists of 696 utterances from 87 speakers and the test data set consists of 1,344 utterances from 168 speakers. Table 1 shows the number of sentences and frames per set (training, cross-validation and test). These numbers are of course the same for posterior and PLP feature vectors. The TIMIT database, which is hand-labeled using 61 labels, is mapped to the standard set of 40 phonemes as explained in [31], except in the way the closures are handled. In our case, when a closure occurs before its own burst, the closure and the burst are merged (*e.g.* /tcl t/ à /t/). On the other hand, if a closure precedes any phoneme other than its own burst, the closure is mapped to its burst (*e.g.* /pcl t/ à /p t/).

	Number of sentences	Number of frames
Training set	3,000	920,166
Cross-Validation set	696	204,657
Test set	1,344	410,920

Table 1 : Number of sentences and frames for the training, cross-validation and test set of posterior and PLP feature vectors

8.2. Features extraction

The general ideas of features extraction have been explained in details in Section 3.1. Each acoustic vector contains 39 components: 13 PLP coefficients and 26 dynamic features (13 delta features + 13 double delta features).

These components were extracted using HTK toolkit, configured as follows (the complete configuration file is given in Appendix 3):

- Analysis window:
    - Length: 25 ms

    - Type: Hamming
    - Shifting period: 10 ms
- Linear Predictive Coding (LPC) analysis:
    - Using power instead of magnitude of the Fourier transform
    - Order of the analysis: 12
    - Using a 24 channels filterbank (to obtain a non-linear frequency resolution, as human ear [6])
    - 12 cepstral coefficients + cepstral C_0 coefficient per vector

8.3. Posterior probability estimation

From these cepstral-based features, the first MLP was trained to produce a "1" on the output associated with the actual class of the input feature vector, and "0" on all the other outputs. The optimization criterion was the cross-entropy criterion.

The MLP shown in Figure 2 has been used, configured as follows:

- D = 9*39 = 351 entry units. A context window of 9 PLP-based acoustic vectors is used: the current input + the 4 previous acoustic vectors + the 4 following acoustic vectors. The choice of 9 is neither magic nor holy, it is just an optimum value for the length of the context window that many researchers have found during their experiments
- H = 2000 hidden units. This is an optimum found experimentally
- C = 40 output units, each of which representing the a posteriori probability of the associated class (phoneme), i.e. $y_1 = p(\omega_1 | X_{n-c}^{n+c})$, $y_2 = p(\omega_2 | X_{n-c}^{n+c})$, …, $y_C = p(\omega_C | X_{n-c}^{n+c})$, where ω_i stands for class i and $X_{n-c}^{n+c} = \{x_{n-c}, \ldots, x_n, \ldots, x_{n+c}\}$ represents the input feature vector consisting of a context of 9 frames (the context window c is equal to 4)

The recognition rates obtained during the training of this MLP are shown in Table 2. The optimal point P was found at epoch 8 (in green in the table).

Epoch	Learning Rate	Training Accuracy (%)	CV Accuracy (%)
1	0.0008	52.5	62.7
2	0.0008	63.0	66.1
3	0.0008	65.8	67.5
4	0.0008	67.5	68.5
5	0.0008	68.8	69.1
6	0.0008	69.8	69.4
7	0.0004	70.8	70.0
8	**0.0002**	**71.3**	**70.1**

Table 2 : Results of the MLP training for estimating the posteriors from the PLP feature vectors

As seen in Table 2, this MLP has an accuracy of around 70%. Five main reasons can explain this behaviour:

- Feature limitation: PLP features. Unless changing our mind and choosing other type of features, the PLP have been estimated using a standard procedure and they cannot be modified or improved

- Speech variability: we have speech variability between speakers and also within speaker. If we pronounced a word two times, it will be (even slightly) different. Therefore, to represent a phoneme that has a large variability, we need a lot of template data, which is not always possible to have (e.g. standardized database that cannot be modified)
- Priors problem: uneven amount of data for each class (each phoneme)
- Labelling is not perfect. We do not speak like a dictionary and the pronunciation of a work can be different from one people to another
- Segmenting sentences into phoneme is not an easy task, because of co-articulation (phoneme overlapping because of the vocal tract inertia [6])

8.4. Pairs creation

The training and cross-validation vectors pairs were created according to the principles defined in Section 5.1:

- The pair priors $p(\Omega)$ are neutralized
- Pairs are created following Approach 1 (phone priors $p(w)$ also neutralized) and Approach 2 (phone priors $p(w)$ preserved), in order to experimentally compare both approaches

A set of about 20,000,000 training pairs and a set of about 4,000,000 cross-validation pairs are created, using the following values for s_k and d_k:

- Approach 1:
  - s_k = 500 for the training set and 220 for the cross-validation set
  - d_k = 80 for the training set and 35 for the cross-validation set
- Approach 2:
  - s_k = $14562.p(w_k)$ for the training set and $6432.p(w_k)$ for the cross-validation set
  - d_k = $3240.p(w_k)$ for the training set and $1450.p(w_k)$ for the cross-validation set

8.5. Hypothesis tests

As explained in Section 6, the hypothesis test is aimed at the classification of two feature vectors as belonging or not to the same class, whatever the class. In this section, a comparative analysis is conducted for:

- Different types of feature vectors: posterior vectors, PLP vectors
- Different types of feature vectors selection: Approach 1 and Approach 2
- The different types of “distances” previously mentioned: Euclidian, Mahalanobis, Kullback-Leiber, Bhattacharyya, MLP similarity and Scalar Product

For each type of features and “distances”, we have computed:

- An estimation of the density $p(\mathbf{l}|\Omega_s)$ and $p(\mathbf{l}|\Omega_d)$, modelled by the training pairs histograms
- The mean and variance of these two distributions

- The continuous approximation of the two histograms
- Their intersection $\mathbf{l}*$
- The training and test[3] pairs classification accuracy obtained from this classification threshold $\mathbf{l}*$

In Section 8.5.2 and 8.5.3, when considering the MLP-based similarity measure, the MLP was trained using the training pairs set defined in Section 8.4. The recognition rates obtained during these trainings when using Approach 2 are given in Table 3 and Table 4, respectively for posterior and PLP feature vectors. The optimal point *P* was found at epoch 3 and epoch 1 respectively (in green in the table).

Epoch	Learning Rate	Training Accuracy (%)	CV Accuracy (%)
1	0.001	77.41	83.73
2	0.001	86.97	84.13
3	**0.0005**	**87.25**	**84.27**

Table 3 : Results of the MLP training using posteriors for hypothesis tests

Epoch	Learning Rate	Training Accuracy (%)	CV Accuracy (%)
1	**0.001**	**91.20**	**55.02**
2	0.001	96.06	53.74
3	0.0005	95.81	54.21

Table 4 : Results of the MLP training using PLP for hypothesis tests

The summary of the results and conclusions of the experimental hypothesis tests are given in Section 8.5.1. The corresponding detailed results obtained for different type of distances, when using pairs of posterior feature vectors and PLP feature vectors created according to Approach 2 are given in Section 8.5.2 and 8.5.3 respectively.

Similar graphs were obtained with Approach 1. For conciseness's sake, only final results are provided in this case, when comparing both approaches in Section 8.5.1.

8.5.1. Summary and conclusions

Table 5 summarizes the classification accuracies obtained over training and test pairs sets of posterior and PLP feature vectors, for the different metrics.

This table clearly shows that:

- The Scalar Product similarity achieves better performance than any other metric. Moreover, this similarity measure is very simple and fast to implement
- Better results are obtained with posterior feature vectors than with PLP ones
- Approach 2 provides better results than Approach 1. This results was expected since Approach 2 preserves the phone priors $p(w)$ while Approach 1 neutralizes them
- Test accuracy is always a few percents lower than training accuracy, as expected

[3] Important note: the test set referred to in this section, is the cross-validation pairs set. The tuning (threshold estimation) is performed using the training set. Nothing is tuned on the test set.

	Hypothesis test - Pairs Classification Accuracy									
Vectors / pairs selection	Distance / Pairs	Euclidian	Mahalanobis	Kullback-Leiber	Bhattacharyya	Scalar Product	MLP 20 hidden units	MLP 200 hidden units	MLP 500 hidden units	MLP 1000 hidden units
	Posterior vectors									
Approach 1	Training	75.3%	▨	83.5%	84.4%	85.3%				86.4%
	Test	71.4%	▨	81.5%	82.3%	83.8%				81.3%
Approach 2	Training	84.4%	▨	88.4%	89.3%	90.2%	89.9%	88.8%	89.0%	87.3%
	Test	78.8%	▨	85.4%	86.6%	88.5%	85.4%	85.1%	85.7%	84.3%
	PLP vectors									
Approach 2	Training	73.0%	59.6%	▨	▨	75.2%	88.6%	93.7%	94.9%	95.0%
	Test	71.0%	60.2%	▨	▨	74.6%	78.6%	72.3%	71.6%	70.5%

Table 5 : Training pairs accuracy and test pairs accuracy for the posterior and PLP feature vectors

Note 1: Training performed with 20,000,000 pairs / test performed with 4,000,000 pairs
Note 2: MLP training performed with RANDOMIZED training pairs (approach 2)
Note 3: Tuning ("threshold" evaluation) performed on training pairs, according to Section 6.3
Note 4: Kullback-Leiber & Bhattacharyya are not applicable with PLP vectors.
Note 5: Mahalanobis is not applicable to posterior vectors (matrix non invertible)

8.5.2. Using posterior feature vectors (Approach 2)

8.5.2.1. Euclidian distance

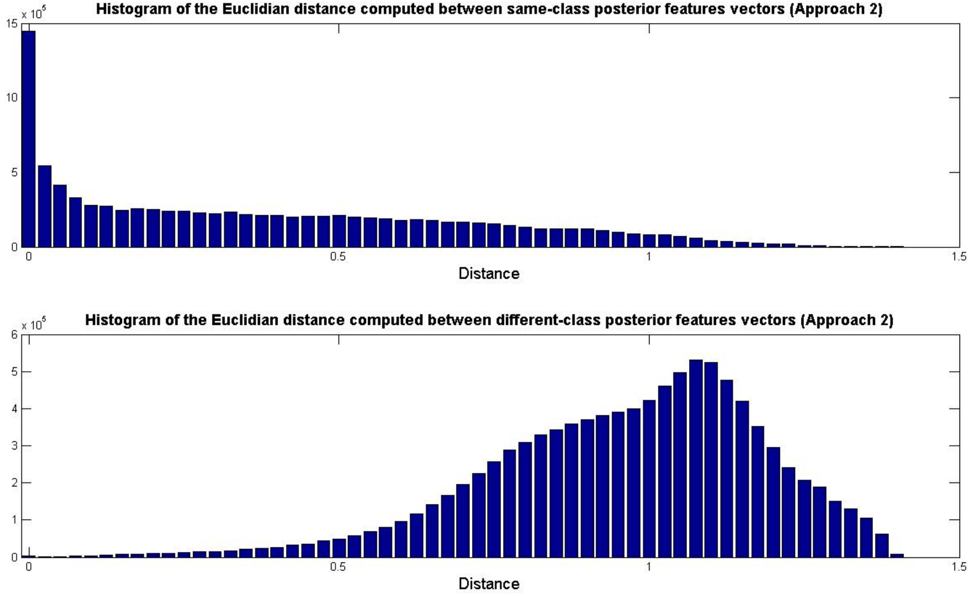

Figure 9 : Histograms of the Euclidian distance computed between same-class and different-class posterior feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	0.39	0.33
Different-class	0.98	0.22

Table 6 : Parameters of the Euclidian same-class / different-class posterior pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in $\mathbf{l}*$ = 0.7.

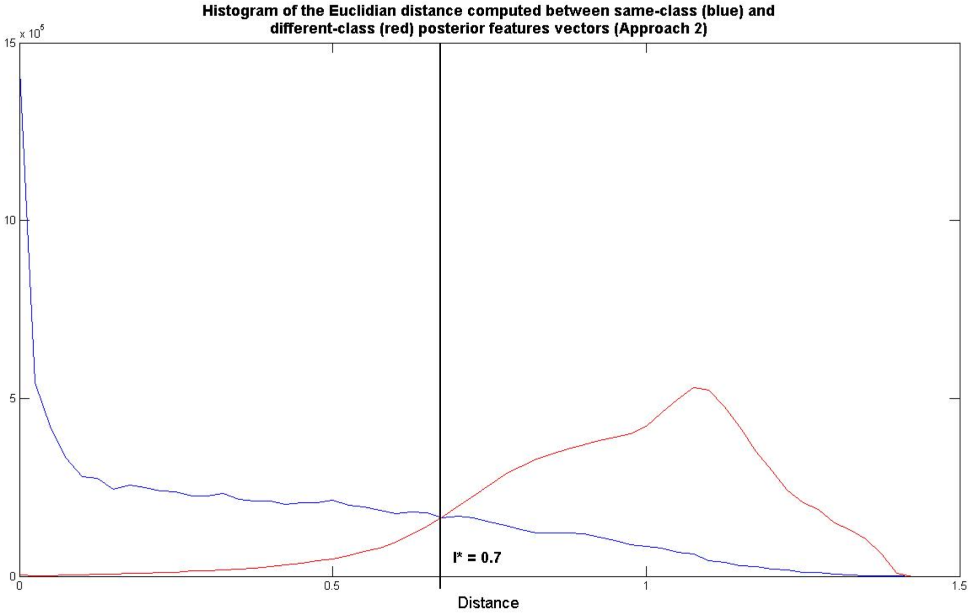

Figure 10 : Continuous approximation of the Euclidian same-class / different-class posterior pairs histogram

The classification accuracy obtained using the classification threshold $\mathbf{l}*$ is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	38.9 %	36.4 %
Different-class	45.5 %	42.4 %
Total	**84.4 %**	**78.8 %**

Table 7 : Classification accuracy for the posterior pairs training set and test set, when using Euclidian distance

8.5.2.2. Kullback-Leibler divergence

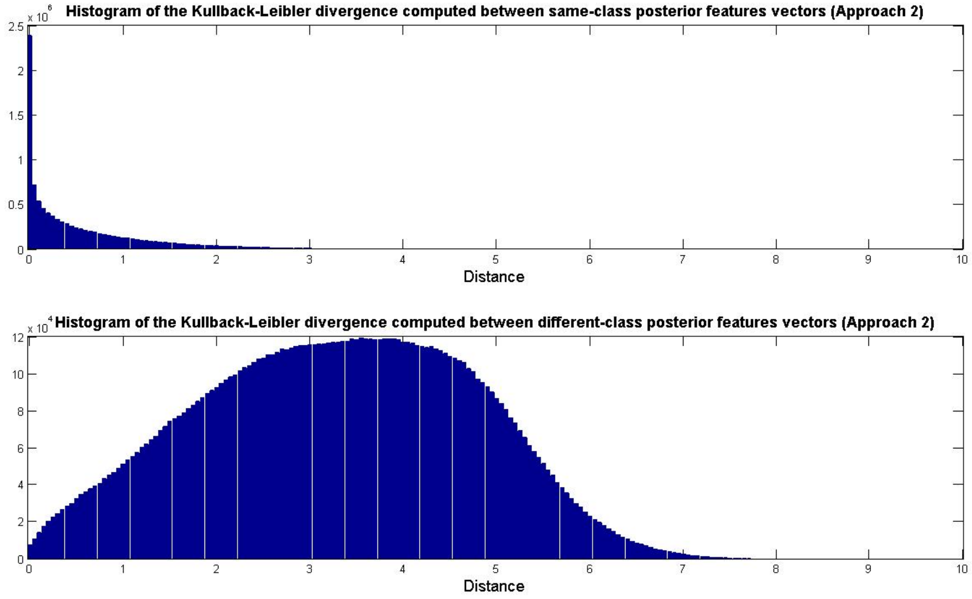

Figure 11 : Histograms of the Kullback-Leibler divergence computed between same-class and different-class posterior feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	0.59	0.71
Different-class	3.32	1.46

Table 8 : Parameters of the Kullback-Leibler same-class / different-class posterior pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in $\mathbf{l}*$ = 1.5.

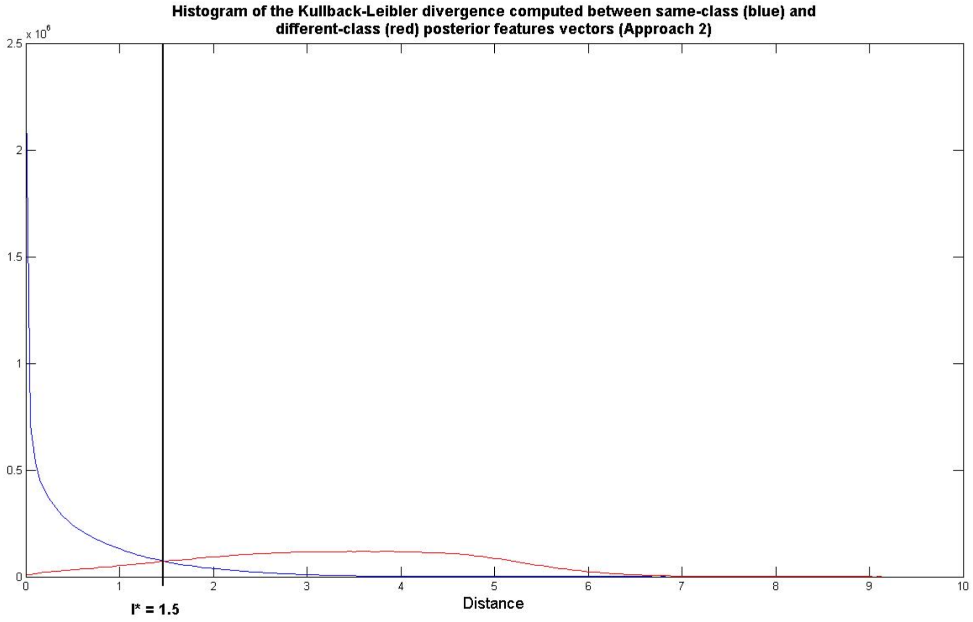

Figure 12 : Continuous approximation of the Kullback-Leibler same-class / different-class posterior pairs histogram

The classification accuracy obtained using the classification threshold $\mathbf{l}*$ is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	44.4 %	42.8 %
Different-class	44.0 %	42.6 %
Total	**88.4 %**	**85.4 %**

Table 9 : Classification accuracy for the posterior pairs training set and test set, when using Kullback-Leibler divergence

8.5.2.3. Bhattacharyya distance

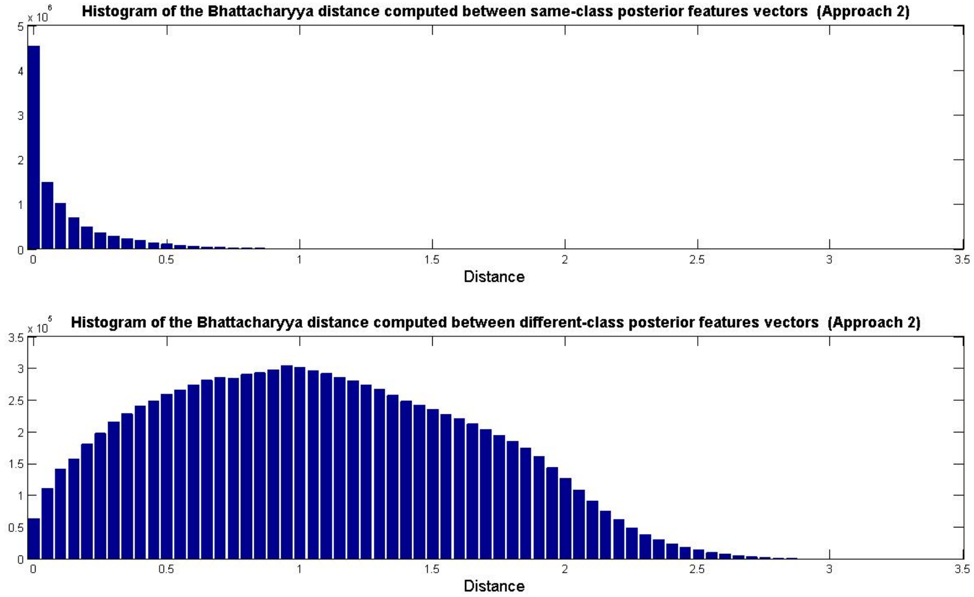

Figure 13 : Histograms of the Bhattacharyya divergence computed between same-class and different-class posterior feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	0.13	0.18
Different-class	1.10	0.58

Table 10 : Parameters of the Bhattacharrya same-class / different-class posterior pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in **l** * = 0.4.

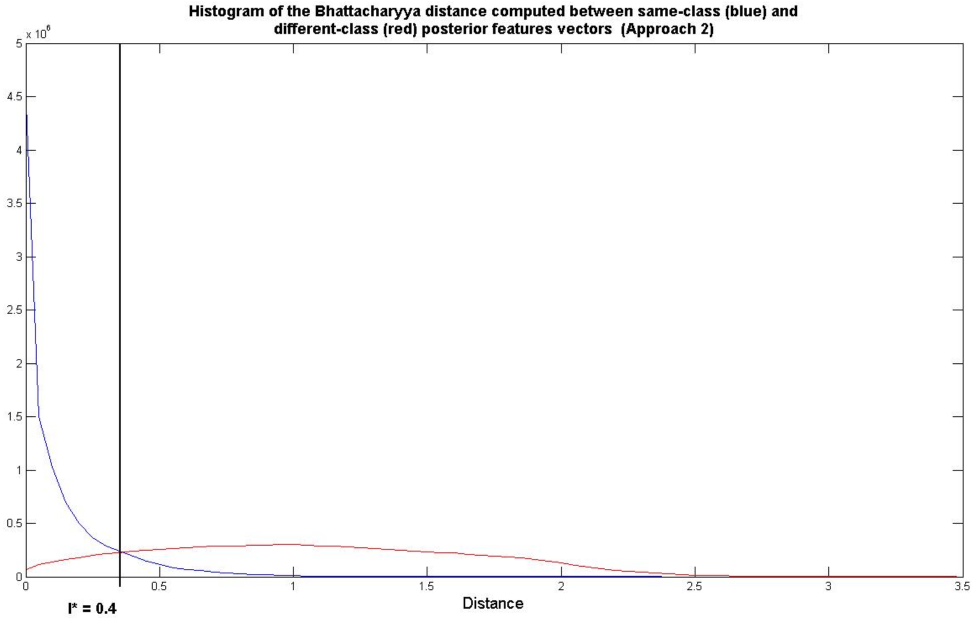

Figure 14 : Continuous approximation of the Bhattacharyya same-class / different-class posterior pairs histogram

The classification accuracy obtained using the classification threshold **l** * is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	44.9 %	43.7 %
Different-class	44.4 %	42.9 %
Total	**89.3 %**	**86.6 %**

Table 11 : Classification accuracy for the posterior pairs training set and test set, when using Bhattacharyya distance

8.5.2.4. MLP-based similarity measure

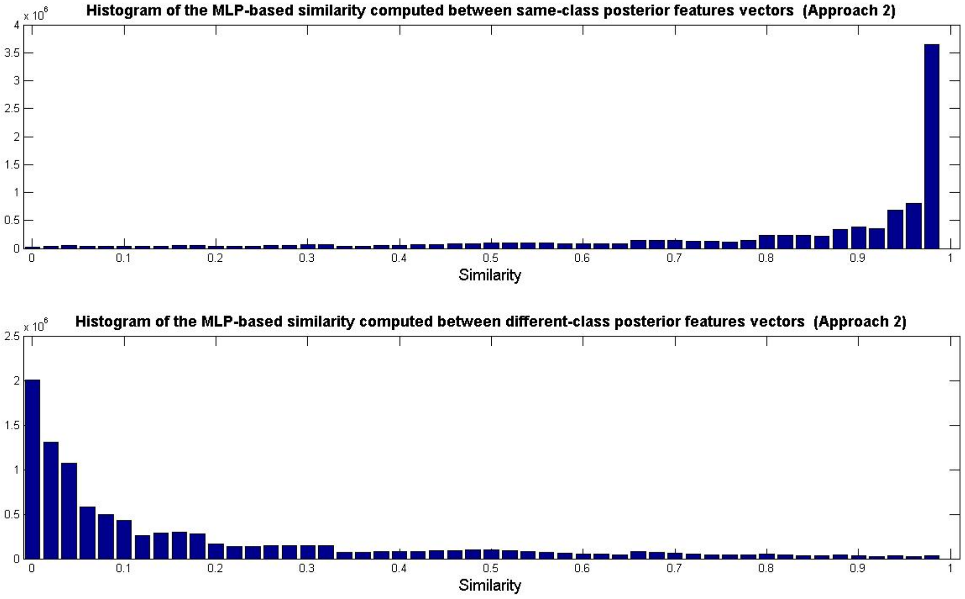

Figure 15 : Histograms of the MLP-based similarity computed between same-class and different-class posterior feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	0.82	0.24
Different-class	0.19	0.24

Table 12 : Parameters of the MLP-based same-class / different-class posterior pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in **l** * = 0.5.

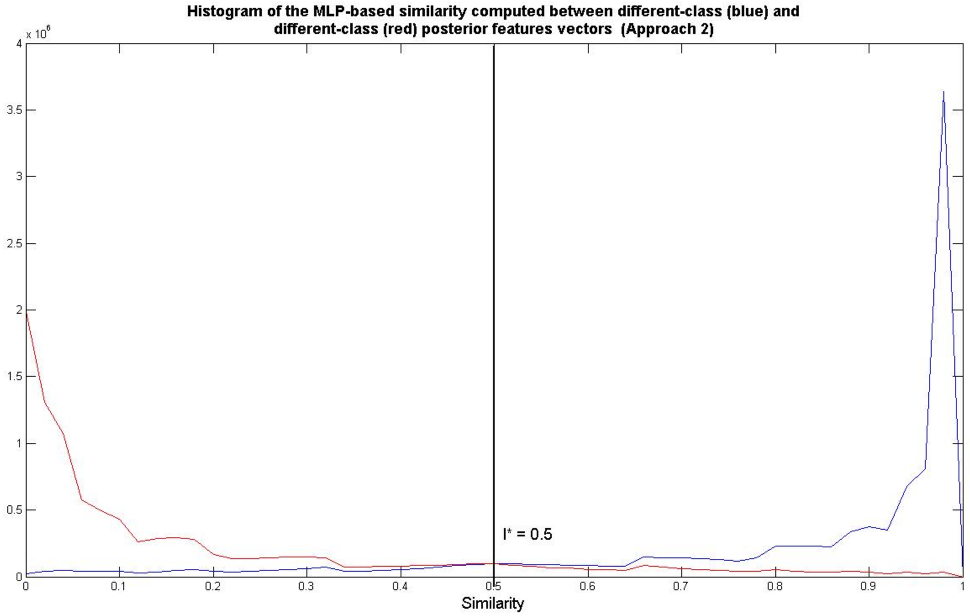

Figure 16 : Continuous approximation of the MLP-based same-class / different-class posterior pairs histogram

The classification accuracy obtained using the classification threshold **l** * is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	43.9 %	41.5 %
Different-class	43.4 %	42.8 %
Total	**87.3 %**	**84.3 %**

Table 13 : Classification accuracy for the posterior pairs training set and test set, when using MLP-based similarity

8.5.2.5. Scalar Product-based similarity measure

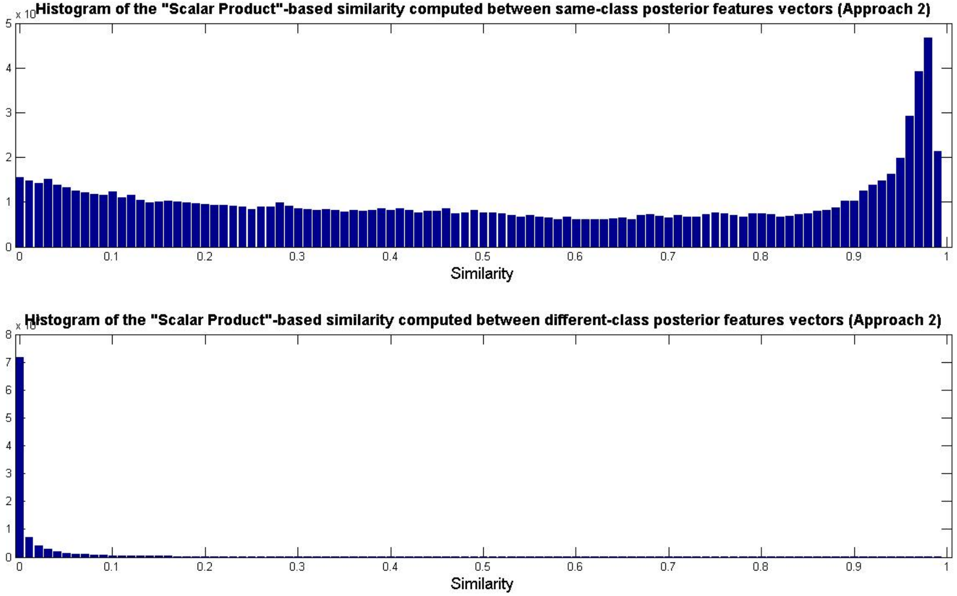

Figure 17 : Histograms of the "Scalar Product"-based similarity computed between same-class and different-class posterior feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	0.52	0.34
Different-class	0.03	0.08

Table 14 : Parameters of the "Scalar Product"-based same-class / different-class posterior pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in **l** * = 0.06.

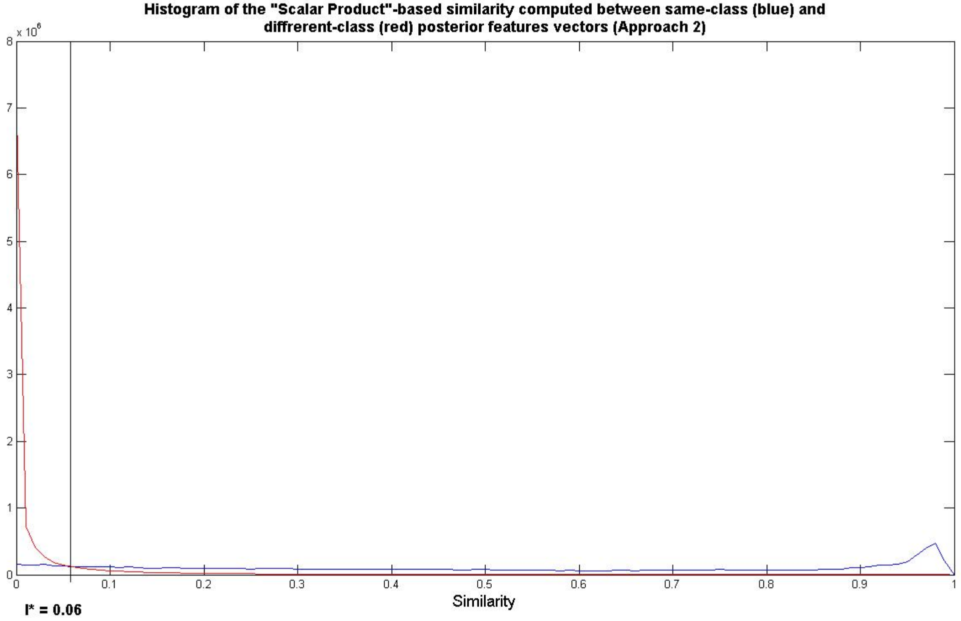

Figure 18 : Continuous approximation of the "Scalar Product"-based same-class / different-class posterior pairs histogram

The classification accuracy obtained using the classification threshold **l** * is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	45.9 %	45.0 %
Different-class	44.3 %	43.5 %
Total	**90.2 %**	**88.5 %**

Table 15 : Classification accuracy for the posterior pairs training set and test set, when using "Scalar Product"-based similarity

8.5.3. Using PLP feature vectors (Approach 2)

8.5.3.1. Euclidian distance

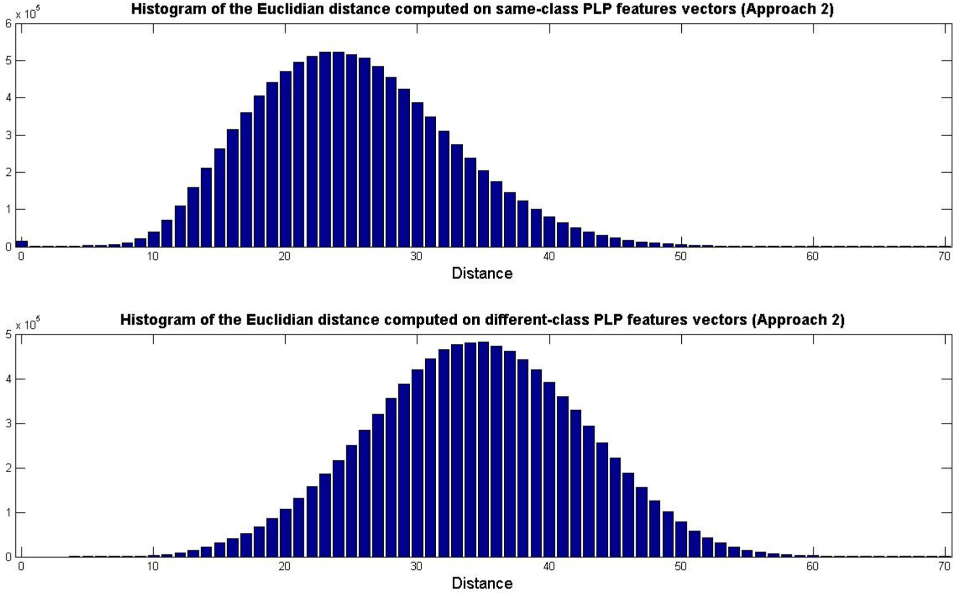

Figure 19 : Histograms of the Euclidian distance computed between same-class and different-class PLP feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	25.55	7.47
Different-class	34.90	8.04

Table 16 : Parameters of the Euclidian same-class / different-class PLP pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in **l** * = 29.53.

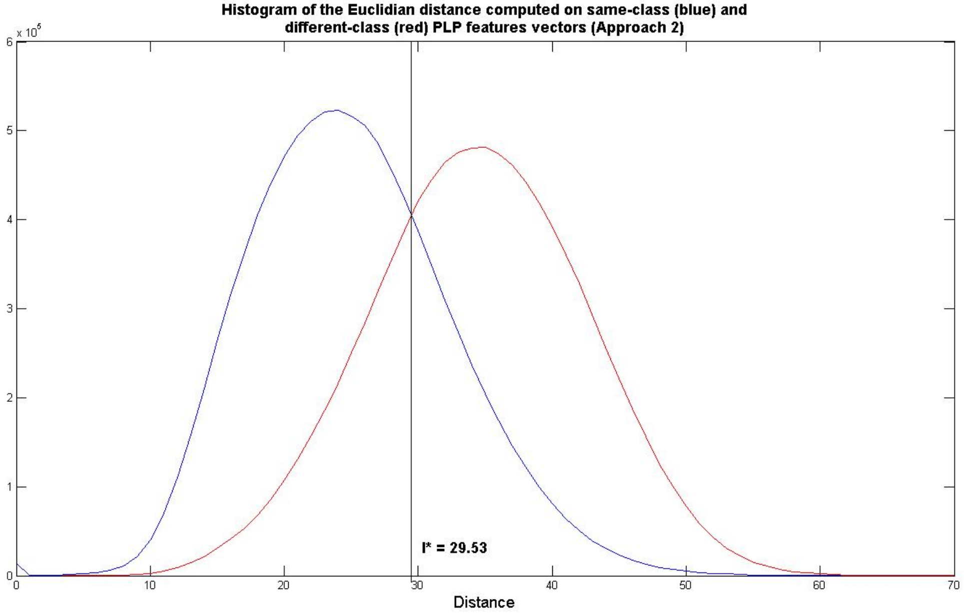

Figure 20 : Continuous approximation of the Euclidian same-class / different-class PLP pairs histogram

The classification accuracy obtained using the classification threshold **l** * is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	35.7 %	34.2 %
Different-class	37.3 %	36.8 %
Total	**73.0 %**	**71.0 %**

Table 17 : Classification accuracy for the PLP pairs training set and test set, when using Euclidian distance

8.5.3.2. Mahalanobis distance

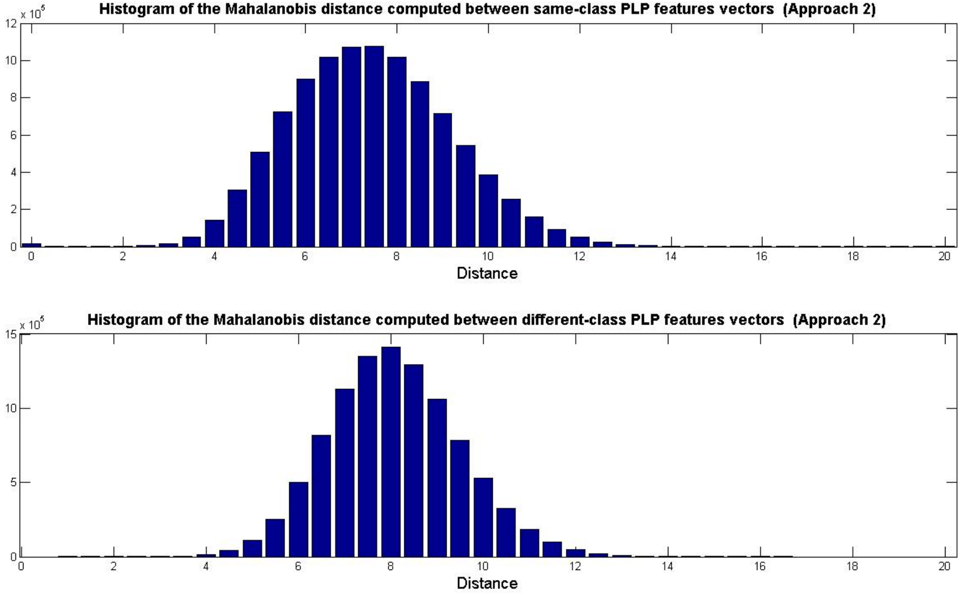

Figure 21 : Histograms of the Mahalanobis distance computed between same-class and different-class PLP feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	7.67	1.78
Different-class	8.33	1.42

Table 18 : Parameters of the Mahalanobis same-class / different-class PLP pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in $\mathbf{l}*$ = 6.89.

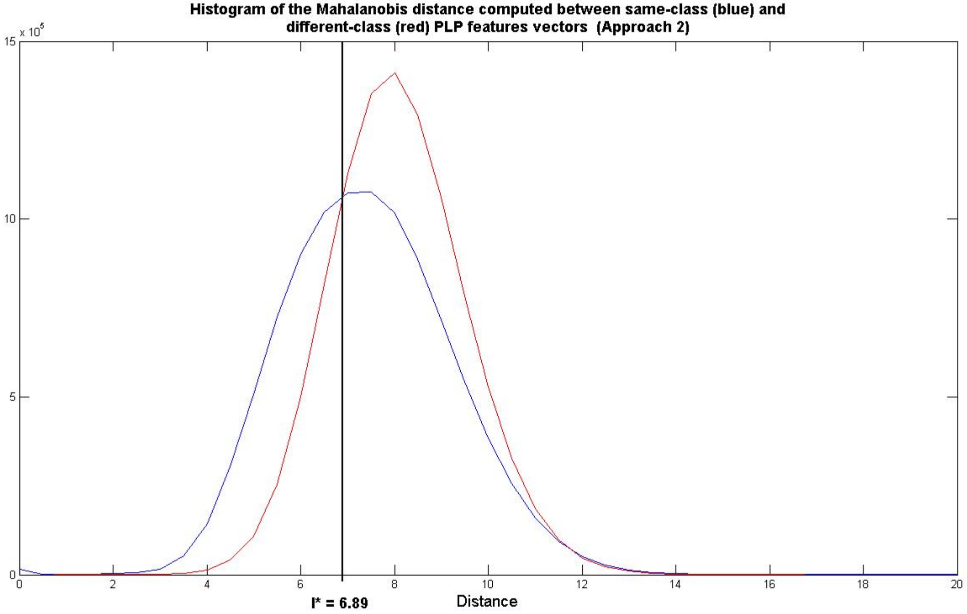

Figure 22 : Continuous approximation of the Mahalanobis same-class / different-class PLP pairs histogram

The classification accuracy obtained using the classification threshold $\mathbf{l}*$ is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	17.3 %	16.7 %
Different-class	42.3 %	43.5 %
Total	**59.6 %**	**60.2 %**

Table 19 : Classification accuracy for the PLP pairs training set and test set, when using Mahalanobis distance

8.5.3.3. MLP-based similarity measure

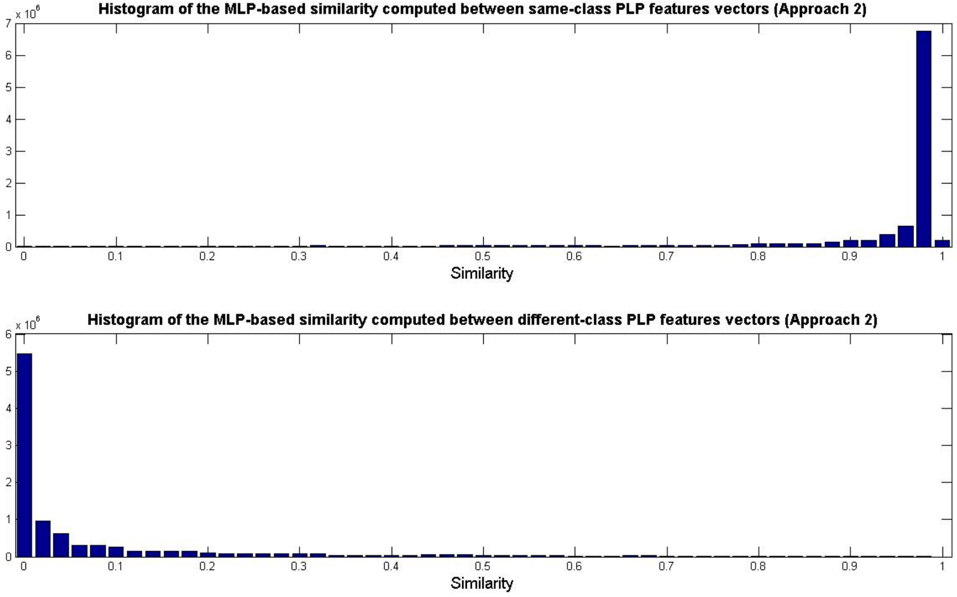

Figure 23 : Histograms of the MLP-based similarity computed between same-class and different-class PLP feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	0.93	0.18
Different-class	0.09	0.17

Table 20 : Parameters of the MLP-based same-class / different-class PLP pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in **l** * = 0.55.

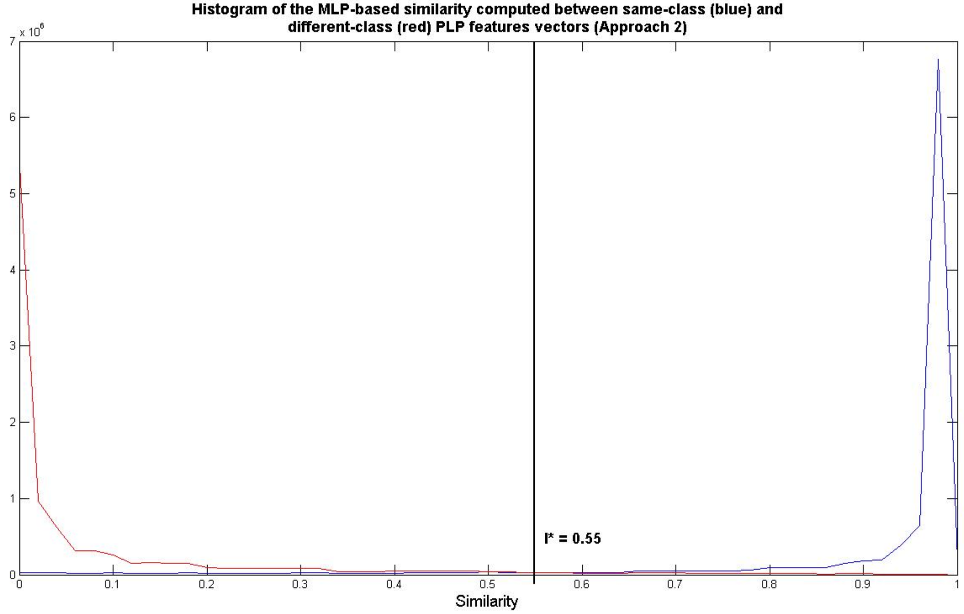

Figure 24 : Continuous approximation of the MLP-based same-class / different-class PLP pairs histogram

The classification accuracy obtained using the classification threshold **l** * is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	47.0 %	47.8 %
Different-class	48.0 %	22.7 %
Total	**95.0 %**	**70.5 %**

Table 21 : Classification accuracy for the PLP pairs training set and test set, when using MLP-based similarity

8.5.3.4. Scalar Product-based similarity measure

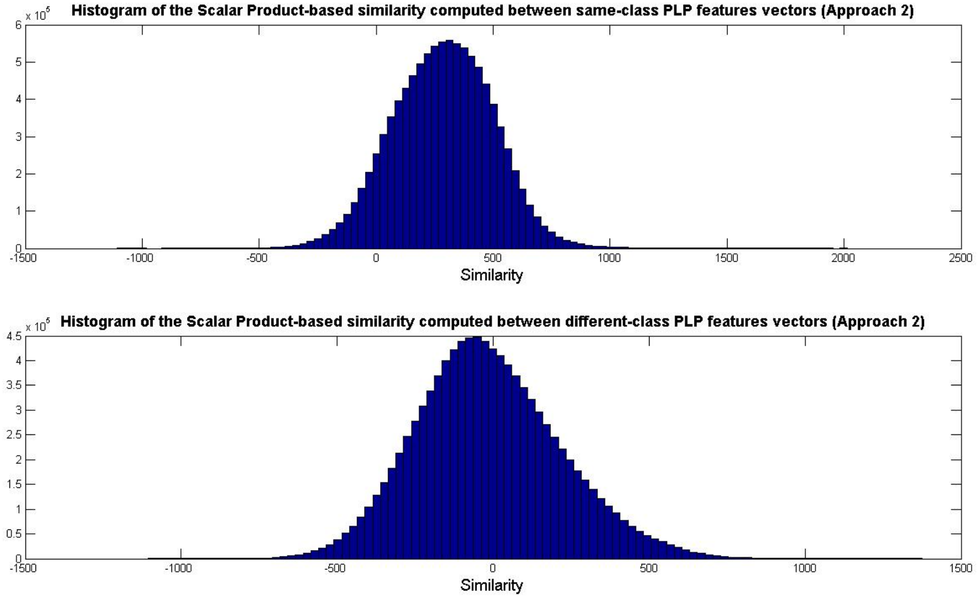

Figure 25 : Histograms of the "Scalar Product"-based similarity computed between same-class and different-class PLP feature vectors, using Approach 2

The parameters of these distributions are:

	Mean	Standard Deviation
Same-class	286.56	215.54
Different-class	-14.46	232.39

Table 22 : Parameters of the "Scalar Product"-based same-class / different-class PLP pairs distribution

The continuous approximation of the same-class and different-class histograms intersect in **l** * = 74.11.

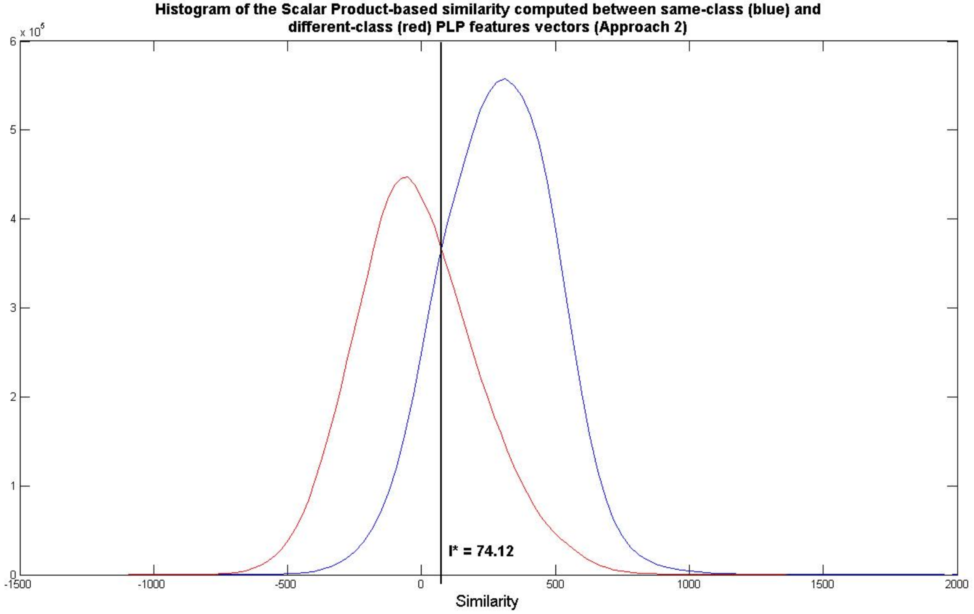

Figure 26 : Continuous approximation of the "Scalar Product"-based same-class / different-class PLP pairs histogram

The classification accuracy obtained using the classification threshold **l** * is given in the following table, for the training set and test set.

	Training Accuracy	Test Accuracy
Same-class	41.7 %	41.4 %
Different-class	33.5 %	33.2 %
Total	**75.2 %**	**74.6 %**

Table 23 : Classification accuracy for the PLP pairs training set and test set, when using "Scalar Product"-based similarity

8.6. Improved acoustic vectors classification

In this section, we investigate the performance of the kNN classifier using the MLP-based similarity and the Scalar Product metrics. The results are then compared with those obtained with more traditional distance measures (Euclidian, Mahalanobis, Bhattacharyya, Kullback-Leibler)[4].

8.6.1. kNN procedure

Using the training, cross-validation and test sets described in Table 1,

- For different values of *k*
    - For each vector in the cross-validation set
        - Compute the distance between the considered cross-validation vector and every training vectors, using the given metric
        - Select its *k* nearest-neighbors among every training vectors
        - Assign to the cross-validation vector the most represented label within these *k* nearest-neighbors
    - Compute the classification accuracy
- Select the value of *k* leading to the best classification accuracy
- Given the “optimum” k value estimated in the previous step, repeat the procedure on the test set:
    - For each vector in the test set
        - Compute the distance between the considered test vector and every training vectors, using the given metric
        - Select its *k* nearest-neighbors among every training vectors
        - Assign to the test vector the most represented label within these *k* nearest-neighbors
    - Compute the classification accuracy

8.6.2. kNN classification using the MLP-similarity (MLP-s)

In this section, the kNN classification rule is applied to posterior and PLP feature vectors.

The MLP-s is trained using the balanced training and cross-validation pairs set defined in Section 8.4. Table 24 shows the best training and cross-validation accuracies obtained during MLP-s training (for conciseness’s sake, only the final results obtained at the end of the training phase are given).

The optimum *k* value has been estimated using 2,000 cross-validation vectors, using the 920,166 training vectors as the prototypes set.

[4] Thank you to Mrs Afsaneh Asaei for contributing to this part of the work.

As the hypothesis tests experiments confirmed that Approach 2 achieves better results than Approach 1 (Section 8.5), results in Table 24 are given for Approach 2.

Type of feature vectors	Number of hidden units	Best training accuracy	Best cross-validation accuracy	Optimum k value	Classification accuracy
Posterior	1,000	87.25 %	84.27 %	20,000	46.8 %
PLP	1,000	96.06 %	55.02 %	150,000	36.9 %

Table 24 : kNN classification results when using MLP-s based similarity measure for Approach 2, for posterior and PLP feature vectors

These bad classification results can be explained as follows:

- The MLP-s used for PLP feature vectors seems to be over-trained because of the large gap between the best training and cross-validation accuracies. An MLP with reduced number of hidden units should probably be better
- Being computational resources consuming, the classification accuracies were obtained from a reduced set of vectors (2,000 frames), which is probably insufficient

For the PLP features, the 49.9% classification accuracy provided by the kNN based on the Euclidian distance also was not very good.

It should be noticed that Table 24 gives the kNN classification accuracy obtained for the cross-validation set used for tuning the value of k. The kNN classification accuracy estimation being time consuming when using the MLP similarity, it was not possible to perform the second part of the procedure (classification accuracy of the test set) in the frame of this work. However, this should not provide better results. More experiments should be first conducted to find the optimum number of hidden units.

N.B.: in our first experiments, we attempted to train the MLP-s with a set of extremely unbalanced pairs containing 97.5% of different-class pairs. As a result, the MLP-s learned the priors of the distribution (the best training and cross-validation accuracies were equal to 97.5%) and classified all the training pairs as belonging to the “different-class”. The only way to avoid such a trivial solution is to use a well balanced training set, as explained in Section 5.

8.6.3. kNN classification using Scalar Product metric

We used the set of 920,166 training vectors as the prototypes set. The optimum *k* value has been estimated using 204,657 cross-validation vectors. The kNN classification accuracy has been assessed on a set of 410,920 test vectors, using this optimum *k* value.

8.6.3.1. kNN on PLP feature vectors

The kNN based on the scalar product of the PLP vectors did not provide very good results with the PLP features, yielding a classification accuracy limited to 38.3 % (optimum *k* = 750), to be compared with the 49.9% obtained with the Euclidian distance. In Section 4, we defined the scalar product similarity between two feature vectors as being the scalar product of their associated posterior vectors, not of the acoustic vectors themselves.

We also showed that this similarity measure was an estimation of the probability that the two feature vectors are part of the same phonetic class. No similar interpretation was given to the scalar product of two acoustic vectors (MFCC or PLP for instance), whose meaning (if any) has still to be analyzed. This has to be taken into account when interpreting the results obtained with the kNN based on the scalar product of the PLP vectors.

8.6.3.2. kNN on Posterior feature vectors

In this case, we found *k* optimum equal to 5 and we obtained a kNN classification accuracy of 68.3% (assessed on the test set). This result may be compared to the MAP classification accuracy (69.6% on the test set).

Figure 27 shows the evolution of the cross-validation set classification accuracy versus *k*. We see that the maximum classification accuracy over this set, equal to 70.8%, is obtained for *k* = 5. The test set classification accuracy, equal to 68.3%, was computed using this optimum value of *k*.

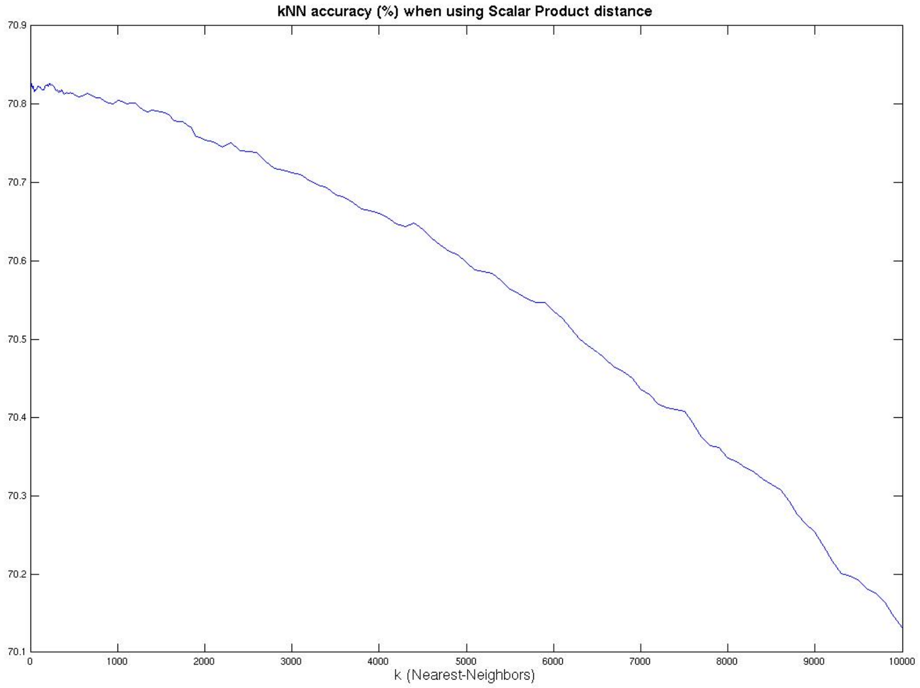

Figure 27 : Evolution of the cross-validation set classification accuracy versus *k*

Figure 28 shows the confusion matrix for the test data. The confusion matrix is a three-dimensional graph, representing the number of time each phoneme label is predicted by the system, for each actual phoneme label. In a perfect confusion matrix the peaks should be exclusively located on the diagonal (meaning that the system is always predicting the correct phoneme label, with a 100% recognition rate). The confusion matrix of our classifier clearly shows its very good results, the higher peaks being located on the diagonal. A similar matrix was obtained for the cross-validation data.

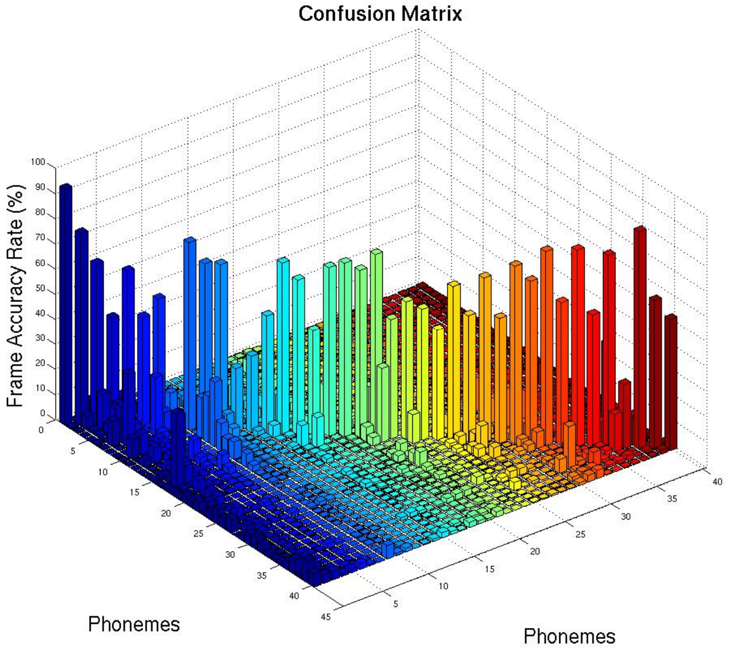

Figure 28 : Confusion matrix for the test set when using kNN with Scalar Product metric

8.6.4. Summary and conclusions

Table 25 shows the results of the kNN classification on posterior and PLP feature vectors, for different types of distances.

We clearly see that:

- Posterior feature vectors achieve better results than PLP feature vectors
- Scalar Product metric achieves comparable results to the other, more traditional, distance metrics

However, we still believe that MLP-s has also the potential to achieve good results on posterior and acoustic vectors. More research and experiments are needed to truly assess this potential.

k-NN (without smoothing) - Vectors Classification Accuracy					
Distance **Test vectors set**	**Euclidian**	**Kullback-Leiber**	**Bhattacharyya**	**Scalar Product**	**MLP 1000 hidden units**
Posterior vectors					
Test vectors	68.3%	68.5%	68.2%	68.3%	46.8%
k optimum	260	200	20	5	20000
PLP vectors					
Test vectors	49.9%	▨	▨	38.3%	36.9%
k optimum	70	▨	▨	750	150000

Table 25 : kNN accuracy when using different types of distances between posterior and PLP feature vectors

Note 1: Results for Euclidian, Kullback-Leiber and Bhattacharyya metrics are Mrs Afsaneh Asaaei's contribution.
Note 2: Classification accuracy estimated on a set of 410,920 test vectors (on 2,000 cross-validation vectors for MLP similarity).
Note 3: Test performed using the 920,000 training vectors as the prototypes set.
Note 4: Tuning ("*k* optimum" evaluation) performed on cross-validation vectors.
Note 5: MLP training preformed with randomized pairs, using Approach 2.
Note 6: Kullback-Leiber & Bhattacharyya not applicable with PLP vectors.
Note 7: Mahalanobis not applicable to posterior vectors (matrix non invertible).
Note 8: MAP rule gives 69.6 % of accuracy on the test vectors set.

9. Going further

As announced by the title, the present document clearly demonstrated the potential in using posterior features together with k-NN classifiers towards improved ASR system. However, it is also clear that as part of a Master project, we only had time to scratch the surface of a very exciting research direction, where we can foresee multiple avenues for further investigations. We briefly discuss below just a very few of those.

As discussed in the present document, we have to carefully select the training vector pairs, given the excessive amounts of training data (pairs) and its unbalanced priors (towards “different class”). Combining hypothesis testing and Support Vector Machine (SVM) principles ([5], [14]) could be considered for selecting the vectors pairs used for MLP similarity training. The idea is to make the MLP learn preferably near the class decision boundary, by selecting the pairs of training vectors located “near” the intersection of the two histograms $P(\mathbf{l}|\Omega_s)$ and $P(\mathbf{l}|\Omega_d)$. A parameter to be tuned in this case is the distance range z of the selected training pairs, around the threshold (as illustrated in Figure 29).

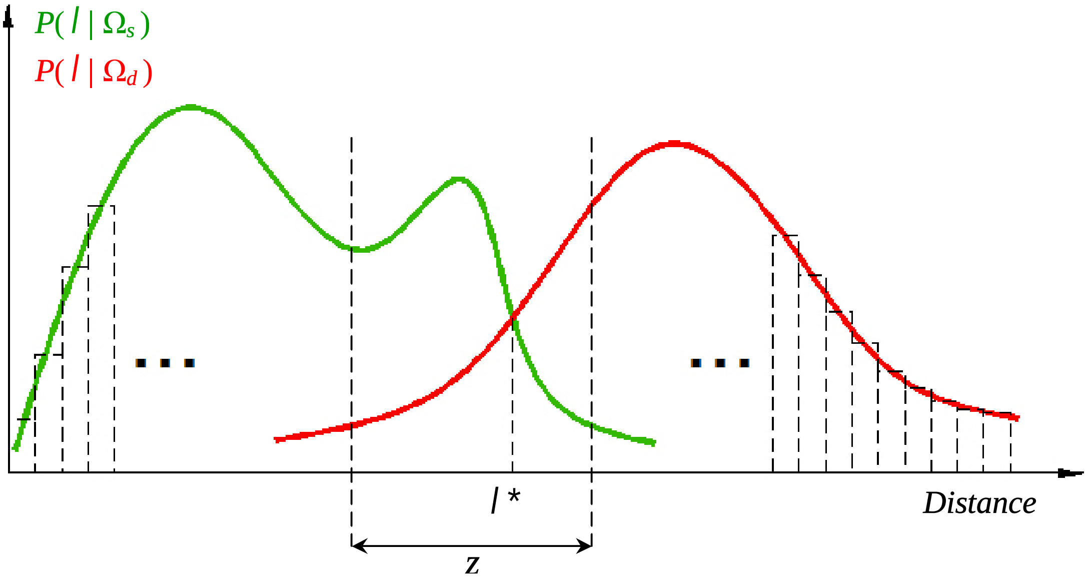

Figure 29 : Training the MLP-s by selecting pairs of vectors near the class decision boundary

As used now the MLP-s is not guaranteed to generate a symmetric distance metric, since it probably depends on the ordering of the each input vector pair. It is, however, possible to enforce symmetry by using $|x_n - y_n|$ and $|x_n + y_n|$ as input pairs instead of instead of x_n and y_n. This approach could perhaps lead to a performance improvement.

Other promising avenues of investigation also include:

- Hypothesis test on variable length speech units. In this work, we have considered the distance between two single feature vectors. However, speech units (e.g. phones, words,…) are represented by sequences of feature vectors. A constant length sequence (independent of the speech unit class and instance), could be considered as a bigger feature vector, and the method used in this work could still be applied. It is not the case in practice, and we have to take into account the variability of the speech unit length. This can by integrating the hypothesis test into Dynamic Time Warping (DTW) process, where the DTW local distance would be defined as the probability that a pair of (test, reference) vectors belongs to the same class or not. In a more extreme case, we could also consider DTW where local distances would simply be binary values (know that about 90% of the time we would get the correct "1" along the optimal path).

- The generalization of previous works on KL-HMM [2] to PSP-HMM (Posterior-Scalar-Product-HMM)

- The use of (temporal) contextual information when classifying posterior vectors. Indeed, in the present work we only looked at 10-ms posterior features, although it is known that looking at larger context (typically 90ms) will help classification.

10. Conclusion

In this master’s thesis, we have considered the speech recognition problem under a different angle, simply formulating it in terms of hypothesis testing. We also provide a first answer to the question: "Given two feature vectors, what is the probability that these belong to the same (phonetic) class or not?". While theoretically proving that the MLP-s trained as we did (feature vectors at the input and desired output is 1 or 0 depending on whether the two input vectors belong to the same class or not) should work properly, we discovered a very simple and interesting property: the “optimal” output of the MLP-s is simply an estimate of the scalar product of the 2 (actual) posterior vectors associated to the 2 input feature vectors. Therefore we decided to introduce a new kind of similarity between feature vectors, the scalar product of their associated posterior vectors, and compared it to the other metrics (Euclidian, Mahalanobis, Kullback-Leibler, Bhattacharyya, MLP-based).

The main conclusions from this part of the work can then be summarized as follows:

- Posterior feature vectors always achieve better results than PLP feature vectors. This result was expected because posterior vectors are more robust, i.e. speaker and environment independent (hence capturing more of the phonetic information contained in the signal)
- The Scalar Product similarity achieves better performance than all other metrics (including the MLP-based one)

Moreover, we have also investigated the possible use of k-NN classifiers to perform frame-based acoustic phonetic classification, resulting in the following conclusions:

- Posterior feature vectors achieve (again) better results than PLP vectors
- Scalar Product achieves comparable results to the other types of distance metrics

Personal Conclusion:

This Master’s thesis has been a great opportunity to collaborate with researchers from various countries and to contribute to the cutting edge field of speech processing. It was also a tremendous experience abroad, in an international scientific and cultural context which makes people grow both personally and professionally.

Bibliography

[1] Abdou, S. and Scordilis, M.S., “Beam search pruning in speech recognition using a posterior-based confidence measure”, Speech Communication, Vol. 42, pp. 409-428, 2004.

[2] Aradilla Zapata Guillermo, “Acoustic Models for Posterior Features in Speech Recognition”, PhD thesis at Ecole Polytechnique Fédérale de Lausanne and Idiap Research Institute, 2008.

[3] Aradilla, G., Vepa, J., Bourlard, H., “An Acoustic Model Based on Kullback-Leibler Divergence for Posterior Features”, ICASSP 2007

[4] Bernardis, G. and Bourlard, H., “Improving posterior confidence measures in hybrid HMM/ANN speech recognition system”, Proceedings of the Intl. Conference on Spoken Language Processing (Sydney, Australia), pp. 775-778, 1998.

[5] Bishop Christopher M., “Pattern Recognition and Machine Learning”, Springer, 2006.

[6] Boite René, Bourlard Hervé, Dutoit Thierry, Hancq Joël, Leich Henri, “Traitement de la Parole”, Presse Polytechniques et Universitaires Romandes, 1999.

[7] Bourlard Hervé & Morgan Nelson, “Connectionist Speech Recognition – A Hybrid Approach”, Kluwer Academic Publishers, 1994.

[8] Bourlard Hervé & Wellekens Christian, “Links between Markov Models and Multilayer Perceptrons”, D.S. Touretzki editor, Advances in Neural Information Processing Systems, volume 1, pages 502-510, San Mateo, CA, 1989. IEEE, Morgan Kaufmann.

[9] Bourlard Hervé, “Reconnaissance de la Parole et du Locuteur”, Course Notes.

[10] Bourlard Hervé, Magimai Doss Mathew, Nelson Morgan, Mesot Bertrand, Bengio Samy, Zhu Qifeng, “Towards using Hierarchical Posteriors for Flexible Automatic Speech Recognition Systems”, Idiap Research Report, November 2008

[11] Chen, B., Zhu, Q., and Morgan, N., “Learning long-term temporal features in LVCSR using neural networks”, Proc. Interspeech’04 (Korea), October 2004

[12] Cover, T.M., Hart, P.E., “Nearest Neighbor Pattern Classification”, lEEE Trans. Information Theory, vol. 13, no. 1, pp. 21-27, Jan. 1967

[13] Devijver, P. A. and Kitler, J., “Pattern Recognition: A Statistical Pattern Approach”, Prentice/Hall International, 1982

[14] Duda R.O., Hart P.E., Stork D.G., “Pattern Classification”, Wiley-Interscience, second edition, 2001.

[15] Dutoit T. and Marquès F., “Applied Signal Processing - A MATLAB-based Proof of Concept”, Chapter “How does a dictation machine recognize speech”, Springer: Boston, 2008.

[16] Elkan, C., “ Results of the KDD ’99 Classifier Learning Contest,” This paper presents a methodology, neighborhood counting, Sept. 1999, http://www.cs.ucsd.edu/users/elkan/clresults.html

[17] Fix E. and Hodges Jr., J. L., “Discriminatory Analysis: Non- parametric Discrimination: Consistency Properties,” Report No. 4, Data set USAF School of Aviation Medicine, Randolph Field, Texas, Feb. 1951

[18] Fix E. and Hodges Jr., J. L., “Discriminatory Analysis: Non-parametric Discrimination: Small Sample performance”, Report No. 11, USAF School of Aviation Medicine, Randolph Field, Texas, Aug. 1952

[19] Fukunaga K., “Statistical Pattern Recognition by Statistical Recognition”, Academic Press, 1990

[20] Furui S., “Speaker-Independent Isolated word Recognition Using Dynamic Features of Speech Spectrum”, IEEE Transactions on Acoustics, Speech, and Signal Processing, volume 34, pages 52-59, 1986.

[21] Garcia Moral Ana I., Pelaez Moreno Carmen, Hervé Bourlard, “On the Use of MLP-Distance to Estimate Posterior Probabilities by kNN for Speech Recognition”, Jornadas en Tecnologia del Habla, Zaragoza, Noviembre de 2006.

[22] Ghosh, A. K., Chaudhuri, P., and Murthy , C.A., “On Visualization and Aggregation of Nearest Neighbor Classifiers”, IEEE Trans. on Pattern Analysis and Machine Intelligence”, vol. 27, no. 10, Oct. 2005

[23] Gosselin Bernard, “Classification et Reconnaissance Statistique de Formes”, Course Notes, Faculté Polytechnique de Mons, 2000.

[24] Hayashi, H., Sese, J. and Morishita S., “Optimization of Nearest Neighborhood Parameters for KDD-2001 Cup ‘the Genomics Challenge’,” technical report, Univ. of Tokyo, 2001, http://www.tsujii.is.s.u-tokyo.ac.jp/GENIA/WS/PDFfiles/Morishita.pdf

[25] Hermansky, H., Ellis, D., Sharma, S., “Tandem Connectionist Feature Extraction for Conventional HMM Systems”, Proceedings of the ICASSP, 2000

[26] Hermansky, H. and Sharma S., “TRAPS Classifiers of Temporal Patterns”, Proceedings of Intl. Conf. on Spoken Language Processing (Sydney, Australia), 1998.

[27] Hermansky, H., Ellis, D.P.W., and Sharma, S., “Connectionist Feature Extraction for Conventional HMM Systems”, Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (Istanbul, Turkey), 2000.

[28] ICSI Quicknet V3.20 tool: http://www.icsi.berkeley.edu/Speech/icsi-speech-tools.html

[29] Ikbal, S., “Non-linear Feature Transformations for Noise Robust Speech Recognition”, Ph.D. Thesis, Ecole Polytechnique Federal de Lausanne, 2004

[30] Ikbal, S., Misra, H., Sivadas, S., Hermansky, H., and Bourlard, H., “Entropy Based Combination of Tandem Representations for Robust Speech Recognition”, Proc. Interspeech’04 (Korea), October 2004

[31] Lee Kai-Fu & Hon Hsiao-Wuen, “Speaker-Independent Phone Recognition Using Hidden Markov Models”, IEEE Transactions on Acoustics, Speech, and Signal Processing, volume 37, no. 11, pages 1641–1648, 1989.

[32] Mangu, L., Brill, E., and Stolcke, A., “Finding consensus in speech recognition: word error minimization and other applications of confusion networks”, Computer, Speech and Language, Vol. 14, pp. 373-400, 2000.

[33] Pinto Joël, Magimai Doss Mathew, Hermansky Hynek, Yegnanarayana B., “Exploiting Contextual Information for Improved Phoneme Recognition”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008.

[34] Rabiner Lawrence, Juang Biing-Hwang, “Fundamentals of Speech Recognition”, Prentice Hall Signal Processing Series, Pearson Education, 1993.

[35] TIMIT Database: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1

[36] Wellekens Christian, “Traitement de la Parole et du Son”, Institut Eurecom Sophia Antipolis, 2008.

[37] Yang, H., van Vuuren, S., and Hermansky, H. “Relevancy of Time-frequency Features for Phonetic Classification of Phonemes”, Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 1999), 1, 225–229.

[38] Zhu, Q., Chen, B., Morgan, N., and Stolcke, A., “On using MLP features in LVCSR”, Proc. Interspeech’04 (Korea), October 2004

[39] http://matlabdatamining.blogspot.com/2006/11/mahalanobis-distance.html

APPENDIXES

Appendix 1 – TIMIT Database

The whole TIMIT database [35] contains a total of 6,300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16 kHz speech waveform file for each utterance. Table 26 shows the number of speakers for the 8 dialect regions, broken down by sex. The percentages are given in parentheses. A speaker's dialect region is the geographical area of the U.S. where they lived during their childhood years. The geographical areas correspond with recognized dialect regions in U.S. (Language Files, Ohio State University Linguistics Dept., 1982), with the exception of the Western region in which dialect boundaries are not known with any confidence and dialect region 8 where the speakers moved around a lot during their childhood.

Dialect Region	Number of male	Number of female	Total
New England	31 (63%)	18 (27%)	49 (8%)
Northern	71 (70%)	31 (30%)	102 (16%)
North Midland	79 (67%)	23 (23%)	102 (16%)
South Midland	69 (69%)	31 (31%)	100 (16%)
Southern	62 (63%)	36 (37%)	98 (16%)
New York City	30 (65%)	16 (35%)	46 (7%)
Western	74 (74%)	26 (26%)	100 (16%)
Army Brat (moved around)	22 (67%)	11 (33%)	33 (5%)
Total	**438 (70%)**	**192 (30%)**	**630 (100%)**

Table 26 : Number of speaker for the 8 dialect regions, broken down by sex

The text material in the TIMIT prompts consists of 2 dialect "shibboleth" sentences designed at Stanford Research Institute (SRI), 450 phonetically-compact sentences designed at Massachusetts Institute of Technology (MIT), and 1890 phonetically-diverse sentences selected at Texas Instruments (TI). The dialect sentences were meant to expose the dialectal variants of the speakers and were read by all 630 speakers. The phonetically-compact sentences were designed to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts thought to be either difficult or of particular interest. Each speaker read 5 of these sentences and each text was spoken by 7 different speakers. The phonetically-diverse sentences were selected from existing text sources - the Brown Corpus (Kuchera and Francis, 1967) and the Playwrights Dialog (Hultzen, et al., 1964) - so as to add diversity in sentence types and phonetic contexts. The selection criteria maximized the variety of allophonic contexts found in the texts. Each speaker read 3 of these sentences, with each sentence being read only by a single speaker.

Appendix 2 – Number of phoneme utterances & pairs selection

1. *Training data*

Number of pairs *M* =	20,000,000
s_k / $p(\omega_k)$ =	14,562
d_k / $p(\omega_k)$ =	3,240

					Number of phonemes for:	
Phoneme	**Phoneme label**	**Total number of phoneme utterances**	**Phoneme prior $P(\omega)$ (%)**	**$p^2(\omega_k)$ (%)**	**"Same class" pairs s_k**	**"Different class" pairs d_k**
sil	0	127,143	13.83%	1.91%	2,014	448
iy	1	35,629	3.88%	0.15%	564	126
ih	2	57,637	6.27%	0.39%	913	203
eh	3	25,344	2.76%	0.08%	402	89
ae	4	25,690	2.79%	0.08%	407	91
ah	5	31,624	3.44%	0.12%	501	111
uw	6	16,483	1.79%	0.03%	261	58
uh	7	3,106	0.34%	0.00%	49	11
ao	8	41,023	4.46%	0.20%	650	145
ey	9	23,333	2.54%	0.06%	370	82
ay	10	24,243	2.64%	0.07%	384	85
oy	11	4,052	0.44%	0.00%	64	14
aw	12	9,699	1.06%	0.01%	154	34
ow	13	17,479	1.90%	0.04%	277	62
l	14	29,292	3.19%	0.10%	464	103
r	15	21,532	2.34%	0.05%	341	76
y	16	4,572	0.50%	0.00%	72	16
w	17	11,023	1.20%	0.01%	175	39
er	18	31,910	3.47%	0.12%	506	112
m	19	19,039	2.07%	0.04%	302	67
n	20	33,611	3.66%	0.13%	532	118
ng	21	6,225	0.68%	0.00%	99	22
ch	22	5,888	0.64%	0.00%	93	21
jh	23	5,163	0.56%	0.00%	82	18
dh	24	7,000	0.76%	0.01%	111	25
b	25	12,691	1.38%	0.02%	201	45
d	26	21,240	2.31%	0.05%	336	75
dx	27	4,326	0.47%	0.00%	69	15
g	28	8,428	0.92%	0.01%	134	30
p	29	24,586	2.67%	0.07%	390	87
t	30	40,107	4.36%	0.19%	635	141
k	31	36,414	3.96%	0.16%	577	128
z	32	25,096	2.73%	0.07%	398	88
sh	33	13,836	1.51%	0.02%	219	49
v	34	9,785	1.06%	0.01%	155	34
f	35	18,451	2.01%	0.04%	292	65
th	36	5,511	0.60%	0.00%	87	19
s	37	57,762	6.28%	0.39%	915	204
hh	38	9,072	0.99%	0.01%	144	32
oth	39	14,117	1.54%	0.02%	224	50
Total :		**919,162**	**100.00%**	**4.72%**	**14,563**	**3,238**

2. Cross-validation data

Number of pairs M =	4,000,000
s_k / $p(\omega_k)$ =	6,432
d_k / $p(\omega_k)$ =	1,450

Phoneme	Phoneme label	Total number of phoneme utterances	Phoneme prior $P(\omega)$ (%)	$p^2(\omega_k)$ (%)	Number of phonemes for:	
					"Same class" pairs s_k	"Different class" pairs d_k
sil	0	29,231	14.29%	2.04%	919	207
iy	1	8,346	4.08%	0.17%	262	59
ih	2	12,851	6.28%	0.39%	404	91
eh	3	5,142	2.51%	0.06%	162	36
ae	4	5,347	2.61%	0.07%	168	38
ah	5	6,497	3.18%	0.10%	204	46
uw	6	3,041	1.49%	0.02%	96	22
uh	7	667	0.33%	0.00%	21	5
ao	8	9,591	4.69%	0.22%	302	68
ey	9	5,296	2.59%	0.07%	167	38
ay	10	5,610	2.74%	0.08%	176	40
oy	11	1,058	0.52%	0.00%	33	7
aw	12	1,995	0.98%	0.01%	63	14
ow	13	3,622	1.77%	0.03%	114	26
l	14	6,266	3.06%	0.09%	197	44
r	15	4,793	2.34%	0.05%	151	34
y	16	864	0.42%	0.00%	27	6
w	17	2,390	1.17%	0.01%	75	17
er	18	7,101	3.47%	0.12%	223	50
m	19	4,255	2.08%	0.04%	134	30
n	20	7,545	3.69%	0.14%	237	53
ng	21	1,243	0.61%	0.00%	39	9
ch	22	1,180	0.58%	0.00%	37	8
jh	23	1,047	0.51%	0.00%	33	7
dh	24	1,545	0.76%	0.01%	49	11
b	25	3,271	1.60%	0.03%	103	23
d	26	4,829	2.36%	0.06%	152	34
dx	27	1,087	0.53%	0.00%	34	8
g	28	1,926	0.94%	0.01%	61	14
p	29	5,417	2.65%	0.07%	170	38
t	30	9,304	4.55%	0.21%	293	66
k	31	8,343	4.08%	0.17%	262	59
z	32	5,841	2.86%	0.08%	184	41
sh	33	2,964	1.45%	0.02%	93	21
v	34	2,218	1.08%	0.01%	70	16
f	35	4,452	2.18%	0.05%	140	32
th	36	1,358	0.66%	0.00%	43	10
s	37	12,298	6.01%	0.36%	387	87
hh	38	2,106	1.03%	0.01%	66	15
oth	39	2,612	1.28%	0.02%	82	19
Total :		**204,549**	**100.00%**	**4.83%**	**6,433**	**1,449**

Appendix 3 – HTK configuration file

```
MAXTRYOPEN=3
SOURCEKIND = WAVEFORM               -->   Parameter kind of source
SOURCEFORMAT = NIST                 -->   File format of source
TARGETFORMAT = HTK                  -->   File format of target
TARGETKIND = PLP_D_A_K_Z_0          -->   Parameter kind of target
TARGETRATE = 100000.0               -->   Sample period of target in 100ns
units
HPARM: SAVECOMPRESSED = T           -->   Save the output file in compressed
form
HPARM: SAVEWITHCRC = T              -->   Attach a checksum to output
parameter file
HPARM: ZMEANSOURCE = T              -->   Zero mean source waveform before
analysis
HPARM: WINDOWSIZE = 250000.0        -->   Analysis window size in 100ns
units
HPARM: USEHAMMING = T               -->   Use a Hamming window
HPARM: PREEMCOEF = 0.97             -->   Set pre-emphasis coefficient
HPARM: NUMCHANS = 24                -->   Number of filterbank channels
HPARM: LPCORDER = 12                -->   Order of the LPC analysis
HPARM: COMPRESSFACT = 0.3333333
HPARM: NUMCEPS = 12                 -->   Number of cepstral parameters per
vector
HPARM: CEPLIFTER = 22               -->   Cepstral liftering coefficient
HPARM: ESCALE = 1.0                 -->   Scale log energy
HPARM: ENORMALISE = T               -->   Normalized log energy
HPARM: SILFLOOR = 50.0              -->   Energy silence floor
HPARM: USEPOWER = T                 -->   Use power not magnitude in fbank
analysis
HPARM: CEPSCALE = 10
```