# LOW-RANK REPRESENTATION FOR ENHANCED DEEP NEURAL NETWORK ACOUSTIC MODELS

Gil Luyet

Idiap-RR-05-2016

MARCH 2016

# Low-Rank Representation For Enhanced Deep Neural Network Acoustic Models

Master Project Report

Gil Luyet[1]

**Supervised by:** Prof. Bourlard Hervé[2], Prof. Ingold Rolf[3], Dr. Afsaneh Asaei[4], Prof. Marcus Liwicki[5]

March 25, 2016

## Department of Informatics - DIVA

Département d'Informatique - Departement für Informatik • Université de Fribourg - Universität Freiburg • Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland

phone +41 (27) 721 77 11 fax +41 (27) 721 77 12 Diuf-secr-pe@unifr.ch `http://diuf.unifr.ch`

## Idiap Research Institute - Speech & Audio Processing

Idiap Research Institute - Institut de Recherche Idiap • Rue Marconi 19 • 1920 Martigny • Switzerland

phone +41 (26) 300 84 65 fax +41 (26) 300 97 31 `https://www.idiap.ch`

[1]gil.luyet@unifr.ch, University of Fribourg
[2]bourlard@idiap.ch, Idiap Research Institute
[3]rolf.ingold@unifr.ch, University of Fribourg
[4]afsaneh.asaei@idiap.ch, Idiap Research Institute
[5]marcus.eichenberger-liwicki@unifr.ch, University of Fribourg

## Abstract

Automatic speech recognition (ASR) is a fascinating area of research towards realizing human-machine interactions. After more than 30 years of exploitation of Gaussian Mixture Models (GMMs), state-of-the-art systems currently rely on Deep Neural Network (DNN) to estimate class-conditional posterior probabilities. The posterior probabilities are used for acoustic modeling in hidden Markov models (HMM), and form a hybrid DNN-HMM which is now the leading edge approach to solve ASR problems.

The present work builds upon the hypothesis that the optimal acoustic models are sparse and lie on multiple low-rank probability subspaces. Hence, the main goal of this Master project aimed at investigating different ways to restructure the DNN outputs using low-rank representation. Exploiting a large number of training posterior vectors, the underlying low-dimensional subspace can be identified, and low-rank decomposition enables separation of the "optimal" posteriors from the spurious (unstructured) uncertainties at the DNN output.

Experiments demonstrate that low-rank representation can enhance posterior probability estimation, and lead to higher ASR accuracy. The posteriors are grouped according to their subspace similarities, and structured through low-rank decomposition. Furthermore, a novel hashing technique is proposed exploiting the low-rank property of posterior subspaces that enables fast search in the space of posterior exemplars.

**Keywords:** Automatic speech recognition (ASR), hidden Markov model (HMM), Deep neural network (DNN), Posterior probability, Low-rank representation (LRR), Fast k nearest neighbor (kNN) search, Posterior Hashing.

**Acknowledgments**

I would like to thank my whole family and friends which supported and helped me achieving my studies.

I would like to express my sincere gratitude to Prof. Hervé Bourlard for the opportunity he gave me to do my Master work at Idiap Research Institute and being my supervisor. I would also thank Prof. Rolf Ingold and Prof. Marcus Liwicki for being my supervisors at my home University of Fribourg. It gives me immense pleasure to thank Dr. Afsaneh Asaei for all the help she gave me during my work without I would never have made a fraction of the work presented. I would like also to thank all other post-doctorates and doctorates present at Idiap for the help and grateful discussions provided. In particular, I thank Pranay Dighe for assistance with ASR tests and Dhananjay Ram for performing query detection.

This project is from my point of view a exciting full dive in the fascinating world of research. It was a great opportunity allowing me to work in an exceptional environment and achieve a self-rewarding job. Most of the achievements reached were made possible by deep and exciting discussions with remarkable researchers.

It was a great personal experience I recommend to any student.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

*The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!", but "That's funny..."*

— Isaac Asimov (1920 - 1992)

# Section 1

# Introduction

Voice applications for computers is a wide area regrouping many fields. One particular application is Automatic Speech Recognition (ASR) which aims to extract from a given speech signal its actual semantic content. It is usually given by the speech transcriptions containing the spoken words. Also, alternative information such as the intonation of the voice, or other variations denoting, for example, joy, sadness, or sarcasm can be extracted. A general technique composed of three sub-tasks: 1) sampling, 2) analyzing, 3) post-processing can be established to extract the speech transcription. This procedure is illustrated in Figure 1.1.

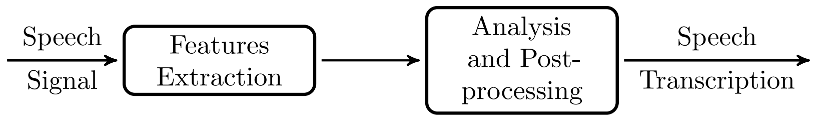


**Figure 1.1:** *Illustration of ASR split into sub-tasks. The speech signal is sampled using feature extraction and processed to extract the speech transcription.*

The sampling part is performed by the feature extraction. Its purpose is to divide the continuous speech signal into discrete chunks which can be processed. The most common types of features used are Mel-frequency cepstral coefficients (MFCC). They rely on the human hearing system specificities to extract from the audio signal only the part corresponding to human speech. The overall procedure consists of dividing the continuous signal into windows of a few milliseconds and using a Fast Fourier Transform (FFT) to transform the time domain signal to the frequency domain. An illustration of FFT is proposed by Figure 1.2. The number of frequencies is then reduced according to the human audible frequencies. The values of the remaining frequencies are representing the components of the vector feature and thus, form the MFCC feature vector for a given time window. The details about speech features are presented in Section 3.2.

The analysis part is classically making use of Hidden Markov Models (HMM). HMM are structures which can map a given sequence of features $[f_1, \ldots, f_k]$ to a sequence of interpretations $[i_1, \ldots, i_k]$. The structure of an HMM is composed of multiple states, which are usually representing a possible interpretation such as a phone. The states are interconnected by following an unidirectional trend : the connections allow to either stay in the same state or to jump in the next state. This idea encodes a time sequence which Figure 1.3 illustrates with the word "cat“. By making use of probabilities to compute the likelihood of a feature to encode a state, HMM can compute the "most probable state sequence“ given a sequence of features. The details of HMM are given by Section 3.3.

Because the voices of different speakers may not encapsulate the same variations, it is usual to abstract the features and produce high-level features which can, for example, become speaker independent. This abstraction can be achieved by using a Deep Neural Network (DNN).

A DNN is an interconnected structure of small independent units called neurons, as depicted by

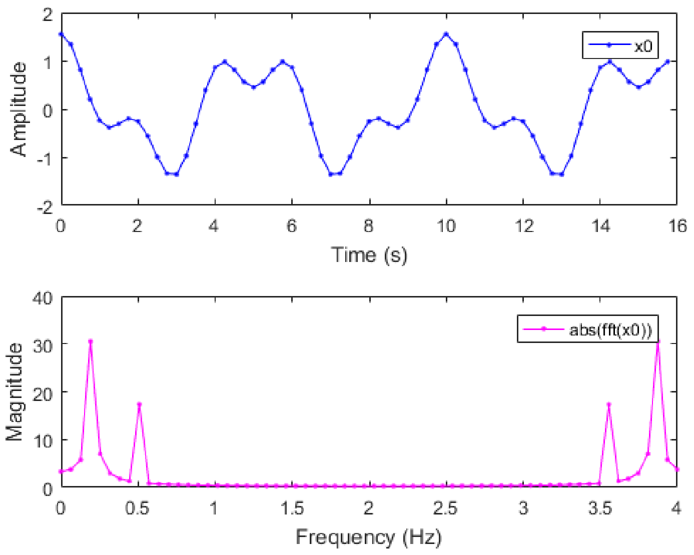


**Figure 1.2:** *Example of a Fast Fourier Transform over the sum of two sinusoidal $x_0$. The result of the FFT yields effectively two spikes (actually four but the results of the FFT are symmetric) which correspond to the two sinusoidal summed to construct the base signal. This figure is taken from MATLAB website*[1]

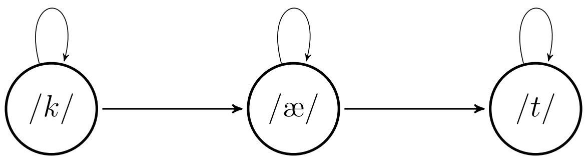


**Figure 1.3:** *Illustration of a simple HMM structure for the word "cat". Each state represents a phone of the word. The next step of a sequence given by the HMM can only be the current or the next state.*

Figure 1.4. Each neuron can forward an incoming signal according to an inner decision criteria. The structure is organized into layers, the neurons within a layer are usually not connected. The first layer is named "input layer", the last layer will output the processed data and thus is called "output layer", and layers in between are called "hidden layers".

By inputting standard MFCC features to such a structure it is possible to collect high-level features at the output layer. Those high-level features represent probabilities of a given MFCC feature to encode an interpretation such as a phone or a word. The vectors of probabilities produced at the DNN output are called posteriors probabilities vectors, often abbreviated posteriors (see Section 2.2).

Those vectors probabilities can be combined with the HMM to produce a paradigm known as DNN-HMM hybrid which constitutes the state-of-the-art system for solving ASR problems. The concept of the DNN-HMM hybrid is given by Section 3.4. This combination is illustrated in Figure 1.5. Even if it is possible to mathematically prove that a DNN can perfectly model the mapping between MFCC features and posteriors, posteriors are often inaccurately estimated. Due to the hard task of DNN training, sparse noise, as well as wrong estimation, reduce the quality of the posteriors. The main goal of this work is thus to re-estimate those posteriors to improve their quality. The main hypothesis of this work is that matrices formed by concatenating posteriors belonging to the same class are low-rank. The low-rankness property can be achieved with a DNN in three distinct ways that are all related to its structure. It is possible to focus on the input, on the connections within the DNN, or at the output.

[1]`http://ch.mathworks.com/help/fixedpoint/examples/convert-fast-fourier-transform-fft-to-fixed-point.html`

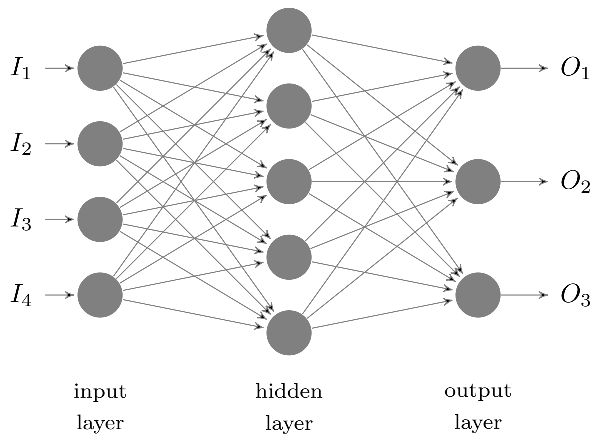


**Figure 1.4:** *Simple Neural Network example composed of one hidden layers of five neurons. Four neurons are used to input the vector* $[I_1, I_2, I_3, I_4]$ *and three neurons are used to compute the output vector* $[O_1, O_2, O_3]$*.*

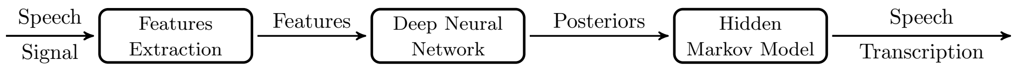


**Figure 1.5:** *DNN-HMM hybrid overall illustration. The speech signal is segmented using MFCC features which are then processed using a DNN to get high-level posterior features. Finally, an HMM allows producing the speech transcription.*

## 1.1 State-of-the-art

To apply low-rankess at the input, it is possible to sparsify the output vectors while using specifically tuned extraction method for input features [58]. Also, it is possible to focus on the structure itself by restructuring the DNN weight matrix. The weight matrix simply represents the connections between neurons. Investigations using PCA [50, 60, 67], structured transforms [54] or rank-constrained topology [37] have been conducted. The ultimate goal consists in reducing the size of the weight matrix as well as its complexity. Such approaches are focused on producing a small footprint DNN to be embedded into mobile devices. By applying such methods it is possible to reduce drastically the size of the DNN while losing slightly no accuracy for the tasks investigated.

The last possible way to apply low-rankness property to a DNN is to focus on the output layer. This is the main goal of the method proposed by this work. Ultimately, low-rank methods are going to be applied to reduce the rank of matrices formed by aggregation of posteriors belonging to the same class. To group the posteriors according to their classes, clustering algorithms are used. This results on a method consisting on

1. Grouping posteriors according to their class-membership and forming matrices by concatenating them.

2. Applying low-rank algorithms to reduce the rank of those matrices.

To the best of the author knowledge, this is a novel method to enhance posterior probability vectors.

By making use of this method and also by relying on experiments on sparse-coded posteriors, low-rankness property of matrices formed by posteriors from the same class-membership is proven to be a strong indicator of the posteriors quality.

Relying on that property, a novel hashing technique for fast search over posterior space is proposed. This technique consists of a flooring function, applied to the posteriors, as an approach to produce hashing keys. A regularization parameter taking the form of quantization bits enables to modify the hashing precision of the keys generated. As matrices of posteriors with the same class-membership are low-rank, the hashing keys will tend to be often the same.

## 1.2 Research Questions

The posterior vectors space has unique properties that this project aims to explore. More precisely, the goal is to exploit low-dimensional structures underlying a large number of training posterior probabilities to enhance local estimates of the test posteriors. Along this line, the following questions are studied:

**Q1.** What is the intrinsic dimension of class-specific posterior probability vector space ?

**Q2.** Can single/multi subspace low-rank representation methods enhance posteriors ?

**Q3.** What are the applications of low-rank posterior property in hierarchical acoustic modeling ?

Furthermore, the impact of exploiting the low-rank property and enhanced acoustic models on other speech applications relying on posterior probability estimation such as query detection and fast search in the space of posterior exemplars are discovered.

It is envisioned that characterizing and exploiting the low-dimensional structures in posteriors space can lead to new paradigms of hierarchical speech recognition that will not by restricted to the constraints of Markov model topologies, thus more flexible and robust than the current technologies.

## 1.3 Report Outline

Section 2 presents the concept of DNN and its use in the estimation of class-conditional posterior probabilities. The state-of-the-art system for DNN based automatic speech recognition (ASR) is reviewed in Section 3. The low-rank representation methods are presented in Section 4 which covers both single subspace and multi-subspace low-rank representation techniques. In addition, sparse coding is considered as it relies on the characterization of the low-dimensional subspace of class-specific posteriors. The evaluation methodology to quantify the quality and performance of posteriors and speech recognition are described in Section 5.

The experimental analysis on low-rank representation of individual posterior classes are conducted in Section 6 followed by low-rank representation of the posteriors of the whole speech corpus to enhance ASR performance in Section 7. The impact of the theory of low-rank posteriors and enhanced acoustic modeling for query detection as well as posterior exemplar-based speech classification are studied in Section 8 where a novel hashing technique is proposed for fast search in posterior space. Finally, the conclusions are drawn in Section 9.

## 1.4 Environment of Work

To achieve this work a lot of resources and programing were needed. The following section details the resources available as well as the programing languages, frameworks and toolkits used.

### 1.4.1 Computing Resources

In order to perform all the computation, a personal workstation was available at Idiap, consisting of the following hardware

**OS** : Debian GNU/Linux 8 (Idiap edited version)

**CPU** : Intel(R) Core(TM) i7 CPU 950 @ 3.07GHz

**GPU** : Nvidia GeForce 8400 GS/PCIe/SSE2

**RAM** : 12Gb

**ROM** : External network storage (up to 1Tb)

In addition a computation grid was available providing the following configuration

**Hardware** : Dedicated computation nodes along with user workstations

**Architecture** : Intel 64-bits (single and multi-core)

**OS** : Debian 8 (Jessie) 64-bit

**Resources** : $\sim$100 hosts / $\sim$400 cores

**DRM** : Sun's Grid Engine (SGE)

### 1.4.2 Programing Languages

Most of the implementation was done using MATLAB[2] which is a well-known high-level language specifically designed for mathematical operations on matrices. MATLAB version used for the workstation use was MATLAB 8.5. MATLAB 8.1, as well as MATLAB to C/C++ compiler (version 6.0), were used for running jobs on the grid. This language was well suited due to the matrix nature of the posteriors probability representation of utterances used in this work. In addition, many algorithms acting on matrices were easily found online. A negligible amount of programs has been written in Python and in Shell for restructuring data sets or for automatic job dispatching on the available grid.

### 1.4.3 Kaldi Toolkit for Speech Recognition

> According to legend, Kaldi was the Ethiopian goatherder who discovered the coffee plant.[3]

For this project, the toolkit used to work with Artificial Neural Networks (ANNs) and perform Automatic Speech Recognition (ASR) is Kaldi [41]. According to its documentation, Kaldi is written in C++ (licensed under Apache License v2.0) and specially designed for research purpose. Kaldi was initially created in 2009 at Johns Hopkins University workshop but the software was not completely finished. Some of the developers that were working on met again in August of 2010 in Brno (Czech Republic) to produce a modern, well-designed version of the toolkit. The first code was officially released on the 14th of May 2011. More than 70 contributors have participated in its creation. Kaldi is complex and allows many combinations of existing ASR techniques such as Deep Neural Networks or Gaussian Mixture Models. Because it is open-source modification of its code[4] it is also possible to integrate new techniques.

[2] `http://ch.mathworks.com/help/matlab/`

[3] http://kaldi-asr.org/doc/about.html

[4] `https://github.com/kaldi-asr/kaldi`

## Section 2

# Neural Network Posteriors

A Deep Neural Network (DNN) is a structure of interconnected small independent units called neurons. While doing a very simplistic analogy about the functioning of a brain, the information will be passed from neuron to neuron to produce a high-level interpretation of it. Each neuron can be seen as a processing unit which will forward the information based on an internal decision system. Such a decision system is often modeled by a sigmoid function taking as the parameter a sum of signal values coming from other neurons. The connections between the neurons are all weighted. Those weights are usually the parameters of the structure needed to be tuned during the learning phase. Figure 2.1 illustrates three incoming signals ($x_1$, $x_2$ and $x_3$) entering the structure of a neuron.

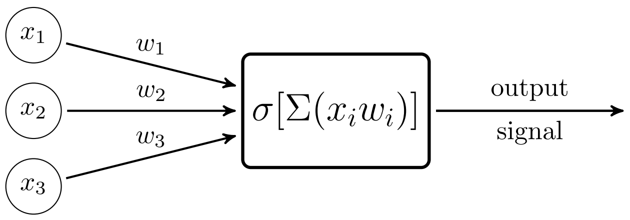


**Figure 2.1:** *Structure of a neuron. A weighted sum of three incoming signals ($x_1$, $x_2$ and $x_3$) is used as parameter to an activation function $\sigma$ to produce the output signal.*

It is convenient to represent the structure of the DNN by making use of layers. Each layer is usually connected to the following one using unidirectional weighted connections. The information is inserted into the DNN at the first layer, the input layer, and then processed layer by layer to finally produce a certain activation scheme at the last layer, the output layer. This type of architecture is depicted in Figure 2.2.

By using this procedure, the information can be, for example, classified with the power of deep learning. Deep Neural Networks achieve impressive results in many domains such as face recognition [56], image segmentation for medical science [28], DNA analysis [1], or in the Google DeepMind project[1] and its related achievements [34, 33, 53].

For the scope of this work, the considered DNNs are estimating class-conditional posterior probabilities given the inputted data. Figure 2.2 illustrates a DNN using three neurons for the output layer. Therefore, the DNN is able to estimate the posterior probabilities for three classes, namely, $p(C_1|I)$, $p(C_2|I)$, and $p(C_3|I)$. This section proposes a review of the DNN structure, training and posteriors estimation.

[1]`https://deepmind.com`

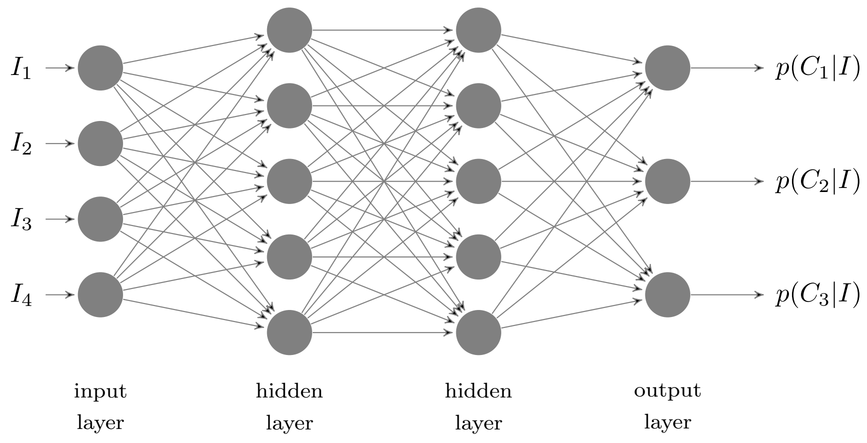


**Figure 2.2:** *Simple DNN example composed of two hidden layers of five neurons. Four neurons are used for the input and three neurons are used for the output. The DNN estimate class-conditional posterior probabilities, for each class, given the input $I$.*

## 2.1 Multi-Layer Perceptron and Deep Neural Network

A Multi-Layer Perceptron (MLP) is a particular kind of Artificial Neural Network (ANN) composed of layers of individually connected units called neurons. Each layer is usually connected to its adjacent ones at the neuron-level. In opposite, within a given layer the neurons are not connected. The connections are unidirectional and weighted. Each neuron can be seen as an independent decision unit which can output a signal depending on the sum of the signals it received. For a given neuron $i$, the output $o_i$ is given by

$$o_i = f(\sum_{j \in J} w_{j,i} a_j) \tag{2.1}$$

where $J$ is the set of neurons connected to the neuron $i$ at the upper layer, $w_{i,j}$ the weight of the connection from $i$ to $j$, and $a_j$ the value of the output of neuron $j$. Typically, the function $f$ is a sigmoid that allows to make a relatively strict decision and can be roughly seen as the following

$$f = \begin{cases} 1, & \text{if } \sum_{j \in J} w_{j,i} a_j < \text{threshold.} \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

More complex functions are generally used. Especially, each function used has to be differentiable in order to be trained (Section 2.1.1). Three specific kinds of layers are distinguishable. The first layer is the input layer and receives information to be classified or interpreted. The last layer is the output layer and outputs a decision about classification or an interpretation of the inputted data. The inner layers are called hidden layers and are used to process the input data to form the output. In the case of multiple hidden layers an MLP is called Deep Neural Network. Also, a bias can be added to each neuron. This bias is usually a weighted connection which is always active and modifies the output formula for each neuron as

$$o_i = f(\sum_{j \in J} w_{j,i} a_j + b_i) = f(v_i + b_i) \tag{2.3}$$

such that $b_i$ is the bias for neuron $i$ and $v_i$ is the weighted sum of the inputs of neuron $i$.

### 2.1.1 Back-Propagation for MLP Training

Given the structure of an MLP, the only modifiable parameter for training are the weights of the connections (including the bias). The training of an ANN is always reduced to a weight adjustment. Two main approaches to training can be distinguished, on-line and off-line training. In off-line training, the weights are re-estimated after all the training data has been processed. In on-line training, the weights are re-estimated after each training data sample (or a batch of training data samples). The most common method used for learning is the back-propagation algorithm.

Its goal is to make use of the error yield by a given input. Each training vector used is a labeled vector which label correspond to a certain output expectation. It is then possible to compute an error for a given vector according to its expected output. As stating the detailed equations of this complex procedure is out of the scope of this report a brief explanation of it will be provided. Back-propagation algorithm uses a cost function $C$ as a basis. Using partial differentiability of this function it is possible to prove that the following equation holds [40].

$$\Delta w_{j,i} = -\eta \frac{\partial C}{\partial v_j} o_i = \eta (p_i - o_i) f'(v_j) \tag{2.4}$$

where $\eta$ is call the learning rate and $p_i$ the expected output of neuron $i$. To summarize, the variation to apply to a given weight depends on the error made (scaled with the learning rate parameter) times the direction of the activation function represented by its derivative. A similar equation can be established for learning the bias of each neuron. This computation is nontrivial and necessitates a lot of calculus. All details about this procedure are present in the famous paper of Rumelhart et al. [49] and on the online book of Michal Nielsen [40].

### 2.1.2 Deep Neural Network Training

From one of it first use, with the concept by Kunihiko Fukushima [19] with the Neocognitron (a precursor to Convolutional Neural Networks) and the wake-sleep algorithm of Hinton et al [25], Deep Neural Network training or Deep Learning has been a challenging task. The most important issue is known as vanishing and exploding gradient [7]. It designates a problem occurring during back-propagation, the impact of the error decreases layer by layer. A succinct example is proposed by [40] which consist on a simple DNN illustrated by Figure 2.3 with three hidden layers with only one cell per layer. Each neuron $n_i$ has a sigmoid activation function, the weighted input of each

$$\begin{tikzcd} \boxed{\text{Input}} \arrow[r, "w_1"] & n_1 \arrow[r, "w_2"] & n_2 \arrow[r, "w_3"] & n_3 \arrow[r, "w_4"] & n_4 \end{tikzcd}$$

**Figure 2.3:** *Small DNN example used in the online book of Michael Nielsen [40] for vanishing gradient illustration. The DNN is composed of three hidden layers and one neuron per layer.*

neuron is denoted by $z_i$, the output by $a_i$ and the bias by $b_i$. It is possible to show that given $C$ any cost function the following equation holds [40].

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) w_2 \sigma'(z_2) w_3 \sigma'(z_3) w_4 \sigma'(z_4) \frac{\partial C}{\partial a_1} \tag{2.5}$$

Given that the activation function is a sigmoid then the maximal of its derivative is $\frac{1}{4}$. This means that the repercussion achieved by back-propagation on the first layer will be downgraded by a factor $\frac{1}{16}$. This is, of course, a qualitative example but it shows how the gradient can react across the layers. For this project, the Karel's DNN implementation of Kaldi[2] was used (Section 1.4.3). To overcome the issues with the gradient, three consecutive steps are proposed:

**Restricted Boltzmann Machine (RBM) pre-training** - Restricted Boltzmann Machines (RBMs) are conceptually restricted Neural Network with only two layers [23]. The connections between neurons inside an RBM are only between layers and bidirectional. This pre-training step consists of training pairs of connected layers by treating each DNN layers two-by-two as an RBM. The method used for training is called Contrastive Divergence [23] or CD-1. By inputting with a given vector to the first layer of the RBM the second layer is then activated according to the length. It is then possible to reconstruct its image by activating the neurons previously triggered. Contrastive Divergence aims to minimize the error between the input vector and its reconstructed image. Conceptually it aims to make the DNN first learn a representation of the data before performing a discriminative training.

[2]`http://kaldi.sourceforge.net/dnn1.html`

**Frame cross-entropy training** - The second step is a back-propagation training using cross-entropy between the input and the output as an error function. The DNN is trained frame-by-frame.

**Sequence-discriminative training** - The last step consists of performing a discriminative training which objective is that the DNN makes a clear distinction between classes. Generally, this is achieved by using a criterion for the error such as Maximum-mutual information (MMI) or Minimum Bayes Risk [60]. The DNN is trained for sentences using error aggregation over many frames and no more frame-by-frame. This last point makes the DNN achieve a more closed behavior to the ASR needs than a frame-level training.

## 2.2 Posterior Probabilities

It has been proven that an MLP can approximate any multi-variate functions [65] and especially [9, 38, 46] model conditional expectation $\mathbb{E}[T|I]$ of a target $T$ given an input $I$. The MLP is then trained in a discriminative way to model posterior probabilities. Concretely, to produce a probability vector at the output of an MLP a softmax function is used at the output layer.

More impressive analysis was conducted by [2] to prove that posteriors are speaker-invariant and encapsulate the phonetic information of the speech. In general, posteriors are high-level features well suited for speech recognition [69, 72].

As the work presented in this report modify posterior probability vectors formed by a DNN by making use of various algorithms, it is convenient to name those posteriors. From now they will be referred as DNN posteriors. It is possible to distinguish several types of posteriors. For the scope of this report, the focus will be laid on phone and senone posteriors.

### 2.2.1 Phone Posterior Probabilities

Phone posteriors are estimating the probability of a phone given an input feature. They are called context independent because they do not encapsulate any knowledge about the phone sequence/context.

Recently, phone posteriors have been used in multilingual ASR [62]. This is mainly due to their invariance given a language. Theoretically, it is possible to identify common phones across languages. The well-known International Phonetic Alphabet (IPA) identifies 107 vowels and consonants with additional variations (such as tone or intonation). Given that observation, it seems reasonable to use them to build ASR systems which could recognize many languages. Investigations on this principle have been recently led by researchers [63].

### 2.2.2 Senone Posterior Probabilities

The second type of posteriors used are called senone posteriors. They estimate the probability of a complex unit specially tuned for an HMM (Section 3.3.3). Those posteriors are context dependent because they are part of a tri-phone scheme which holds previous and incoming phone context. Each senone state describes a phone given a certain situation (previous and next occurring phone). In that sense, they are generally more accurate than phone posteriors.

The drawback of this method is the number of data needed to train the DNN so their estimation becomes accurate and thus even if the states of the tri-phones are tied to produce senones.

Furthermore, their use in multilingual ASR is restricted. As the tri-phone structure may vary from the language it is hard to relocate a trained DNN to another language. In addition, to this, the fact that the tri-phones states are often tied to form senones makes the process even more difficult.

## 2.3 Diagonal Plot of MAP vs. Classification Accuracy

A common experimental technique to visualize the probability correspondence of the posteriors is to plot the correspondence of the classification rates according to the value of the Maximum

A-Posteriory (MAP). Each posterior probability vector $p$ consists of the probability of individual classes given the input feature $[p(c_1|x_t), \ldots, p(c_k|x_t)]$. Given that observation, it is possible to confirm that indeed the given posterior probabilities are describing a true probability distribution.

One can simply regroup the posteriors by the value of their MAP. The MAP associate the most probable output class to the one yield by the maximum a posteriory probability.

Given a posterior $P = [p_1, .., p_n]$ the MAP is defined as $\arg\max_i p_i$. This can be seen as the most likely class that the posterior probability vector represents. Typically, it is convenient to form bags between 0 and 1 and assign every posterior to each bag given by its MAP. Then it is simple to compute the percentage of correctly classified posterior for each bag given the class labels for each posteriors. This value should be similar to the average value of the bag (ie. if the posteriors have MAP between 0.6 and 0.7 then the percentage of correctly classified posteriors should be 0.65). Intuitively speaking, the plot should follow a diagonal trend. This diagonal analysis is proposed by [10].

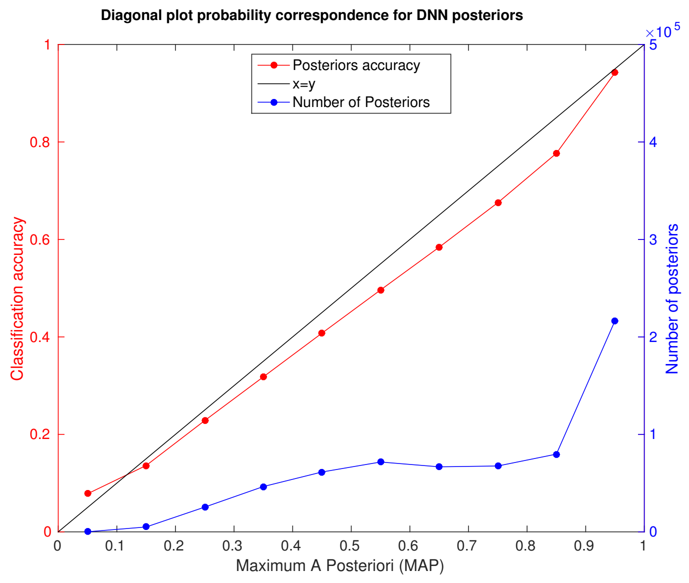


**Figure 2.4:** *Diagonal plot (red) together with assignation (blue), distribution of posteriors for each bin, depicting the probability correspondence of posterior probabilities.*

Figure 2.4 presents two plots illustrating the distribution of a large number of posterior probabilities from test posteriors of the database presented in Section 5.2.1. The first plot (in red) shows the correspondence between posterior classification accuracies as a function of the observed posterior probabilities (as estimated at the DNN output) of the most likely class. As discussed earlier, the ”optimal” Bayes classifier should match the diagonal since posterior probabilities are theoretical estimates of classification accuracies. The closer the posterior distributions are to the upper right side of this diagonal, better the final performance should be. Hence, the second plot (in blue) also shows the distribution (number) of posterior vectors within the different bins in [0,1]. The better the quality of the posteriors is the more they will be shifted to the right and thus be more often correctly classified.

## 2.4 Summary and Posterior Enhancement

This section explained the structure of a Deep Neural Network as well as its training method. Moreover, while trained in a discriminative way, the DNN can model posterior probabilities of its output classes.

A DNN can model a various type of posteriors. For the scope of this report, both phone and senone posteriors have been detailed, being respectively context independent and context dependent high-level features.

Relying on the theory, posteriors probability should optimally be sparse, but, given the arduous task of DNN training, they are often wrongly estimated and tend to become spurious.

The novel method proposed in this report intents to sparsify the posteriors probabilities by re-estimating them using low-rank techniques at the output of the DNN. This method will be investigated on the task of automatic speech recognition and by making use of a combination of Deep Neural Networks and Hidden Markov Models. This system is described in the next section.

# Section 3

# Automatic Speech Recognition

Deep neural network and Hidden Markov Models (HMM) can be combined to achieve a state-of-the-art system for Automatic Speech Recognition [10, 24]. An HMM is a sequential structure which enables to compute the likelihood of a sequence of interpretations that match a sequence of input. For the case of speech recognition, the interpretations can be for example phones. A state $S_i$ is used for each possible interpretation. Each state owns a probability density function which models the probability of an input given the state itself. The transitions between states are driven by transition probabilities $P(S_{i+1}|S_i)$. Such a structure is illustrated in Figure 3.1. Ultimately, the HMM is

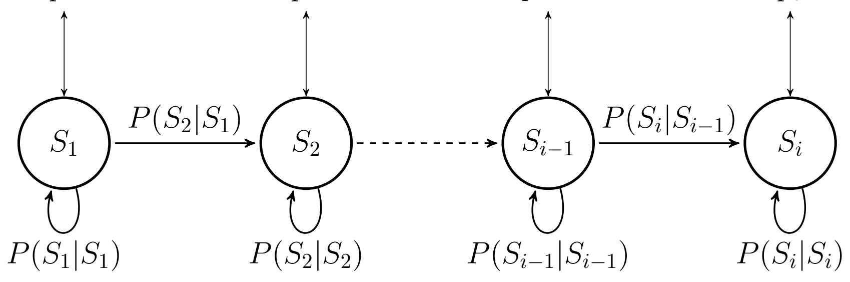


**Figure 3.1:** *Example of the structure of an HMM. Each state owns its own probability density function. The transition between states is driven by transition probabilities.*

used to compute the most probable sequence of states given an observation sequence of speech features. This is conceptually done by computing all the possible probabilities yield by aligning the observation sequence to a sequence of states. The likelihood is given by a multiplication of all the obtained probabilities. The goal is then to find the sequence of states which maximize the likelihood.

As previously stated, an HMM needs probability density functions for each of its states in order to compute the probability of a speech features given a class $p(I|C)$. By making use of a DNN, it is possible to estimate the probability of a class given some input $p(C|I)$. This probability can be scaled with the class likelihood, $p(I|C) \sim \frac{p(C|I)}{P(C)}$ , to obtain the desired probability density function. The DNN then provides an acoustic model in the form of a probability vector which suits the needs of the HMM. It is then possible to combine the estimation of the posteriors produced by the DNN with the power of the HMM to get the most likely sequence of states which model the observation sequence. This sequence of states is known as the HMM decoding. Such a combination is illustrated in Figure 3.2. The obtained decoding usually represents a sequence of phones or other sub-phones units which can be easily converted to words and sentences which represent the speech transcription.

While estimating class-conditional posteriors probabilities for an HMM, a DNN outperforms other methods, such as Gaussian Mixture Models (GMMs). This section proposes a description of the principles of the DNN-HMM hybrid used in speech recognition tasks.

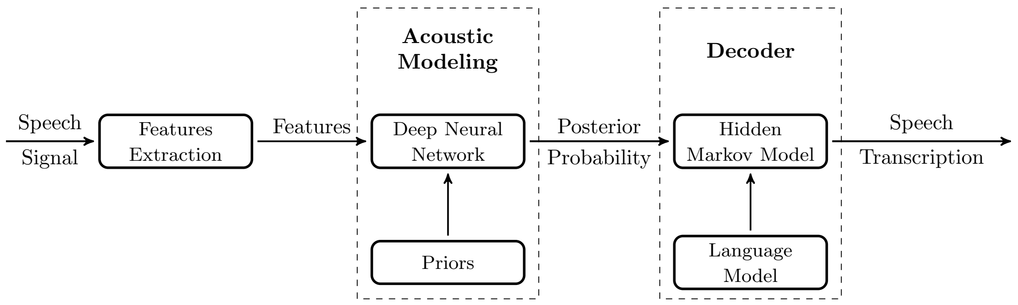


**Figure 3.2:** *Automatic Speech Recognition division in sub-tasks for a DNN-HMM hybrid.*

## 3.1 Automatic Speech Recognition

Automatic speech recognition (ASR) refers to the task of extracting speech transcription from spoken signals. It can be implemented using the classical structure of pattern recognition which can be split into sub-tasks, as illustrated in Figure 3.2, which are

**Feature Extraction** - Given an audio signal that contains spoken utterances the goal of the feature extraction is to transform this continuous signal into discrete values (usually vectors which encode the time-frames). Those values should be specifically designed to help the system to perform recognition on this signal. Meaning that most of the semantic information of the time frame should be sampled to achieve recognition. This extraction is usually done using Spectral features.

**Acoustic Modeling** - The features are used to produce acoustic models. For the scope of this report, a Deep Neural Network has been used to produce acoustic models know as posterior probabilities.

**Decoding** - Decoder purpose is the extraction of text transcriptions using a sequence of acoustic models. Such a work is usually done by Hidden Markov Models. It generally makes use of grammatical priors such as words used in typical context for example.

Many combinations are possible and others sub-tasks can be added such as pre-processing on the audio data to remove noise. This general scheme roughly describes how proposed solutions to ASR are designed.

## 3.2 Spectral Features Extraction

The purpose of feature extraction is to distillate from a signal only the useful information which allows performing optimum decision on its content. In the case of speech recognition given a complex continuous audio signal, feature extraction allows to transform it into discrete and simpler features. The standard model for speech signal is based on the following principle [2].

$$f(t) = e(t)h(t) \tag{3.1}$$

where $h$ denotes the modification of the voice, which is more speaker or context dependent, and $e$ denotes the actual informative content of the speech (usually phonemes). The common features used for speech applications are the Cepstrum-based features. It is possible to list two main class of features Perceptual Linear Prediction (PLP) and Mel-frequency cepstral coefficients (MFCC). For the scoop of this work, only MFCC features were used. The creation of MFCC features given some input audio signal can be decomposed into the following steps.

1. A division of the signal using Hamming windows is performed. The signal is divided into frames using sampling. Hamming windows then allow putting more emphasis in the center amplitudes of the windows.

2. Fast Fourier Transform (FFT) is applied to extract the power spectrum. Only the frequency domain is needed because it is where the speech information is mostly encoded [42].
3. The interesting power of the resultant spectrum are kept by using Mel scale filterbanks. It keeps only the powers that human is supposed to hear. As speech applications are mostly human-oriented it is not meaningless to rely on the human hearing system to model the features. In this case, it means that more the components with lower frequency tend to be kept.
4. Logarithm operation is also applied to the previously obtained vectors.
5. The number of components is then reduced usually down to 13.
6. Finally the vectors obtained are decorrelated by applying a Discrete Cosine Transform (DCT).

In general, as speech is continuous in time, there is a strong correlation between each feature computed. For this reason it is common to had to those features vectors first and second order temporal derivatives know as $\Delta$ and double-$\Delta$ (or $\Delta$ $\Delta$). Those derivatives are computed over a context of 5 frames [20]. Which leads to an 39 dimensional vector (13 MFCC features plus 13 $\Delta$ and 13 $\Delta\Delta$ derivatives)

## 3.3 Hidden Markov Model

Hidden Markov Models (HMM) is an efficient method to model a piecewise stationary observation sequence [43]. An HMM follows the structure of a Markov Chain which the states sequence underlying the observable stochastic process is hidden.

As example, [43] proposes a game with multiple jars containing colored balls. The player picks and put back balls from the jars in a certain sequence. The sequence of jars hidden but the probability of picking a certain color given a jar is known. By having the sequence of color it is possible to compute the most probable sequence of jars on which the balls has been picked.

For the case of speech recognition, the goal is to align extracted speech feature to a previously drawn Markov model. It is like trying to find the most likely sequence of jars (states) producing the best match to the sequence of colors (features). In that case, the sequence of states that should encode the sequence of features is hidden and needs to be estimated. Concretely, a Bayesian probability maximization problem is computed, expressed as

$$\tilde{I} = \arg\max_{I} \frac{p(X|I)P(I)}{p(X)} \tag{3.2}$$

where $I$ is the set of all the possible interpretations, $X$ is the feature sequence and $\tilde{I}$ represent the most probable interpretation. In the case of speech recognition, the interpretations can be for example phonemes or words. $P(I)$ can be estimated by prior knowledge about the grammar or the language. $p(X)$ is generally computed using a training set and denotes the probability of the observation to occurs. The production of $p(X|I)$ which is the core of the HMM method is more complicated. One way to do is to introduce a new variable $Q$ to get (3.3).

$$p(X|I) = \sum_{Q\in\Psi} p(X|Q,I)P(Q|I) \tag{3.3}$$

where $X = [x_1, \ldots, x_N]$ denotes an sequence of $N$ observations and $\Psi = [q_1, \ldots, q_N]$ denotes the set of all state sequence of length $N$. Rearranging (3.4) yields to :

$$p(X|I) = \sum_{Q\in\Psi^I} p(X|Q)P(Q) \tag{3.4}$$

where $\Psi^I$ is a restriction of $\Psi$ where $P(Q|I)$ needs to be non-zero. Assuming now that

1. The states follows a Markov chain, $P(q_i) = \prod_{t=1}^{i} (q_t|q_{t-1})$. This means that the probability of then next event only depends on the previous event.
2. Each observation is independent given the others: $p(X|Q) = \prod_{t=1}^{N} p(x_t|q_t)$. Meaning that the components of the multivariate random variable X are not correlated and thus independent.

The probability can be expressed as

$$p(X|I) = \sum_{\Psi^I} \prod_{t=1}^{N} p(x_t|q_t) P(q_t|q_{t-1}) \tag{3.5}$$

Given (3.5), it is possible to compute the probability of the observation $X$ given the class $I$. An illustration of this procedure is given by Figure 3.3.

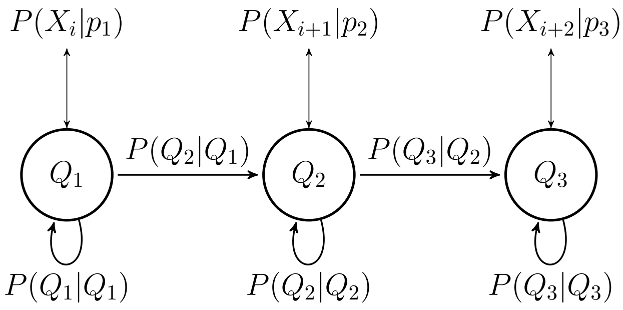


**Figure 3.3:** *Illustration of the HMM structure with three states. Each state is in relation with a probability density function $p_i$ and $X_t$ represent some observation at time $t$. This kind of model is often called left-to-right model and encodes a temporal structure.*

In practice, the structure of the HMM is generally drawn by hand. Some methods propose to learn the HMM structure [51].

The states probabilities are then learned from training data. Given some observations coming from the test data, one can compute all the probabilities among all the states and choose the path that leads to the highest probability as an interpretation of the observations. This path is called a decoded alignment. On the opposite, it is possible to align a signal, which transcription is known, to a sequence of states. This alignment is called forced alignments and is generally used to label the training data. Details about this procedure are given by [43].

Of course, computing all paths is CPU demanding. If the number of states is $N$ and the number of observations is $T$ then one should do $2 * N * N^T$ computations to get all the path probabilities. Given a simple example if $N = 10$ and $T = 100$ (10 states and 100 observations which relatively low comparing a real-case scenario) then the number of computations needed is $\approx 10^{100}$. More suitable solutions in terms of computation steps are given by the Viterbi algorithm [18].

### 3.3.1 Viterbi Algorithm for Word Sequence Decoding

Based on dynamic programing the Viterbi algorithm is used in this specific case to find the lowest-cost path in the graph of all possible probabilities alignments for a given observation. Given the states $\Omega = [q_1, \dots, q_t]$ of a hidden Markov model, initial probabilities $\pi_i$ of being in state $q_i$ and the transition probabilities $p(q_j|q_i)$ describing the possibility of transition from state $q_i$ to $q_j$. The most likely state sequence $\tilde{q_1} \dots \tilde{q_r}$ that describes some observation $I = [i_1, \dots, i_r]$ is given by Algorithm 1. This algorithm may seems hard at first glance but can be seen by a straightforward low-cost path selection on a graph. It can be summarized into finding the shortest path on that graph by finding step-by-step the lowest-cost path to the next state starting to the previous state reached. The path gives the state sequence $[\tilde{q_1} \dots \tilde{q_r}]$ which fits the best the observation $I = [i_1, \dots, i_r]$. This graph description is illustrated by Figure 3.4.

**Algorithm 1:** Sample Viterbi algorithm for HMM states alignment

```
Input  : $\Omega = [q_1, \dots, q_t]$, initial probabilities $\pi_i$ of being in state $q_i$, transition probabilities
         $p(q_j|q_i)$ and observation $I = [i_1, \dots, i_r]$.
Output : $\tilde{q_1} \dots \tilde{q_r}$ the most likely state sequence that describe $I$ and its associated likelihood.
1 $C_{1,n} := p(i_1|q_n)\pi_n$;
2 $c_{1,n} := \arg\max_n p(i_1|q_n)\pi_n$;
3 $\tilde{q_1} = q_{c_{1,n}}$;
4 for $k \leftarrow 2$ to $r$ do
5     $C_{k,n} := \max_m(p(i_k|q_n)p(q_m|q_n)C_{k-1,n})$;
6     $c_{k,n} := \arg\max_m(p(i_k|q_n)p(q_m|q_n)C_{k-1,n})$;
7     $\tilde{q_i} = q_{c_{1,n}}$;
```

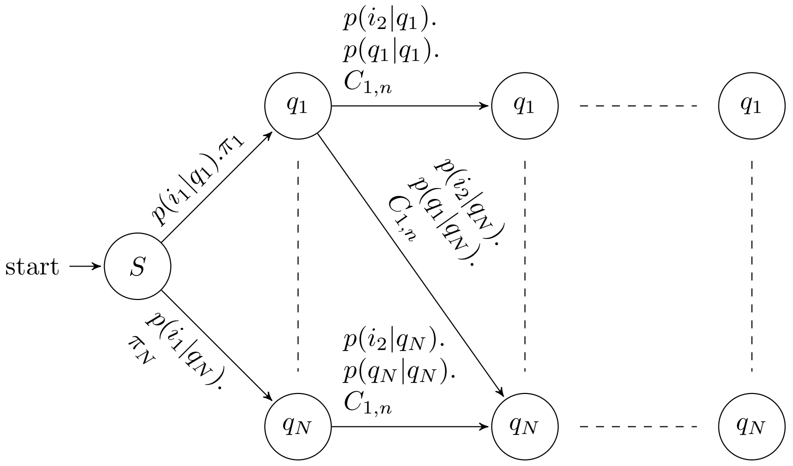


**Figure 3.4:** *Viterbi algorithm seen as a lowest-cost path problem.*

### 3.3.2 Gaussian Mixture Model for State Emission Probabilities

Given an observation $I$ the probability distribution $p(I|q_t)$ has to be modeled from the training data for each state $q_t$. A simple way to do is to use a single Gaussian and to model it by finding the variance $\sigma_j^2$ and mean $\mu_j$. Those parameters are usually estimated by $\hat{\mu}_j$ and $\hat{\sigma}_j^2$ found by applying statistical methods. But the reals probability distributions may not be Gaussian. In order to counteract this effect Gaussian Mixture Model (GMM) are often used to model those complex probability density functions. The idea consists by the density functions of the state by a weighted sum of $k$ Gaussian (or Normal).

$$q_i(x_t) = \sum_{j=1}^{k} w_j \mathcal{N}(x_t, \mu_j, \sigma_j^2) \tag{3.6}$$

The numbers of Gaussian used, their parameters and the weights are learned using the training data [2]. The method first make use of Expectation-Minimization [13, 36] algorithm to find the actual values of $w_j$, $\mu_j$ and $\sigma_j^2$. The number of Gaussian is then estimated using two strategies

1. The number of Gaussian used is increased until some convergence criteria is reached.
2. The data used from training is clustered using an unsupervised clustering algorithm such as k-Means [66]. A Gaussian is then associated with each cluster. Each Gaussian parameters can be estimated by using the data on each cluster.

### 3.3.3 Tied-States of Tri-Phones or Senones

Often by using HMM the main problem is the lack of training data, especially while using GMM. If the model is complex and the amount of available data is limited then problems of training can occur. One can overcome this issue by making use of tied-states or senones in a tri-phone HMM model [68]. In that case states within and across phones are tied together by using the same probability distribution. This allows sharing data between states for modeling the probability distribution functions. This can model phones which start slightly with the same acoustic and thus model similar tri-phones with the same training data. In fact, apparently non-related states will be put together for modeling and during decoding, tri-phone context will help to differentiate them. This modelling takes place in four steps.

1. The tri-states are modeled for each monophone using their own Gaussian.
2. States are bound together to form tri-phones.
3. Similar states are tied together inside tri-phone with similar monophones. A representative state is chosen and other similar state are identified to it.
4. The number of Gaussian needed for each new tied-state can be estimated using some development set.

Figure 3.5 illustrates this procedure.

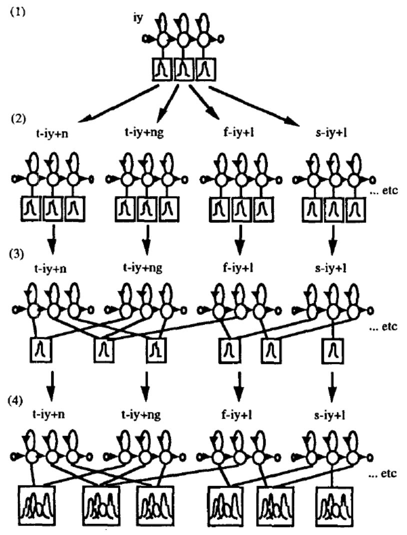


**Figure 3.5:** *Procedure to build tied-states. Figure from Young et al. [68]*

## 3.4 DNN-HMM Hybrid

Section 2.2 explains that the output of a DNN can be seen as a posterior probability vector of its outputs classes conditioned on the current input In the case of ASR, they often model high-level components such as phones or sub-phones probabilities.

As introduced in [9, 10], hybrid HMM-ANN systems are using ANN posterior output instead of GMMs by considering an HMM which states correspond to the outputs of the DNN. Using training data one can compute the prior of each class and thus compute the scaled probability that can be used in an HMM with the following formula using Bayes' rule :

$$p(I_t|q_t = i) = \frac{p(q_t = i|I_t)p(I_t)}{p(q_t = i)} \tag{3.7}$$

where $I_t$ is the observation at time $t$ for state $q_t$ and $p(q_t = i)$ is the prior probability of class $i$. This estimation method for the HMM states is described in details in [10]. According to [2] using DNN to estimate the emission from the data is better for three reasons

1. The DNN is discriminant among the classes.
2. Being able to use frame context can better model correlation between the acoustic features.
3. The non-linear classification of the DNN is better to achieve segmentation on the non-trivial boundaries of the acoustic space.

The model mainly used is a tri-phone DNN-HMM hybrid [60]. Generally, the training of such an architecture can be done in five main steps.

1. Construct a monophone HMM making use of standard MFCC or PLP features.
2. Produce a tri-phone HMM by making use of prior knowledge about the grammar.
3. Use tied-states (Section 3.3.3) to overcome the potential lack of training data.
4. Produce forced HMM alignments for the training-data to have ground truth to train the DNN.
5. Train the DNN using the forced alignments as ground truth and acoustic features as input.

Once the DNN-HMM hybrid is constructed the decoding of any new speech signal consist of processing extracted features with the help of a DNN and use the posteriors produced to fill the HMM and to get decodings.

## 3.5 Summary

The DNN-HMM hybrid has been presented in this section. By making use of posterior probabilities estimation, the DNN can be coupled to an HMM and produce a robust paradigm that can be used to solve ASR problems without the need of GMMs.

As stated in Section 2.4, the posteriors estimated by the DNN of the DNN-HMM hybrid system are going to be enhanced. This will be achieved by using the following methods.

- Low-Rank Representation methods on class-specific posteriors (Section 6).
- Low-Rank Representation combined with clustering methods to achieve multi-subspace enhancement (Section 7).

The following sections detail the low-rank representation methods needed to lower the rank of matrices of posteriors of the same class-membership.

# Section 4

# Low-Rank Representation Methods

To enhance the posterior probabilities acoustic models, several low-rank representation algorithms are used. Their main goal is to reduce the rank of matrices formed by aggregating posteriors of the same class-membership. Four main approaches are considered here; Principal Component Analysis, Robust Principal Component Analysis, Low-Rank Representation, and Sparse Recovery. Especially, Robust Principal Component Analysis and Low-Rank Representation both use specifically a low-rank criterion in their respective cost function by making use of the nuclear norm (sum of singular values of a given matrix) as well as an error extracting part using $l_1$ norm to extract sparse error. Principal Component Analysis can be computed using an SVD singular values truncation procedure which similarity reduces the rank of the matrix. Sparse Recovery is not intended for rank reduction, but taking into account that posteriors should ideally by sparse vectors, thus, the projection achieved by sparse recovery will tend to sparsify the posteriors and reduce the rank of matrices formed by posteriors of the same class. A brief review of those techniques is stated in this section.

## 4.1 Principal Component Analysis

Principal Component Analysis (PCA) is a powerful tool to perform analysis and denoising of high-dimensional data when the structure is unknown. Relying on singular values decomposition (SVD), PCA performs a linear transformation of the data. Let $X$ be the matrix containing the observed samples PCA aims at finding $P$ such that

$$Y = PX \tag{4.1}$$

where $Y$ is the new representation of $X$ over the new set of basis $P$ that expresses the samples in a better way. An explanation of PCA is proposed by [52]. This better expression of the samples can be represented by the covariance matrix of $X$. By setting $C_X$ as the covariance matrix of $X$ and $C_Y$ as the covariance matrix of $Y$ PCA aims for finding $C_Y$ such that its off-diagonal terms are zeros (meaning that $Y$ is fully decorrelated). PCA use singular value decomposition to achieve this goal. Let $E$ denote the matrix containing the eigenvectors of $C_X$ as the columns then the following equations hold. If $P$ is chosen such that each row is an eigenvector of $XX^T$ the following equation holds.

$$\begin{aligned} C_Y = \frac{1}{n}YY^T &= \frac{1}{n}(PX)(PX)^T \\ &= P(\frac{1}{n}XX^TP^T) \\ &= P(C_X)P^T \\ &= PP^TDPP^T = D \end{aligned} \tag{4.2}$$

where $D$ is a diagonal matrix. The last step holds due to the fact that $P^T = P^{-1}$ which is proven in [52]. The eigenvectors of $XX^T$ are called principal components of $X$. To obtain a matrix with reduced rank by using PCA, SVD can be used directly [52]. The procedure consists of two steps:

1. Scale the data points by subtracting the mean of the points.
2. Apply SVD on this new matrix.

A dimensionality reduction can be achieved by forcing the smaller singular values to zero up to the desired rank.

This method is straightforward and no parameters are required. The real problem is that the data is assumed to be Gaussian and the axis to be orthogonal. The orthogonal assumption comes from the orthogonal projection that PCA does. The Gaussian assumption is slightly different and comes from the covariance matrix. If the covariance of two Gaussian random variable is zero then they are not correlated (by definition of the correlation) and in addition, they are also independent. This property only holds for Gaussian distributions and thus, it is needed to be sure that $C_Y$ is decorrelated.

## 4.2 Robust-PCA

Robust Principal Component Analysis (RPCA) is a robust version of PCA. Given a matrix $D \in R^{m \times n}$ and $r \ll min(n, m)$, the dimension aimed after reduction, PCA can be seen as the following minimization.

$$\min_{A,E} \|E\|_F, \text{ s.t. rank}(A) \leq r,\ M = A + E \tag{4.3}$$

where $E$ represents the noise matrix and it is assumed to be i.i.d and Gaussian. However, RPCA proposes a slightly different minimization criteria and aims not to reduce the dimensionality of the representation by "truncating" a certain number of components but tries to reduce the rank of the given matrix $M$ through

$$\min_{A,E} \text{rank}(A) + \lambda\|E\|_1 \text{ , s.t. } M = A + E \tag{4.4}$$

The rank function is relaxed to the nuclear norm to obtain a convex cost function for optimization to get (4.5).

$$\min_{A,E} \|A\|_* + \lambda\|E\|_1 \text{ , s.t. } M = A + E \tag{4.5}$$

where $\|.\|_*$ denotes the nuclear norm that is the sum of the singular values of the matrix. It was proven that RPCA perform well when the data structure suggested by the matrix $M$ lies on a single subspace [30]. Implementation of RPCA used for this work has been done by Minming Chen and Arvind Ganesh, copyright: Perception and Decision Laboratory, University of Illinois, Urbana-Champaign[1] which make use of the Augmented Lagrange Multiplier (ALM) method to solve the convex optimization problem.

## 4.3 Low-Rank Representation

If the data matrix $M$ does not represent a single low-rank subspace (but a combination of many low-rank subspaces for example) one can use the Low-Rank Representation (LRR) algorithm to extract noise from the matrix. LRR proposes a slightly different minimization function than RPCA which is

$$\min_{Z,E} \text{rank}(Z) + \lambda\|E\|_l \text{ , s.t. } M = DZ + E \tag{4.6}$$

where $D$ is a dictionary that linearly spans the data space. $l$ denotes a various choice of norm that can be used. The rank function will also be relaxed to the nuclear norm. As indicated by its name, LRR propose to find a low-rank representation of the data $M$ using the dictionary $D$ while

[1] `http://perception.csl.illinois.edu/matrix-rank/sample_code.html`

extracting sparse error. The $l_1$ norm that defines the sparse error can also be changed to $l_{21}$ norm and will enable removal of outliers. Others variants are developed to overcome under sampling problems [31]. The code used in this project is taken from LRSLibrary[2]. The code itself is part of the work of Zhouchen et al. [29].

## 4.4 Sparse Reconstruction

By using sparse techniques, it is also possible to decrease the rank of matrices. Posterior probability vectors lie in a low-dimensional space. Using dictionary learning and sparse recovery, it is possible to project the data into the space of the dictionary. Since the dictionary is learned to represent the space of posteriors, sparse recovery enables projection into this low-dimensional space, thus reduces the rank of posterior subspaces. The following sections review the procedure of sparse recovery to achieve this goal.

### 4.4.1 Dictionary Learning for Sparse Representation

Consider a set of observations $X \in R_n$ and $D$ a dictionary (usually an over-complete basis) that spans the set of observations. The sparse coding of a vector $x \in X$ over $D$ aims to find a sparse vector $\alpha$ (with only a few coefficients that are non-zeros) such that $x = D\alpha$. The solution of such a problem can be find by solving the following equation

$$\arg\min_{\alpha} \; \|x - D\alpha\|_2^2 + \lambda\|\alpha\|_0 \tag{4.7}$$

where $\lambda$ is a constant. Because of the non-convex nature of the $l_0$ norm the equation is relaxed to the $l_1$ norm. The main problem with such an approach is always to find a reliable $D$ that spans the data. Given a space $N$ such that $dim(N) = n$, an over-complete basis $D$ must satisfy the following properties.

1. $\forall x \in N, x = \sum_{i=1}^{k} a_i D_i$, for $D_i$ a entry of $D$
2. $k < N$

Using an over-complete basis might not immediately seem useful. On the standard case of a non-noisy matrix one only needs the same amount of vectors as the dimension of the space to cover all possible linear combination and thus span the whole sparse. As the basis is now over-complete each x has no more unique representation in the basis $D$. One must so force the algorithm to take only a few components to form the linear combination. This action of taking "just a few" is called sparsity enforcement. To achieve a sparse linear combination of the vectors a new criterion has to be introduced. The over-complete basis and the coefficients for each projection have to be learn in parallel. This leads to solving the following equation.

$$\min_{a_i, B_i} \|x - \sum_{i=1}^{k} a_i B_i\|_F^2 + \lambda \sum_{i=1}^{k} |a_i| \tag{4.8}$$

Solving such an equation in parallel in not feasible and need a smarter approach. Such an algorithm is proposed by [32] and its pseudo code is given by Algorithm 2 taken from [15].

### 4.4.2 Sparse Recovery

Given an observed vector feature $y$ and a dictionary $D$ that supports $y$, the idea of sparse-recovery is to find the projection $x$ such that

$$\min_{x} \|x\|_0 \text{ , s.t. } Dx = y \tag{4.9}$$

[2]`https://github.com/andrewssobral/lrslibrary`

**Algorithm 2:** Online Dictionary Learning Algorithm

```
Input  : $Z = [z_1, ..., z_t] \in \mathbb{R}^{k \times m}$ the input vectors, $\lambda \in \mathbb{R}$ a regularization parameter and an
         initial estimate for the dictionary $D^{(0)} \in \mathbb{R}^{k \times m}$
Output : $D^{(t)}$ dictionary f
1 for $t \leftarrow 1$ to $t$ do
      /* Sparse Coding of $z_t$ to get $\alpha_t$ */
2     $\alpha_t := \arg\min_\alpha \{\frac{1}{2}||z_t - D^{(t-1)}\alpha||_2^2 + \lambda||\alpha||_1\}$;
      /* Updating $D^{(t)}$ with previous $D^{(t-1)}$ */
3     $D^{(t)} := \arg\min_D \{\frac{1}{t}\sum_{i=1}^{t}(\frac{1}{2}||z_i - D\alpha_i||_2^2 + \lambda||\alpha_t||_1)\}$;
```

Because of the non-convexity of $l_0$ norm (4.9) is often relaxed to $l_1$ norm which gives

$$\min_x \|x\|_1 \text{ , s.t. } Dx = y \tag{4.10}$$

Subject to the following conditions, it is proven [70] that the solution $x$ to the upper problem is unique. Let $I = \text{supp}(y)$ and $s = \text{sign}(y_I)$

1. $A_I$ is full column rank.
2. It exist v such that $A_I^T = s$ and $\|A_I^T\|_{Inf} < 1$.

## 4.5 Summary

The proposed low-rank representation methods, namely, PCA, robust-PCA, LRR, and sparse reconstruction will be used to achieve posteriors enhancement as suggested by Section 3.5. PCA and robust-PCA are mainly focused on single subspace cases while LRR is designed for multi-subspace rank reduction. Sparse reconstruction will be used in Section 6.1.1 to produce enhanced sparse coded posteriors using dictionary learning and sparse recovery. Furthermore, clustering techniques, detailed in Appendix A, will be needed to achieve a multi-subspace enhancement.

The next section proposes the evaluation criterion to estimate the quality of the posteriors after rank lowering as well as their impact on the task of automatic speech recognition.

Section 5

# Evaluation Procedure

Experimental analysis must rely on established methods to evaluate the effectiveness of a given system. In this section, the metrics and setup for the experimental evaluation of posterior acoustic models and ASR are described.

## 5.1 Posterior Probabilities

Two main approaches are used to compute the quality of a given set of posteriors probabilities. Relying on the low-rank hypothesis, the rank of a matrix formed by an aggregation of posteriors belonging to the same class is used as quality indicator. To compute the rank of a matrix of posteriors, which are often corrupted with sparse noise, an approximate rank computation is needed. Furthermore, the quality of the posteriors also depend on their ability to be accurate estimates of posterior probabilities. For that reason, the diagonal plot is presented in this section.

### 5.1.1 95-V Rank

The "95-V rank" is a method to compute an approximate rank of a matrix of low-rank components contaminated with sparse noise. The "95-V rank" of a matrix is indicated by the number of singular values to keep so that 95% of the matrix variability in term of Frobenius norm is keep. The matrix produced by keeping the $k$ highest singular values of the matrix $X$ is denoted by $X_k$. The method finds the smallest $k$ that satisfy the following equation:

$$\min_k \frac{\|X - X_k\|_F}{\|X\|_F} < 0.05 \tag{5.1}$$

The procedure for solving this problem is summarized in Algorithm 3. It is possible to change the value of 95% to any other value. This values indeed depends on the quality of the given matrix as well as the level of sparse noise present.

### 5.1.2 Diagonal Plot of MAP vs. Classification Accuracy

The diagonal plot proposed by Section 2.3 describes the quality of a given set of posteriors in two ways

1. The more the posteriors are confined to the right of the plot the better their underlying quality is. While lying to the right side of the plot, posteriors tend to be classified correctly more often.

2. If the plot produced by this correspondence tend to follows the diagonal then it means that they are accurate estimates of the posterior probabilities.

The class label for each posterior are obtained using HMM forced alignments. This test is important to examine if any method modifying posteriors still gives a good approximation of posterior probabilities.

**Algorithm 3:** Pseudocode to compute the "95-V rank" of a matrix

```
   Input  : Noisy matrix $X$
   Output : $k$ "95-V rank" of the matrix $X$
1  $a$=number-of-rows($X$); $b$=number-of-columns($X$)
2  Initialization: $k$=0; $X_k$=zeros(a,b); $\hat{S}$=zeros(a,b)
3  $X$=log($X$+eps)
4  [$U$,$S$,$V$]=svd($X$)
5  while $\frac{\|X-X_k\|_F}{\|X\|_F} < 0.05$ do
6      $k$+=1
7      $\hat{S}(k,k)$=$S(k,k)$
8      $X_k = U\hat{S}V^T$
9  return $k$
```

## 5.2 Automatic Speech Recognition

The database as well as the setup for the DNN-HMM hybrid is presented in this section. In addition, a quality indicator for the accuracy of the transcriptions obtained using the DNN-HMM hybrid is computed using the Word Error Rate (WER).

### 5.2.1 Speech Database

An important aspect of the evaluation procedure is the database used to judge the quality of the methods. The experimental evaluation of this report is based on the Numbers'95 database [12]. More particularly on a subset of it which consists only of the digits from *"zero"* to *"nine"* also including *"oh"*. This set will be called for further references the Digits database. This leads to a total of 11 distinct words plus silence. The data set is multi-speaker, and it is divided into 3 different sets: training set (51'311 words) used to train the DNN, development set (17'916 words) used as cross validation to fine-tune the DNN and test set used for speech recognition (13'967 words).

### 5.2.2 DNN Architecture

A DNN has been used to estimate the posterior probabilities acoustic models. Its architecture consists of three hidden layers with 1024 nodes. It is trained to estimate both context depend and context independent posteriors by using respectively 557 tri-phone tied states (senones) and 27 phones probabilities.

The number of tri-phones states can reach up to 18252 (26×27×26), and has been reduced by Kaldi algorithms to 557 tied-states (Section 3.3.3).

The input features of the DNN consist of MFCC features plus $\Delta$ and $\Delta\Delta$ by making use of a context of 9 frames. The window size was set to 10ms. This leads to an input layer of dimension 351 (39×9) and output dimension of either 557 (senones) or 83 (states). The phone posteriors are obtained by mapping every 3 exclusive states to a phone to obtain phone posterior probabilities. Specific Kaldi commands which allows to get context independent states representing high level phones, details can be found in Kaldi webpage[1]

The DNN trained using Kaldi toolkit (Section 1.4.3). Phone posterior probabilities were inferred using state-of-the-art method relying on state posteriors re-estimation.

### 5.2.3 ASR Word Error Rate

The Word Error Rate (WER) of an ASR system is one of the most important metric to measure its effectiveness. Given a reference transcription $T_R$ of $N$ words and a decoded transcription $T_D$ the goal is to know how well $T_D$ estimate $T_R$. It is possible to compute via dynamic programing

[1]see `http://kaldi-asr.org/` for details.

the number of insertions $I$, deletions $D$ and substitutions $S$ made by $T_D$ given the reference. The WER is then defined as

$$\text{WER} = \frac{D + S + I}{N} \tag{5.2}$$

This measure simply describes the edit distance between $T_D$ and $T_R$. Even if this measure is not the only criterion for ASR quality, it is still a decent indicator [64].

## 5.3 Summary

In this section, methods for evaluation of the posteriors quality ("95-V rank" and the diagonal plot) and a method for automatic speech recognition evaluation (WER) have been detailed. The "95-V rank" allows to compute an approximate rank of a noisy matrix and the diagonal plot gives a visual feedback of the quality of posterior probabilities. The impact on the ASR system of the enhanced posteriors can be estimated using the WER, to compute the effectiveness of the posteriors, and the McNemar significance test (see Appendix B), to compare results yield by different systems.

The database and the DNN architecture have also been presented. All those methods aims to prove that the proposed method lead to posterior enhancement and thus ASR improvement.

The next sections, Section 6 and Section 7, present the experimental results conducted on respectively class-specific and multi-subspace acoustic models.

# Section 6

# Experiments on Class-Specific Enhanced Acoustic Models

In this section, various metrics are used to evaluate the quality of posterior probabilities including the entropy, correlation, MAP accuracy, as well as rank reduction. Analysis on the rank of matrices of posterior vectors belonging to the same class is found to be a strong indicator of their quality. It is found that as the rank of the class-specific matrices is decreased, the ASR accuracy is improved.

## 6.1 Analysis of Sparse Reconstructed Posteriors

Earlier work has shown that sparse reconstruction leads to enhanced posterior probabilities, and higher ASR accuracy [16]. Hence, different analysis are conducted to compare the DNN posteriors with the sparse reconstructed enhanced posteriors. Deep investigations help better understanding of the properties of posterior vector space, and quantify their accuracy for ASR acoustic models.

### 6.1.1 Projected Posterior Probabilities

The procedure to obtain the sparse reconstructed posteriors is as follows. First, the posterior probability vectors from the training set are used to learn dictionaries for sparse representation of each senone posterior class $D_1, \dots, D_m$. Then, sparse representation of the test posterior vectors are computed using group sparse recovery over the aggregation of all dictionaries $D = [D_1, \dots, D_m]$. The test posterior is then reconstructed using sparse representation vector $\alpha$ and the dictionary $D$, as $D\,\alpha$. This sparse reconstruction enables projection of the test posteriors in the space of training posterior vectors. This procedure is illustrated in Figure 6.1. The algorithms are briefly described in Section 4.4.1; more details can be found in [16].

### 6.1.2 Analysis of Entropy

The first analysis carried out was on the entropy of the projected posteriors. The idea was to perform the same analysis as in [2] to verify if any variation of the entropy could give an explanation of this ASR improvement. Entropy is a good measure of the error / uncertainties that may be present in the posterior probability vectors. Hence, the entropy of each posterior probability vector for both DNN and projected posteriors were computed.

The posteriors are processed in two groups: correctly and incorrectly classified posteriors based on matching MAP labels with their ground truth labels obtained from HMM forced alignments. The results are illustrated in Figures 6.2 and 6.3

The entropy seems to increase slightly for projected posteriors. However, the overall number of high entropy posteriors are reduced. This behavior does not give a clear explanation of the acoustic modeling improvement. Slightly higher entropy may be the result of the cost function used for dictionary learning. Revision of the cost function, enabling sparse and low-entropy representation and projection, has to be investigated.

$$\begin{array}{ccccccc}
z & & & \alpha & & D\alpha \\
\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix} & \xrightarrow[\text{Recovery}]{\text{Sparse}} & \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,N} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m,1} & d_{m,2} & \cdots & d_{m,N} \end{pmatrix} & \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix} & \xrightarrow[\text{on } D]{\text{Projection}} & \begin{pmatrix} D\alpha_1 \\ D\alpha_2 \\ \vdots \\ D\alpha_m \end{pmatrix} \\
\text{DNN posterior} & & D = [D_1, D_2, \ldots, D_m] & \text{Sparse representation} & & \text{Projected posterior}
\end{array}$$

**Figure 6.1:** *Projection of test posteriors into the low-dimensional space of training posteriors using sparse recovery. In the first step, class-specific dictionaries $D_1, D_2, \ldots, D_m$ are learned for sparse representation of training posteriors. An aggregated dictionary will be formed using all class-specific dictionaries as $D = [D_1, D_2, \ldots, D_m]$. Sparse representation $\alpha$ of test posteriors are obtained through group sparse recovery that takes into account the internal partitioning structure of the dictionary. Finally, the projection of test posteriors is obtained via the reconstruction $D\alpha$.*

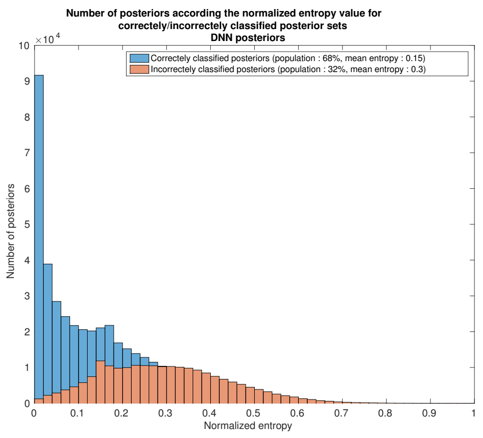


**Figure 6.2:** *Entropy of DNN posteriors.*

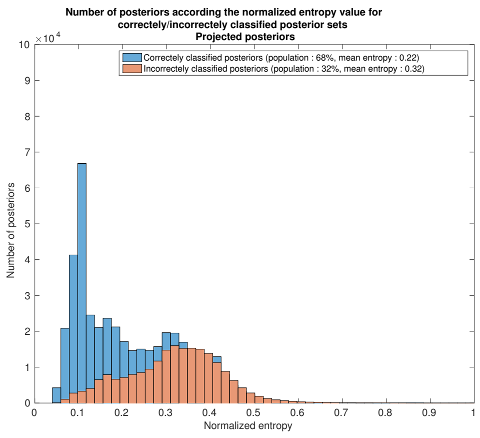


**Figure 6.3:** *Entropy of projected posteriors.*

### 6.1.3 Analysis of Correlation

The second analysis was performed on the correlation between senone posterior classes. The hypothesis is that the correlation between posteriors of different classes is reduced, thus, classification of posteriors becomes more accurate. In order to compute a correlation between all senone classes a representative of each senone class was computed as an average vector. All the posteriors belonging to a specific senone class were extracted and the average representative is computed as the mean of all posterior vectors.

The correlation between all the senone representatives is computed using Pearson product-moment correlation coefficient and an average is computed for all the senone classes. Pearson coefficient was chosen as it is the most usual coefficient used for random variables comparison. Table 6.1 shows the average correlation coefficients computed. The results show that the projected posteriors seem to have a slightly higher correlation between the classes, although the difference is substantially small.

| Method | Average correlation |
|---|---|
| DNN posteriors | 0.010 |
| Projected posteriors | 0.015 |

**Table 6.1:** *Average correlation between all senone classes for DNN and projected posteriors.*

### 6.1.4 Analysis of MAP Accuracies

The third analysis is performed on the probability correspondence of the posteriors. The theory of posterior probability as stated in Section 2.2 asserts that each posterior vector ideally follows a probability correspondence rule. Namely, the value of the MAP should indicate the classification accuracy of the posterior vectors. This rule is shown in Figure 6.4 and 6.5 respectively, both describing the behavior of the DNN and projected posteriors.

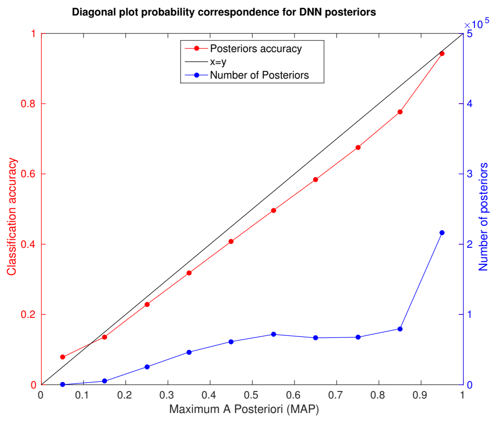


**Figure 6.4:** *Diagonal plot of MAP vs. classification accuracy for DNN posteriors.*

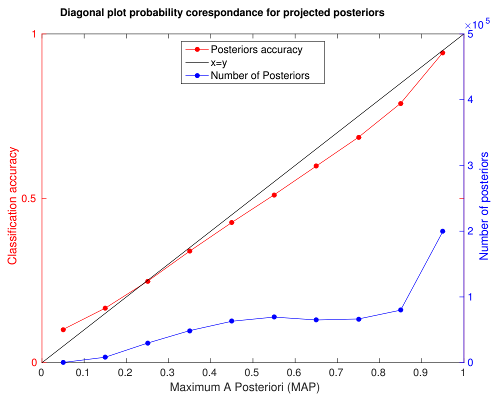


**Figure 6.5:** *Diagonal plot of MAP vs. classification accuracy for projected posteriors.*

By computing the squared error between the computed classification accuracies and the optimal diagonal line, it is possible to estimate which curve fit the best. The results are listed in Table 6.2, projected posteriors fit slightly better to the optimal diagonal line.

| Method | Squared error |
|---|---|
| DNN posteriors | 0.0023 |
| Projected posteriors | 0.0016 |

**Table 6.2:** *Error between computed and expected accuracies.*

### 6.1.5 Analysis of Ranks

The fourth analysis was based on the rank of the posteriors subspaces. The hypothesis is that DNN uncertainties in the estimation of class-conditional probabilities is reduced in projected posteriors. These uncertainties lead to spurious and unstructured components in the posterior probabilities. This spurious noise can be quantified by analyzing the rank of class-specific matrices of posterior vectors.

Using directly a mathematical rank computation over those matrices is not really meaningful. In fact, the spurious uncertainties in the posterior probability vectors could easily make those matrices full rank. To alleviate this problem, a slightly different approach based on SVD decomposition (Section 5.1.1) was used. The idea is to keep a certain number of singular values and drop the lowest ones such that reconstruction with this truncated singular values preserves a certain amount of Frobenius norm of the original matrix. In this analysis, 95% variability was preserved.

It is possible to make this analysis more precise by splitting the posteriors belonging to each class into two sub-groups as correctly and incorrectly classified posteriors based on MAP. This splitting allows seeing more contrasted results. Figure 6.6 and 6.7 show the results of the "95V-rank" computation over the set of all the senone classes of the posterior probabilities. Table 6.3 also shows the results for the average of the rank over all the senone classes. This analysis clearly shows the average rank reduction by comparing the DNN and projected posteriors. Therefore,

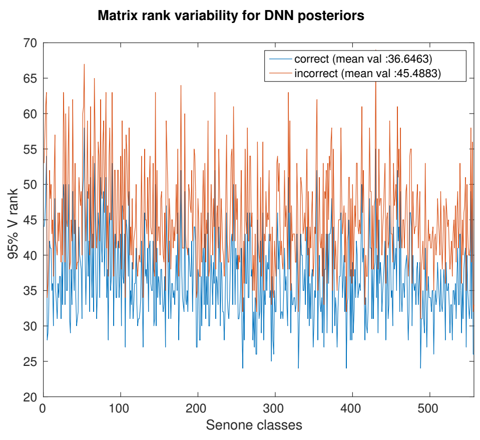


**Figure 6.6:** *"95V-rank" plot for DNN posteriors*

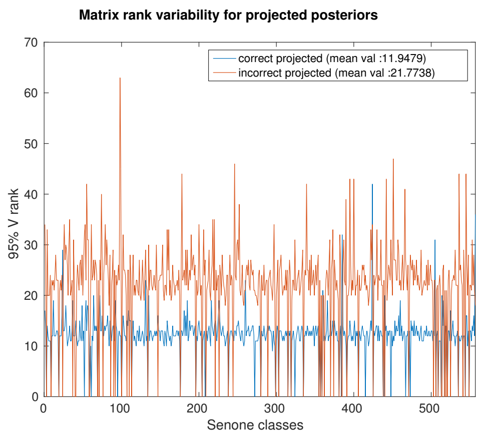


**Figure 6.7:** *"95V-rank" plot for projected posteriors*

| **Method** | **Correct rank avg** | **Incorrect rank avg** |
|---|---|---|
| DNN posteriors | 36.7 | 45.5 |
| Projected posteriors | 12.0 | 21.8 |

**Table 6.3:** *Average rank over all the senone classes for both DNN and projected posteriors and correctly/incorrectly classified posteriors .*

**to enhance the acoustic models, the rank of matrices of posteriors belonging to the same class has to be reduced.**

Further studies are conducted to enforce this observation. In particular, the use of low-rank representation algorithms to achieve enhancement of the posteriors quality in a supervised and unsupervised approaches are considered. Along with the rank analysis, the diagonal plot of MAP vs. classification accuracy is also a good indicator of the improved quality of the posteriors for ASR acoustic models.

## 6.2 Analysis of Single Subspace Low-Rank Representation of Posteriors

In this study, three low-rank representation algorithms are considered, namely, PCA (Section 4.1), robust-PCA (Section 4.2) and LRR (Section 4.3).

Senone posteriors are divided into class specific groups according to their ground truth labels obtained from HMM forced alignments. For each class, individual matrices of senone posteriors are constructed. Then, three algorithms are applied on those matrices to obtain low-rank representation. To evaluate the new posteriors,"95V-ranks" and diagonal plots are used. For computation efficiency, the matrices were processed in batches of 1000 posterior vectors. Also, PCA has been applied on log-posteriors. Table 6.4 shows the results using the three algorithms. It is clear that the more the rank decreases, the more the performance of the ASR increases.

The diagonal plots are illustrated in Figures 6.10, 6.11, 6.12 and 6.13 which confirms that the posteriors after low-rank representation are better than the baseline DNN posteriors. This is the case because most of the posteriors have been pushed to the right side of the diagonal which means that more of them are now classified correctly. A strange behavior might be observed when too small number vectors are considered for MAP estimation and classification accuracy; thus, the computed percentage of correctly classified posteriors is no more reliable. This can be observed in Figure 6.11. Using the ground truth labels for low-rank representation to achieve a better ASR is

| Method | ASR WER | Correct rank avg | Incorrect rank avg |
|---|---|---|---|
| DNN posteriors | 2.6% | 36.7 | 45.5 |
| LRR enhanced posteriors | 0.4% | 11.4 | 14.3 |
| robust-PCA enhanced posteriors | 0.4% | 7.6 | 11.7 |
| PCA enhanced posteriors | 0.3% | 4.6 | 7.9 |

**Table 6.4:** *Results in terms of ASR WER and ranks for baseline DNN posteriors and various low-rank methods (size of DNN output = 557).*

of course not a real case scenario, but it confirms that lowering the rank of posterior feature space indicates more accurate acoustic modeling for ASR.

Until now a clear distinction between the senone classes has been made. As the posteriors representing each senone class have their own subspace, single subspace rank reduction algorithms such as PCA and robust PCA perform very well. This is no more the case when dealing with mixed class posteriors in complete utterances.

Each posterior probability should ideally look like a standard basis, i.e. [00...010...0], which is not the case due to DNN uncertainties or noise. However, the posterior probability vectors lie in the low-dimensional subspaces. This intuition can be visualized by using a t-SNE plot as described in Appendix C; t-SNE is an interesting algorithm which allows 2D or 3D visualization of high-dimensional data while preserving the notion of locality in space. As shown in Figure 6.8, posteriors probabilities belonging to the same senone class form visible patterns, and it might indicate that they lie in multiple low-dimensional subspaces. In addition, the 2D distribution of posteriors according to the word they belong is illustrated in Figure 6.9. This also clearly shows the clustering of posteriors at the word level.

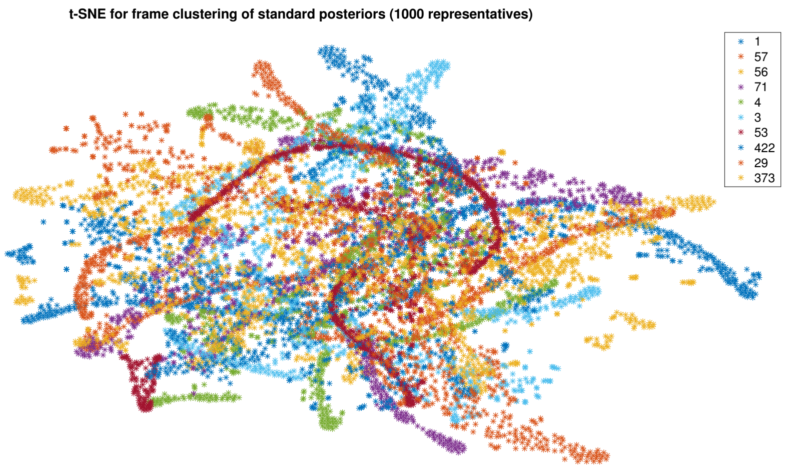


**Figure 6.8:** *t-SNE visualization of senone posterior of some classes: Each senone class has 1000 representatives plotted. Distinct patterns are visible across different classes.*

Since PCA and robust-PCA are not formulated to work with multiple subspaces [30], it was chosen to focus on LRR which has been specially designed to lower the rank of matrices that encapsulate more than one subspace [30]. After seeing that indeed the rank of posteriors was closely related to ASR improvement, the idea was to use some enhancement method directly at the utterance level and to enhance the posteriors in an on-line fashion.

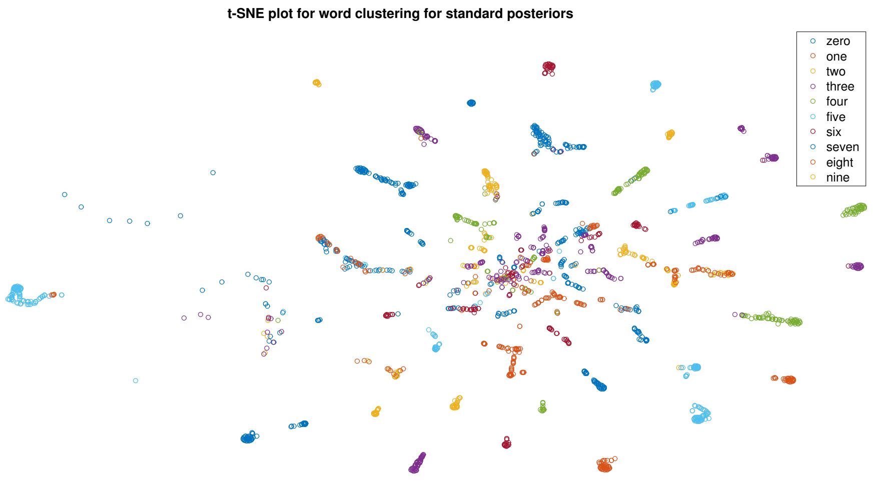


**Figure 6.9:** *t-SNE plot for DNN posteriors distinguished with different colors for each words. The posteriors belonging to the same word tend to be clustered.*

The LRR algorithm parameters have been chosen as $\lambda = 0.2$ and $D$ has been chosen as the left eigenvectors of the singular value decomposition over representatives from the training data that preserve 95% of the Frobenius norm. The details of this procedure will be explained later in Section 7.1.1.

Figures 6.10–6.13 illustrate the diagonal plot of MAP vs. classification accuracy for multi-subspace low-rank representation of posteriors. Although, a great number of posteriors are now having higher MAP accuracy, a single utterance low-rank representation can suffer from the lack of representatives to characterize the subspace of different classes. Hence, the multi-subspace low-rank representation is entangled with processing a large number of posteriors grouped together based on their subspace similarities. This idea is presented in Section 7.

## 6.3 Summary

This section has investigated the posteriors space properties. The rank of matrices of class-similar posteriors has been found a strong indicator of their quality.

Class-specific posterior enhancement was performed using various low-rank representation methods, namely, PCA, robust-PCA, and LRR. HMM forced alignments were used to define the class membership for each posteriors. The enhancement enables to reduce the WER drastically as well as the rank.

Next section proposes a multi-subspace enhancement of posteriors by making use of clustering methods. LRR parameters are also investigated.

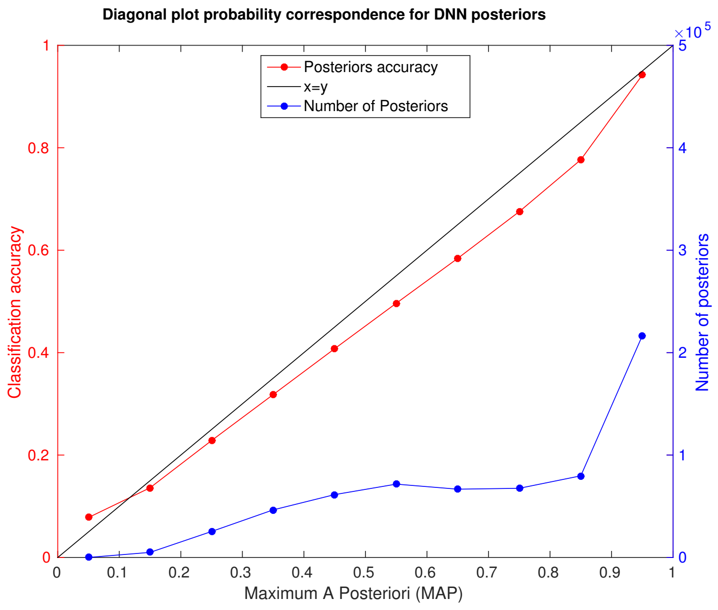


**Figure 6.10:** *Diagonal plot for standard DNN posteriors.*

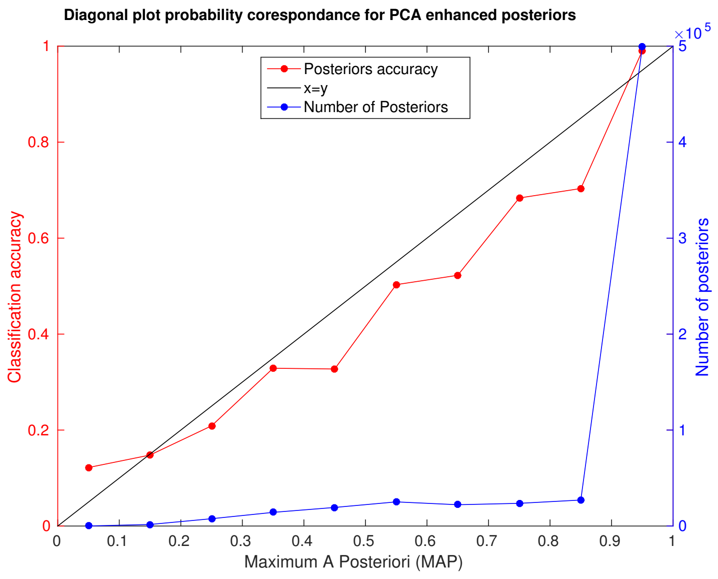


**Figure 6.11:** *Diagonal plot for PCA enhanced posteriors.*

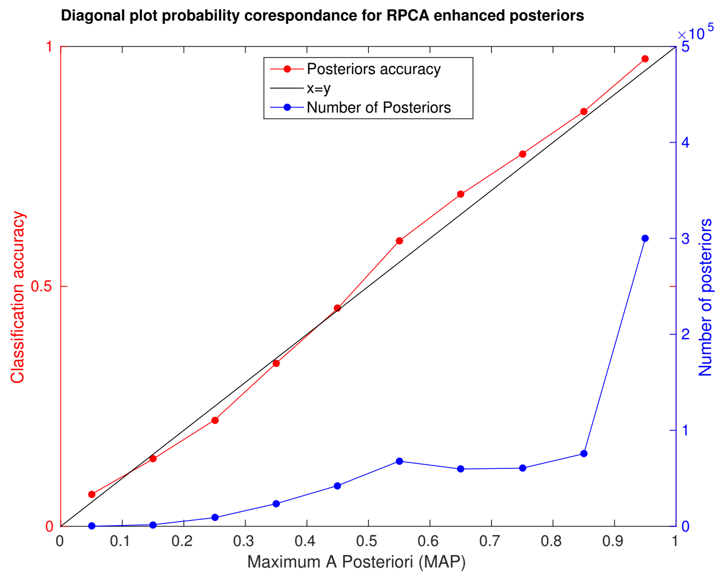


**Figure 6.12:** *Diagonal plot for robust-PCA enhanced posteriors.*

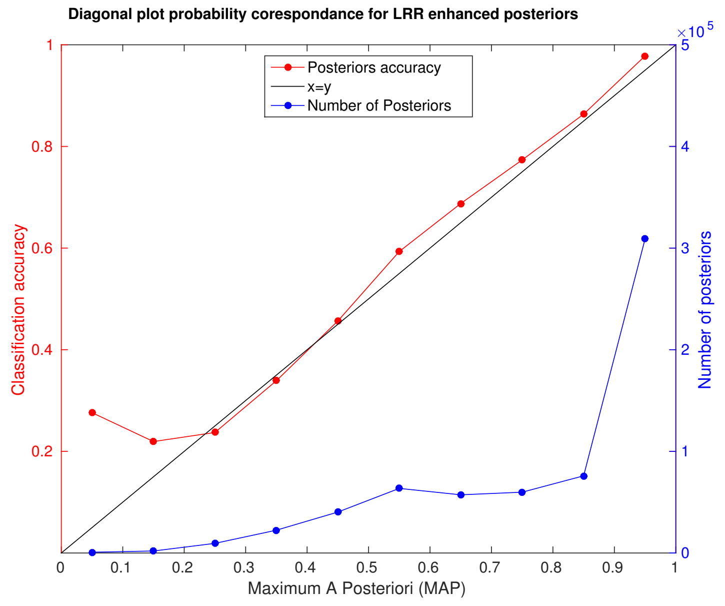


**Figure 6.13:** *Diagonal plot for LRR enhanced posteriors.*

# Section 7

# Experiments on Multi-Subspace Enhanced Acoustic Models

In this section, multi-subspace low-rank representation of posterior probabilities is investigated. The importance of low-rank representation parameters is studied followed by an incremental analysis of low-rank representation of multiple senone classes up to utterance level low-rank representation. To ensure having enough number of representatives to characterize the subspace of each class, various techniques based on classification and clustering of posteriors are considered.

## 7.1 Low-Rank Representation Parameters

Unlike class specific acoustic model enhancement, utterance level enhancement requires processing aggregation of multiple classes. Therefore, it was chosen to use LRR because of its ability to process data lying in multiple subspaces. To do so, the main parameters of LRR had to be investigated. As explained previously in Section 4.3, the LRR is formulated to obtain the low-rank representation of a matrix $M$ through the following optimization

$$\min_{Z,E} ||Z||_* + \lambda||E||_1, \text{ s.t. } M = DZ + E \tag{7.1}$$

where $D$ is a given dictionary which spans the space of data encapsulated by $M$, $E$ is called the sparse error; $Z$ is the low-rank representation of $M$ in $D$, and $\lambda$ is the regularization parameter. This gives the two parameters: (1) dictionary $D$, and (2) $\lambda$ the regularization parameter. The good choices for these parameters to enhance posteriors are investigated in the subsequent sections.

### 7.1.1 Dictionary $D$

The dictionary characterizes the low-dimensional subspaces, and accordingly the type of errors to be extracted; hence, a good choice of $D$ can improve the performance of low-rank representation [30].

The dictionary span the desired data lying in $M$ as expressed in (7.1). In addition, the sub-dictionaries characterizing each senone class should be exclusive to each senone class as explained in [71]. For that reason, it was decided to apply algorithms to the training data to extract dictionaries that are supposed to span only the desired subspaces.

PCA algorithm (Section 4.1) was used to obtain the left singular vectors of $M$. The 95V-rank procedure (Section 5.1.1) is used to choose the number of left singular vectors. The procedure consists in iterative truncated SVD reconstruction to preserve 95% variability after discarding some of the smaller singular values. The value of 95% was fixed arbitrary at the beginning of the work to compute the rank, and it was also used to compute the first set of dictionaries from the training data. Figure 7.1 shows the relation between increase in MAP accuracy and decrease in matrix rank (or increase in reconstruction error after PCA). It is clear that the maximum accuracy is achieved around 82%, and this value will also be taken to be used for the creation of “82V-basis” instead of “95V-basis”.

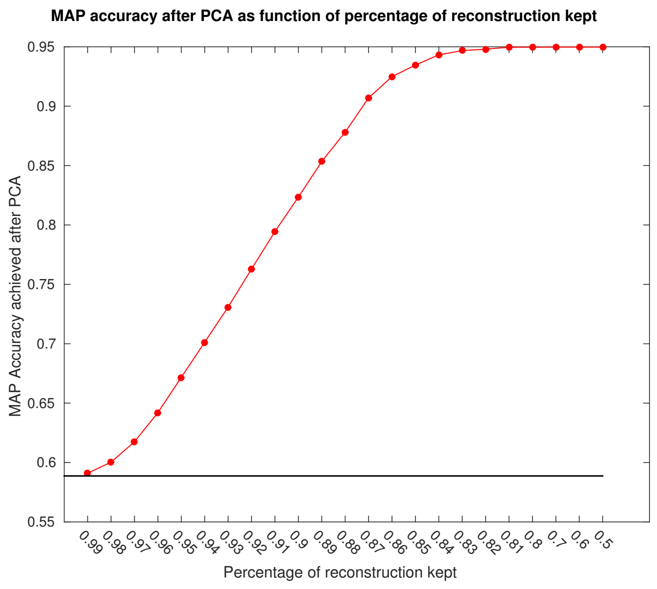


**Figure 7.1:** *MAP accuracy according to the level variability preserved after reconstruction.*

Once those values were defined, it was possible to build the dictionaries $D$ from the training data using a few left singular vectors, and compare them for LRR. The first comparison was performed between "82V-basis" and "95V-basis". Based on a subset of the development set, a fair comparison was established by comparing the MAP accuracy achieved after applying LRR on each different senone class. The parameter $\lambda$ was chosen the best overall the senone classes or the best for each senone class. For each senone class, at most 1000 representatives were used and all the results are averaged over all the senone classes. The results obtained are listed in Table 7.1, and show clearly that the "95V-basis" is better than the "82V-basis". This might be due to the fact that the "82V-basis" are really restrictive, and a small variability of posterior probability is characterized. Hence, many components are regarded as sparse noise. This also suggests that if many representative posteriors are considered, the data itself could also be used as the dictionary for LRR [30]. Hence, another test was conducted to see if indeed the basis extracted from such an

| **Mean MAP accuracy** | **82V-basis** | **95V-basis** |
|---|---|---|
| **LRR on individual senones classes** | | |
| **Best $\lambda$** | 96.2% | 91.9% |
| **Best $\lambda$ for each senone** | 96.2% | 97.1% |
| **LRR on aggregation of all senone classes** | | |
| **Best $\lambda$** | 54.6% | 71.7% |
| **Best $\lambda$ for each senone** | 72.2% | 76.9% |

**Table 7.1:** *MAP accuracy after applying LRR for both "82V/95V-basis" used as the dictionaries.*

algorithm were better than just using the matrix $M$ itself as dictionary.

For a computational reason, it was not possible to use directly the same setup as the previous test. The LRR implementation available was not able to handle 1000 representatives as input data and dictionaries. This is the reason why a smaller, but identical, test set was used to compare the procedure known as "data as dictionary" implying the use of input data as dictionaries and "95V-basis". Ten representatives for each of the 557 senone classes were randomly extracted and used to form combinations of 10, 20, 50, 100 and 250 senone classes. The results are listed in Table 7.2, and represent MAP accuracy after applying LRR with the given dictionary and various values of the regularization parameter $\lambda$.

| **Mean MAP accuracy** | **Dictionary method chosen** | | |
|---|---|---|---|
| **Number of combinations** | **Data** | **"95V-basis"** | **Aggregation of all "95V-basis"** |
| **10** | 71.2% | 89.9% | 53.8% |
| **20** | 67.9% | 86.4% | 52.6% |
| **50** | 61.5% | 78.2% | 51.1% |
| **100** | 57.3% | 68.9% | 49.2% |
| **250** | 52.1% | 58.1% | 50.8% |

**Table 7.2:** *MAP accuracy for "95V-basis" and data used as dictionary: MAP accuracy results after applying LRR with different combination of basis are presented. Each MAP accuracy is computed for various combination of senone classes and averaged over all different combinations. The best choice of regularization parameter $\lambda$ was considered.*

The best MAP accuracy was picked for each senone class. It is clear that using the data itself as dictionary performs better than using the aggregated basis. Using senone class-specific basis is, of course, the best, but in real case scenario, it is impossible to find in advance which senone classes are present or not in a given utterance. For that reason, it was chosen to discard any use of dictionaries and rely only on the self-expressiveness property of the data lying on multiple low-dimensional subspaces [30]. As stated before the dictionary should ideally be exclusive to each senone class. Using data directly is a trade-off allowing to be restricted to the set of basis which spans only the desired space where $M$ lies.

### 7.1.2 Regularization Parameter

The choice of regularization parameter $\lambda$ is proportional to the size or rank of matrix $M$. More investigation on the best $\lambda$ for different sizes has been done. Interpolation functions are used to fit an approximate function characterizing the best $\lambda$ given the size of matrix $M$. For the dictionaries, the eigenvectors corresponding to the "95V-basis" were used.

The evaluation principle is to compute the MAP accuracy after applying LRR algorithm for a wide range of length of representatives matrix from the training data (between 10 and 1000), and various values of $\lambda$s (between 0.01 to 1). The MAP accuracy is computed as an average over each senone classes for a different combination of matrix length and $\lambda$ values. The best choices of $\lambda$ result in a set of points that can be interpolated to find a function that could model as close as possible the desired function to map the matrix size to the best regularization parameter. Three types of functions have been used to fit the data. Figure 7.2 shows the results for three functions: $a + \frac{b}{\sqrt{x}}$, $a + b \log x$ and $a\frac{1}{x^2} + b\frac{1}{x} + x$. Considering the values in the range $[100, 250]$ in Figure 7.3, a big mismatch to any approximation function can be noted. Thus approximating $\lambda$ based on $a + \frac{b}{\sqrt{x}}$ may result in low accuracy. Table 7.3 shows the approximation errors for the three functions used.

| **Approximate function** | $a + \frac{b}{\sqrt{x}}$ | $a\frac{1}{x^2} + b\frac{1}{x} + x$ | $a + b \log x$ |
|---|---|---|---|
| **Mean error** | 0.0076 | 0.0140 | 0.0215 |

**Table 7.3:** *Fitting errors for three approximation functions.*

Based on above observations, a look-up table has been preferred to model the change of parameter $\lambda$ according to the size of the matrix. This might sound a brute-force approach, but this is expected to be robust to discontinuities, thus it is the strategy chosen here. Each value between the look-up table entries is selected using linear interpolation.

Furthermore, another test was conducted to see how the parameter $\lambda$ changes according to the type of dictionary being used. By using the data itself as the dictionary, a similar scenario was conducted by varying the parameter $\lambda$ and the length of the representative matrices. It is clear

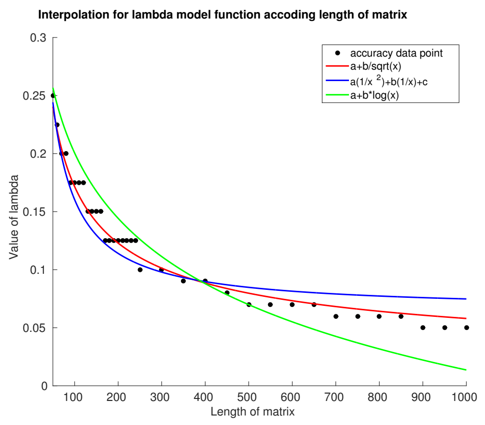


**Figure 7.2:** *Approximate function to estimate the best $\lambda$ values according to the size of posterior representative matrix.*

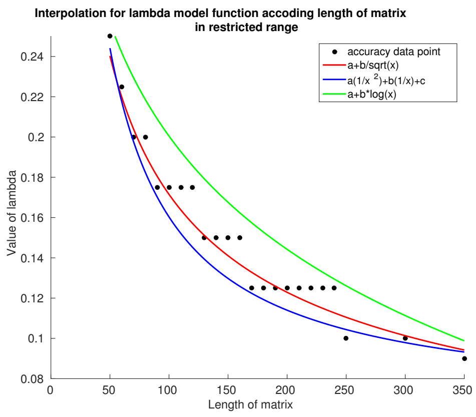


**Figure 7.3:** *Approximate function to estimate the best $\lambda$ values according to the size of posterior representative matrix in the discontinuous range* $[100, 250]$

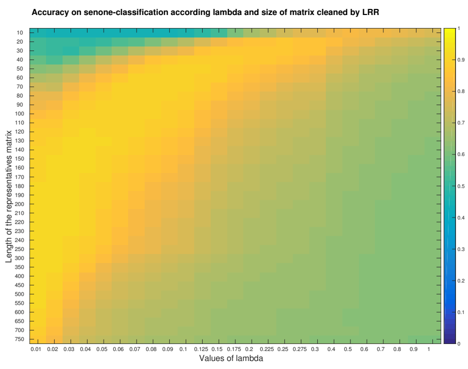


**Figure 7.4:** *MAP accuracy of senone posteriors after LRR using data itself as the dictionary for different choice of matrix size and regularization parameter $\lambda$. Yellow: 100%, blue: 0%*

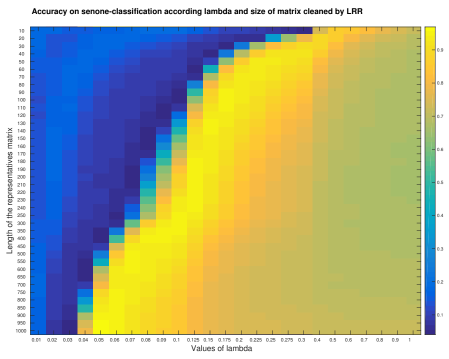


**Figure 7.5:** *MAP accuracy of senone posteriors after LRR using "95V basis" as dictionary for different choice of matrix size and regularization parameter $\lambda$. Yellow: 100%, blue: 0%*

by comparing the variation of the $\lambda$ parameter shown by Figure 7.4 and the previous one showed by Figure 7.5 that the behavior of the parameter is not the same. Again, a heuristic function approximation may not a robust model due to similar discontinuities as shown in Figure 7.3.

A final test was performed using different values of $\lambda$ and aggregation of senone basis to check the robustness of this parameter. As listed in Tables 7.4, selecting $\lambda$ according to the senone class is better than selecting an overall $\lambda$ for all the senone classes. This indicates that the optimal value of $\lambda$ depends on the properties of the underlying class, but the performance is fairly robust for a single $\lambda$ being used in LRR.

| **Mean MAP accuracy** | **DNN posteriors** | **Best $\lambda$** | **Best $\lambda$ for each senone** |
|---|---|---|---|
| **LRR on individual classes** | 50.5% | 86.3% | 97.0% |
| **LRR on aggregation of classes** | 50.5% | 76.4% | 76.8% |

**Table 7.4:** *MAP accuracy after LLR when "95V-basis" is used as dictionaries, and LRR is applied either on individual senone classes or on aggregation of all senone classes. In the former case, the best value of $\lambda$ is* 0.04 *whereas in the latter case, the best value of $\lambda$ is* 0.01.

## 7.2 Incremental Multi-Space Low-Rank Representation

The experimental analysis is performed in an incremental manner, starting with introducing more and more senone classes until the real-case scenario of on-line utterance enhancement is reached. The incremental analysis would help deep investigations on the impact of low-rank representation in a controlled setup.

The first test conducted is based on a combination of pairs of senone classes. An equal number of senones representatives are selected and combined. The LRR algorithm is applied to the combination and the results of the best MAP accuracy is shown in Figure 7.6. In comparison, Figure 7.7 was the exact same set directly at the output of the DNN, before LRR.

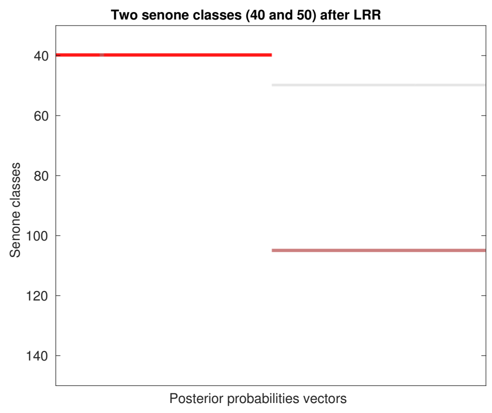


**Figure 7.6:** *Result of LRR applied on a pair of senone classes; $\lambda$ parameter chosen is* 0.01.

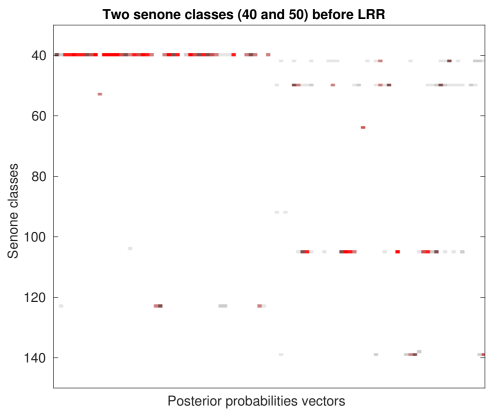


**Figure 7.7:** *Pair of senone classes directly at the DNN output.*

The baseline DNN MAP accuracy of the DNN posteriors is 42.8%. After using LRR with $\lambda = 0.01$, the MAP accuracy achieved by the enhanced posteriors is 50.0%. Indeed, LRR performs very well if the classes have evenly balanced number of representatives.

The second test conducted to see more challenging scenarios with more senone classes used for LRR decomposition. For that reason, arbitrary six senone classes with 200 representatives each were selected. MAP accuracy achieved by the algorithm increases from 47.8% to 66.5% using $\lambda = 0.01$. The achieved enhancement of the posteriors is clearly visible in Figure 7.9 with respect to Figure 7.8 which represent respectively the set of posteriors after LRR and before.

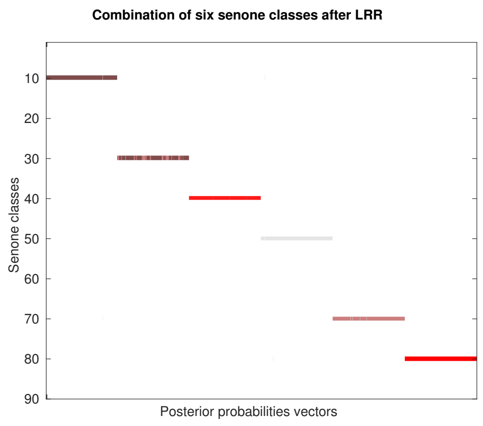


**Figure 7.8:** *LRR applied on an even combination of 6 senone classes. $\lambda$ parameter chosen is* 0.01.

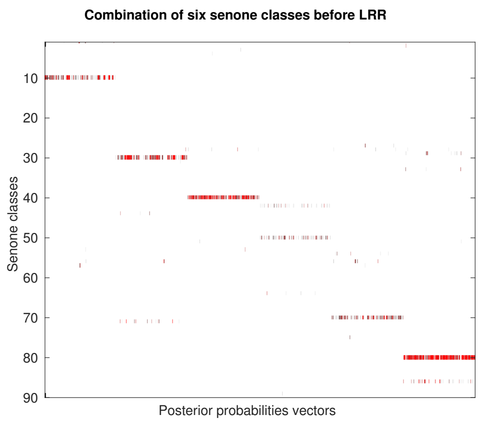


**Figure 7.9:** *Even combination of 6 senone classes before LRR.*

This scenario is not yet a real-case scenario since the numbers of senone class representatives in each utterance is typically not equal. For that reason, it was decided to introduce randomness in the choice of the number of representatives. To see the variation, it was decided to insert more and more senone classes with a randomly chosen number of representatives per senone class. Figure 7.10 shows the change of MAP accuracy for DNN posteriors and LRR "enhanced" posteriors with respect to the number of senone classes. It is possible to see a clear decrease of the MAP accuracy of the LRR "enhanced" posteriors.

A more real case scenario is to introduce an over-dominant senone class. This is typically the case of silence which is generally the most recurrent class. Figure 7.11 shows the behavior of the MAP accuracy using an over-dominant class.

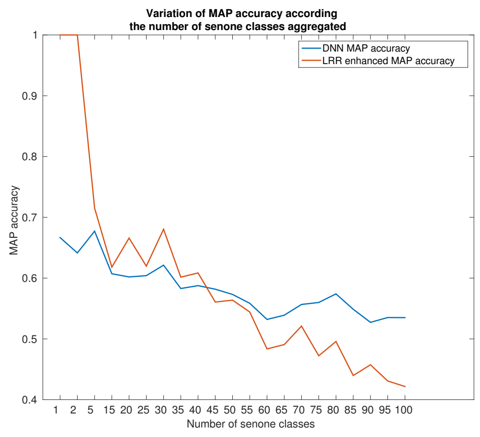


**Figure 7.10:** *MAP accuracy variation according to the number of senones with randomly taken representatives.*

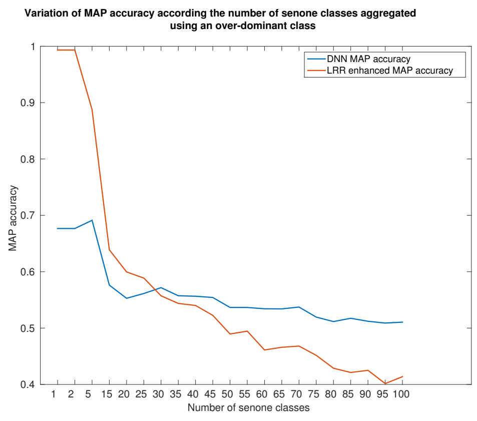


**Figure 7.11:** *MAP accuracy variation according to the number of senones with randomly taken representatives. A senone class is set over-dominant by setting its number of representatives as four times the average of the others.*

The last step was, of course, to try this algorithm directly with utterances. As an example, the biggest utterance available in the test-set is selected. LRR is then performed over a range of $\lambda$ values. The results obtained are then presented by Figure 7.12.

The small gain achieved by using $\lambda$ in the range $[1, 0.1]$, does not provide a promising evidence that LRR can enhance single utterances. Also, the value of $\lambda$ might change for each utterance due to their own senone classes distribution. By investigating smaller values, the MAP accuracy was decreased so ASR performance is also expected to decrease.

The main problem encountered is a problem of over-dominance of posterior exemplars. This is typically the case of the silence classes which are over-represented than other senone classes. As it can be seen in (4.6) bigger $\lambda$ values leads to less error separation. Hence, the low-rank results of the algorithm will tend to be exactly the same as the input matrix. On the contrary, if the $\lambda$ parameter is chosen too small, then the algorithm will tend to separate more sparse errors from the posteriors to give a representation of the lowest rank possible. This is actually done by keeping only the class which has more representatives. Such a behavior can be seen in Figures 7.13 and 7.14 which represent the utterance chosen before and after applying LRR with $\lambda = 0.01$. It is clear that the algorithm faces a problem of over-dominance. The senone classes that are being raised are the ones describing the silence.

## 7.3 Processing a Large Number of Posteriors

To overcome the problem of over-dominant senone representatives, a large number of posterior vectors were grouped using supervised classification or unsupervised clustering techniques. Two types of posteriors, namely class-independent phone and class-dependent senones are considered for this study. The following methods were investigated.

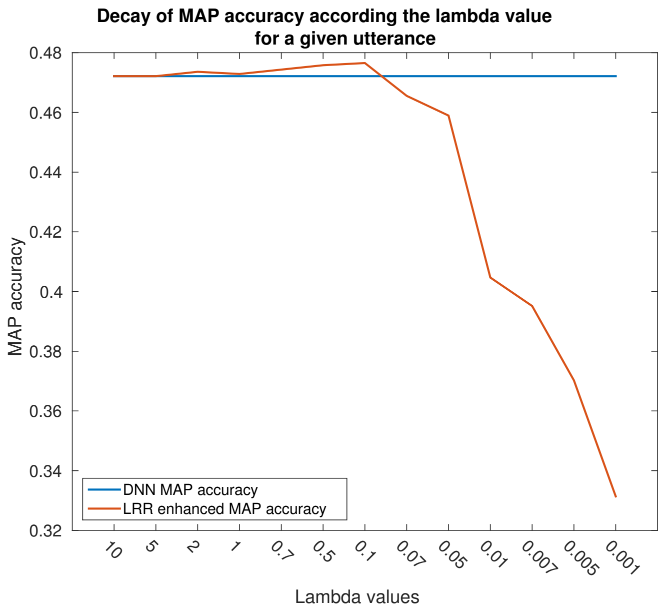


**Figure 7.12:** *MAP accuracy variation according to $\lambda$ parameter for a given utterance. The utterance is the biggest available in the test set.*

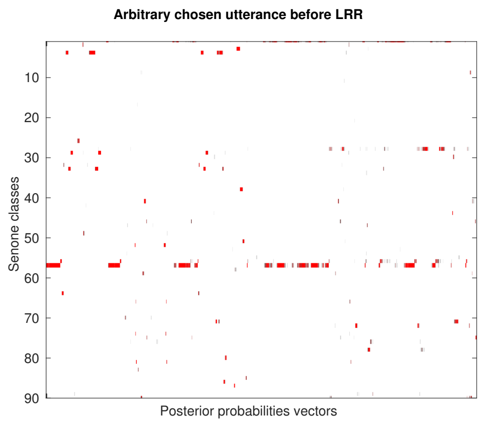


**Figure 7.13:** *Arbitrary chosen utterance before LRR.*

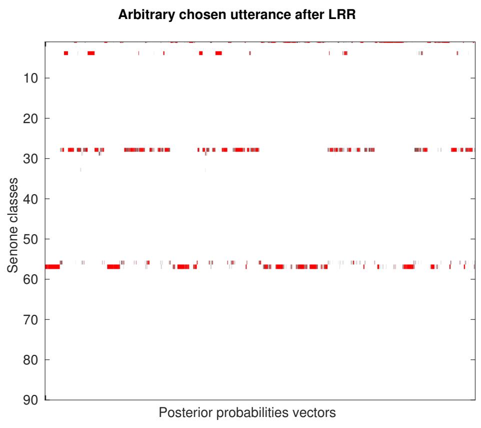


**Figure 7.14:** *Arbitrary chosen utterance after LRR. $\lambda$ parameter chosen is* 0.01*. The remaining senone classes are describing the silence.*

**MAP** - Use simply the MAP as classifier. If the DNN is not accurate about the classification of input acoustic features, MAP decision might not be the best way to classify posteriors. Hence, this classifier will be the baseline for other clustering methods.

**kNN** - It performs classification according to the majority label of k most similar vectors from the training set. The cosine distance was used as the similarity measure. This procedure is briefly described in Appendix A.1. The previous work [3] has shown the potential of kNN for accurate classification of posteriors. For phone posteriors, only a small set of training data was used to enable fast computations. For senone posteriors, both training and development sets were used.

**HMM decoding** - Alignments obtained after HMM decoding can be used to group the posteriors, and enhance them for a second pass HMM decoding.

**k-Means** - This is a well know algorithm for unsupervised clustering based on low-rank representation [6]. Its process is detailed in Appendix A.2. The cosine distance was used as the similarity measure. This choice is motivated by the analysis performed by [5] and [55]. The number of clusters k was chosen according to the ground truth.

**SSC** - This clustering algorithm is based on sparse recovery to cluster the data according to the underlying subspaces [39]. It makes use of data self-expressiveness property of data lying on a union of low-dimensional subspaces to build a similarity graph using sparse representation, and use this graph to form the clusters. This method is described in Appendix A.3.

If the classification or clustering is so accurate that single subspace posteriors are grouped together, any low-rank representation algorithm can be used to enhance the posteriors. In reality, it is expected that a mix of multiple class posteriors will group, thus, it was decided to avoid single subspace low-rank algorithm such as PCA or robust-PCA, and to focus on LRR. This was done because LRR is designed to work with multi-subspace matrices and this could be the result of classification or clustering errors.

A short experimental analysis was performed to confirm this hypothesis. Table 7.5 lists the results of the combination of PCA, robust-PCA, and LRR using two unsupervised clustering methods to produce the labels. The MAP accuracy of the selected set before enhancement is 61.3% which is close the MAP accuracy of the complete test-set. The regularization parameters of each low-rank method were tuned over the same set to produce the best results achievable. For both unsupervised clustering methods, LRR performs the best. This experiment confirms the hypothesis of LRR being able to consider erroneous labeling.

| **MAP accuracy** | **k-Means** | **SSC** |
|---|---|---|
| **PCA** | 71.2% | 74.4% |
| **robust-PCA** | 68.5% | 68.7% |
| **LRR** | 77.0% | 75.8% |

**Table 7.5:** *MAP accuracy after posterior enhancement using PCA, robust-PCA, and LRR with labels extracted using k-Means and SSC The MAP accuracy of the set before enhancement is* 61.3%*.*

## 7.3.1 Phone Posteriors Rank Reduction

The first tests conducted were based on context independent phone posteriors. The phone posteriors have lower dimension, hence, the computations were faster than senone posteriors and also required less memory[1].

Baseline system constructed using Kaldi toolkit (Section 1.4.3) using DNN posteriors yields WER of 5.9%. Results of the phone posteriors rank reduction are presented in Table 7.6. The values presented are the optimum results according to a wide range of $\lambda$ parameter values to ensure the best regularization. Also "95-V rank" (Section 5.1.1) was no more usable due to the low dimension of phone posteriors. It was decided to compute the rank using 99% variability instead of 95%. Each rank values presented in Table 7.6 are both for correctly and incorrectly classified sets of posteriors.

| | **DNN posteriors** | **MAP** | **k-Means** | **kNN** |
|---|---|---|---|---|
| **ASR WER** | 5.9% | 4.9% | 4.8% | 3.5% |
| **Rank** | 23/24 | 15/24 | 14/22 | 14/23 |

**Table 7.6:** *Results for classification and clustering to group a large number of posteriors followed by LRR for low-rank representation. Rank values are respectively for both correctly/incorrectly classified sets of posteriors.*

[1] Just for the test set, there is a total of 640586 posteriors obtained by the DNN. This lead to a total of 640586 × 27 × 10 bytes ≈ 170 Mb of data for the phone posteriors. For the senone posteriors, the total is approximately 3.5 Gb (640586 × 557 × 10 bytes).

It can be seen that kNN outperforms all the other methods and achieves a relative WER improvement of nearly 40% with respect to the baseline system. This improvement is even more impressive as it was only made possible using a small amount of training posteriors rather than all the available labeled data. Indeed, kNN exploits more effectively the DNN variability encoded in the training posteriors than k-Means and MAP. Earlier work also show the potential of kNN in the classification of posterior exemplars [3]. If no labeled data is available, unsupervised clustering based on k-Means is also beneficial to enhance the posteriors through low-rank representation.

Going more into the details of kNN results, it is possible to see how the rank reduces according to the reduction in WER. This variation is depicted in Figure 7.15 and illustrates clearly the consistent reduction of the rank and WER.

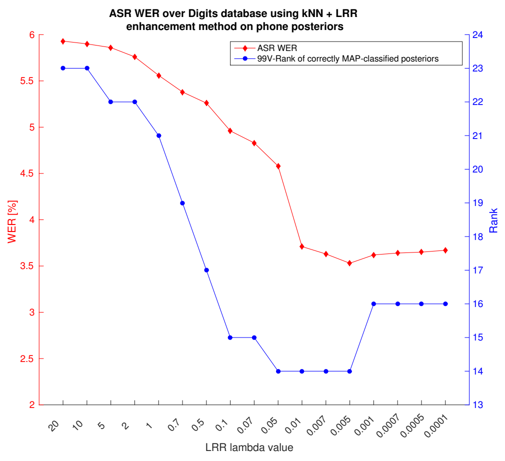


**Figure 7.15:** *Variation of the rank and WER according to $\lambda$ parameter. The rank decreases, so does WER.*

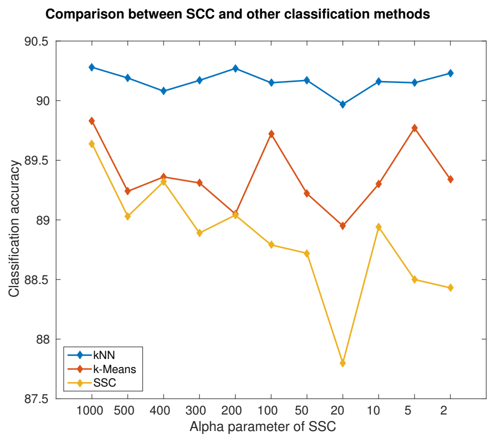


**Figure 7.16:** *Comparison of SCC, kNN and k-Means according to $\alpha$ parameter which is the regularization parameter of sparse representation of SCC. SCC does not outperform kNN.*

Another test was performed using SSC (Appendix A.3). SSC relies on self-expressiveness and sparse representation of posteriors to cluster the data according to the underlying subspaces. This algorithm exploits the subspace similarity through sparse representation, thus, it was interesting to investigate its effectiveness along with LRR to enhance posteriors.

A fast comparison was performed between SSC and kNN and k-Means using the training data. Once the test data was clustered, its labeled was determined using the majority label of the training posteriors in the same cluster. This method allows classification of the test posteriors using a clustering technique. Classification accuracy was computed over a restricted set and results are given in Figure 7.16. SSC does not outperform the alternative kNN and k-Means clustering. Again, kNN is more effective to exploit the variability encoded in the training posteriors, thus it performs better than all unsupervised methods. In addition the available implementation of SCC was extremely slow, that hinders its use on large data sets.

### 7.3.2 Senone Posteriors Rank Reduction

While the theory was confirmed on phone posteriors, it is also interesting to test it on context-dependent senone posteriors (Section 3.3.3). To have a fair comparison, the same experiments that were conducted with phone posteriors were repeated. Four clustering algorithms were considered, namely, MAP, k-Means, kNN and HMM decoded alignments.

The MAP-based clustering was first studied. LRR was computed using various values of lambda to see a global trend. The MAP accuracy of senone posteriors is 67.5%. Figure 7.17 shows the results of the experiment. It is expected that this method does not lead to any enhancement.

The second experiment was performed with k-Means clustering method. The cosine distance was used and $k$ was set using the ground truth forced alignments. By using such a composition a maximal clustering accuracy of 47.8% was achieved which corresponds to the average number of same class posteriors in the same cluster; the classification accuracy of the MAP method is 67.5% which is greater than the accuracy achieved by k-Means clustering. Given that observation, WER computation was not performed. As previously discussed MAP accuracy of the posteriors is a good measure of the posteriors quality. It is the case because the decoding of the HMM depends on the MAP accuracy to compute the optimum paths.

Furthermore, kNN computation was performed using a combination of the training and development data as labeled posterior exemplars. The computation was performed over the GRID available and took approximately two months using a MATLAB implementation. The accuracy achieve is illustrated in Figure 7.18. The biggest kNN classification accuracy is lower than the baseline MAP accuracy. Due to the trend of the curve, greater values of $k$ must be investigated to find a better classification accuracy. For the scope of this report, WER was not performed due to the relative poor quality of classification.

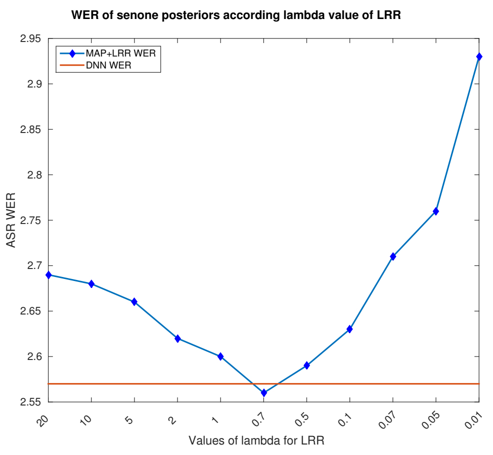


**Figure 7.17:** *WER for senone posteriors with respect to the parameter $\lambda$. DNN posteriors WER is also plotted for the comparison.*

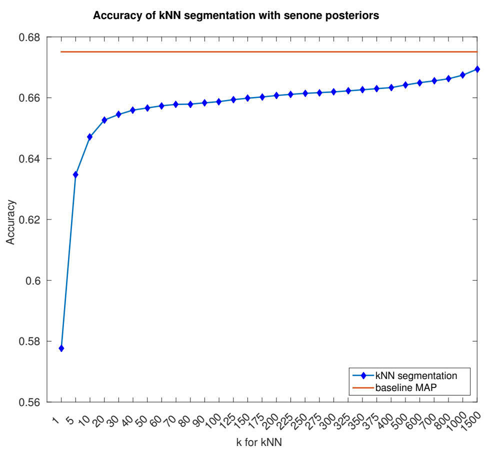


**Figure 7.18:** *Accuracy achieved by kNN clustering on senone posteriors. The segmentation accuracy is lower than the baseline MAP accuracy for the chosen $k$ values.*

Finally, HMM decoding alignments are used to cluster the same class posteriors. Using HMM alignment, a method can be developed which acts like an adaptation (through LRR) followed by a second pass HMM decoding. This idea was tested on both projected posterior (Section 6.1.1) alignment and DNN posterior alignments.

The approach is also likely to have two drawback. As the MAP classification accuracy of the senone posteriors is relatively poor, it might be possible that even this clustering method is not sufficient. Also, while using decoded alignments, it is possible that the best path decoded by HMM will be enforced, and thus somehow freezes the decoding path. To prevent this problem, a double clustering on HMM alignments was performed. Each cluster was formed using HMM alignments, and then again clustered into two sub-clusters. If the quality of the HMM decoding is sufficient, then the largest one is supposed to only contain the similar posteriors. For that reason, k-Means and SSC clustering were considered.

Classification accuracy of HMM decoded alignments are 75.9% and 80.2% respectively for DNN and projected posteriors. Tables 7.7 contains the results of this analysis. Given that the DNN baseline system achieves a WER of 2.6%, the results obtained using DNN posteriors HMM decoded alignments are slightly improved. For the case of projected posteriors HMM decoded alignments, no improvement is obtained regarding the results obtained with projected posteriors.

It was decided to perform a statistical test on the method which achieves better WER. Using HMM-based clustering with the second pass of k-Means yields a WER of 2.46%. Baseline DNN posteriors based systems WER is 2.57%. This means that this method gives a relative improvement of $\approx 4\%$. The McNemar's test (Appendix B) was chosen for this purpose. The hypothesis tested is the null hypothesis $H_0$, describing the two systems equality. The $P$-value given by this test is 0.2936. Generally taking a value of $\alpha = 0.05$, then the hypothesis $H_0$ cannot be rejected, and thus,

| **ASR WER** | **HMM-based** | **+k-Means** | **+SSC** |
|---|---|---|---|
| **LRR $\lambda$ parameter** | **DNN posteriors alignments** | | |
| $\lambda$**=0.01** | 2.5% | 2.5% | 2.6% |
| $\lambda$**=0.1** | 2.5% | 2.5% | 2.5% |
| | **Projected posteriors alignments** | | |
| $\lambda$**=0.01** | 2.2% | 2.3% | 2.3% |
| $\lambda$**=0.1** | 2.2% | 2.3% | 2.2% |

**Table 7.7:** *WER achieved using HMM-based clustering followed by LRR of senone posteriors. Both DNN and projected posteriors HMM alignments have been used. k-Means and SSC clustering methods were used. As baseline, DNN posteriors achieve* 2.6% *WER and projected posteriors achieve* 2.2% *WER.*

it is more likely that the small difference obtained is happening by chance.

Previous work showed that low-rank representation of senones leads to WER of 0.4% if the ground truth senone classes are known [16].

## 7.4 Summary

In this section, multi-subspace posterior enhancement was proposed. By making use of clustering methods followed by Low-Rank Representation, the WER given by the ASR system as well as the rank of phone posteriors were simultaneously reduced. An application of this method on query detection is proposed in Section 8.1.

Clustering and enhancement of senone posteriors were proven a difficult task. This is mainly due to the nonheterogeneous distribution of the senone posteriors. Lack of data either for DNN training or as exemplars for supervised clustering methods might be an explanation.

kNN is potential for accurate classification of posteriors if the value of $k$ is chosen properly, and effective search mechanisms can be devised. A novel hashing technique is proposed in Section 8.2 which can lead to a tremendous speedup in methods relying on posterior exemplars search including kNN.

Section 8

# Applications of Low-rank Posterior Subspace Beyond ASR

The focus of this project was on ASR. However, several other systems also benefit from higher accuracy in posterior estimation. Hence, investigating on the impact of enhanced posteriors on two different tasks were conducted: (1) query detection using posterior exemplars, and (2) fast search in posterior space. This study shows the potentials of exploiting the low-rank property in other domains relying on posterior probabilities.

## 8.1 Enhanced Posteriors for Query Detection

In this section, the results of enhanced posteriors to improve query detection are presented. Spoken query detection is a task close to ASR that is based on posterior probabilities for exemplar modeling.

### 8.1.1 Sparse Query Detection Method

Recently, a novel method for query detection is proposed at the Speech & Audio Processing group at Idiap Research Institute [45, 44]. The core idea is to use dictionary learning (Section 4.4.1) to build two dictionaries for $Q$ and $B$ where $Q$ characterizes the query exemplars, and it will be used to spot their occurrences; $B$ characterizes the background exemplars, composed of background noise and non-query spoken words. Given a posterior vector $z$, it is possible to compute its sparse representation $\alpha_Q$ and $\alpha_B$ over respectively $Q$ and $B$. The sparse reconstruction are then computed as

$$e_Q(z) = \|z - Q\alpha_Q\|_2, \quad e_B(z) = \|z - B\alpha_B\|_2 \tag{8.1}$$

The errors are used to classify whether a posterior vector belongs to the query or background. If the difference $\Delta(z) = e_B(z) - e_Q(z)$ is greater than a given threshold $\delta$ then the frame $z$ is labeled as part of the query. In opposite, if the difference is smaller than the threshold, it is labeled as part of the background. After all frames have been processed it is possible to descry if a query appears in a given test signal by counting the number of consecutive matches. If a certain number of consecutive match is reached the query is detected. This procedure is illustrated in Figure 8.1. This method differs from the standard dynamic time wrapping (DTW) based methods [48].

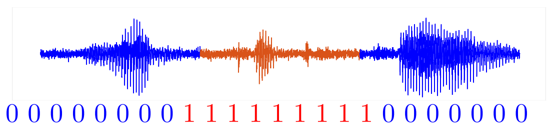
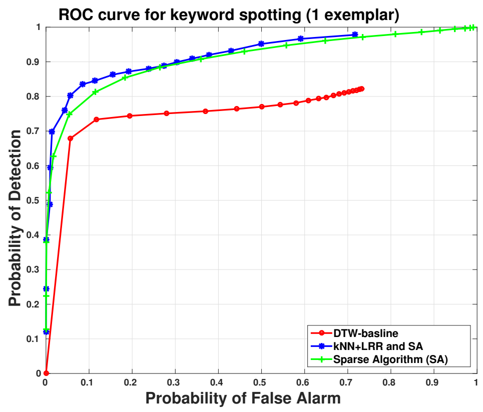


**Figure 8.1:** *Query detection by counting for consecutive frames classified using sparse reconstruction. The query is detected if a certain number of match is reached.*

### 8.1.2 Results of Enhanced Posteriors for Query Detection

The system proposed in [45, 44] uses posteriors as exemplars for dictionary learning and sparse coding. This is the reason why it was possible to use this system to check the impact of the posteriors enhanced via low-rank representation for tasks other than ASR. Similar database (Section 5.2.1) was used for this evaluations. All words are used as independent queries and the average results are presented. Figure 8.2 and 8.3 shows the results obtained by using a baseline system based on exemplar matching using dynamic time warping (DTW) [48], and the system based on sparse representation with DNN posteriors as well as the enhanced posteriors obtained through kNN classification followed by LRR.



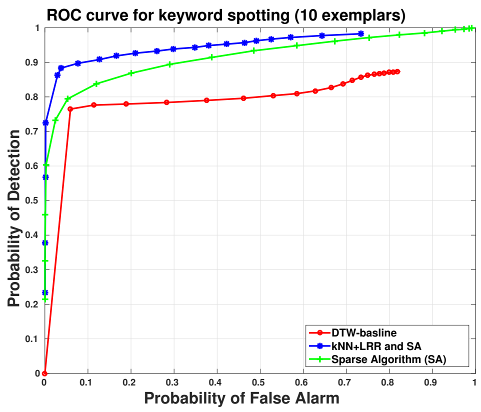


**Figure 8.2:** *Results of query detection using 1 exemplar of query for baseline DTW system, Sparse Coding System using and not the enhanced posteriors produced via kNN and LRR.*

**Figure 8.3:** *Results of keyword detection using 10 exemplars as keywords. Results for baseline DTW system, Sparse Coding System using and not the enhanced posteriors produced via kNN and LRR.*

It can be seen that enhanced posteriors outperform the baseline system as well as the sparse query detection system using DNN output posteriors.

## 8.2 Hashing for Fast Search of Posterior Exemplars

While processing high-dimensional data, efficiency problems are often encountered. Inspired by the previous work using locality sensitive hashing [57], a new procedure designed for posteriors space hashing is proposed. This procedure relies on the low-rank property of posterior subspaces, and ultimately enables very fast search in the space of posterior exemplars.

### 8.2.1 Hierarchical Hashing Procedure on Low-rank Posteriors Subspace

Taking advantage of the underlying low-rankness of the posteriors space, it is possible to design a hashing technique to divide the space into smaller size buckets of similar posteriors. It was already shown in Section 6 that posterior exemplars are sparse vectors residing in low-dimensional subspaces. Hence, for any posterior, the energy is confined to a very small number of components where the indices of high energy components identify the unique structure of the underlying subspace. The following hashing formula can exploit this structure using different levels of quantization.

$$H(z) = \frac{\lfloor 2^b z \rfloor}{2^b} \tag{8.2}$$

where $z$ is a posterior probability vector and $b \in \mathbb{N}^*$ is the number of bits for quantization.

**The main idea is that the number of quantized posteriors is far less than the training size, thus they can be used as hash keys to form buckets of similar**

**posterior exemplars. Thus, the size of the search space can be reduced to only the size of the corresponding bucket.**

Many levels of quantization can be computed to get the hash keys. While fixing a minimum size for the resulting buckets, a hierarchical hashing can be established. The greater the quantization bit will be, the more specific the buckets will become. Also, it is possible to establish a similarity search in case of missing key. A key that could not be found into the set of keys produced by an exemplar set could be compared to this given set using a similarity distance such as the dot product.

This hashing technique can be combined with kNN to enable efficient computation. Furthermore, as it is possible to compute a hash-key for each query posterior, it is possible to split the query and the exemplar sets. By assigning a bucket to each query posteriors using a decreasing order in terms of quantization bits, each sets become independent. It is possible to use parallel computing very efficiently.

### 8.2.2 Results on Fast kNN based on Posterior Hashing

To show this behavior, phone posteriors from the development set were used. Hashing was computed for quantization bits 1 up to 4. Quantization bit level 0 correspond to the whole development set without hashing. The minimum bucket size was fixed to 500 posteriors. Results are listed in Table 8.1.

| # of bits | # of buckets | Avg size | Std |
|---|---|---|---|
| **0** | 1 | 804'381 | 0 |
| **1** | 28 | 28'728 | 32'961 |
| **2** | 55 | 14'343 | 27'142 |
| **3** | 82 | 9'677 | 23'234 |
| **4** | 109 | 7'302 | 20'582 |

**Table 8.1:** *Results of posteriors exemplars hashing on Digits Database using development set. For each number of quantization bits taken, the number of buckets, their average size with standard deviation is given. Quantization bit level 0 represent the whole size of posteriors search space.*

As the number of available buckets increases, the average bucket size decrease. By using 4 bits to quantize the posterior spaces, the search space was divided by nearly 100. The standard deviation might seem quite high but comes from the uneven distribution of phone posteriors due the hierarchical procedure.

More analysis was performed on the 1-bit quantization level. It was seen that each bucket was encoding its own phone by yielding a MAP accuracy of nearly 100%. This behavior was expected due to the low-rank property of the posterior subspaces. Furthermore, the remaining buckets were encoding the DNN uncertainties.

Those results show that using this method might improve drastically the efficiency of an algorithm such as kNN which requires a search over a large space of posterior exemplars. The accuracy of the segmentation went down from 94.1% while using whole exemplars set for kNN down to 94.0% while using any level of quantization, and comparing only the posteriors hash keys to find the appropriate bucket of a local search space. As previously stated, computational speed could impressively increase due to space search reduction, and especially by using parallel computing.

## 8.3 Summary

In this section, an application of the enhancement method involving clustering and LRR to query detection is investigated. A sparse query detection method coupled with enhanced posteriors outperforms the baseline DTW based system.

In addition, relying on the low-rankness of posteriors, a hashing technique is proposed. Its impact on search in posterior space is suggested by the results achieved using kNN. The search space was reduced by a factor 100 while the segmentation accuracy was staying stable.

# Section 9

# Concluding Remarks

In this project, the low-rank property of the posterior probability subspaces was investigated. It was shown that the actual intrinsic dimension of a high-dimensional space of posterior vectors is far less than what is estimated as DNN outputs. This evidence suggests that a large number of training posteriors can be exploited to enhance the local estimates of the testing posteriors obtained from DNN.

The proposed approach leads to a hierarchical processing framework relying on posterior probabilities towards accurate acoustic models for higher level speech classification applications. The main focus of this project was to improve the ASR performance. The potential impact of the enhanced posteriors and low-rank subspace property on query detection as well as fast posterior exemplar search were also demonstrated.

One key outcome of this work is a new method to enhance the posteriors through low-rank representation (LRR) as a post-processing step which can overcome the DNN uncertainty and errors. Previous work on exploiting the low-rank representation for posterior estimation are based on restructuring the DNN using low-rank decomposition of the weight matrix [54, 37, 37, 60, 67, 50]. This approach was shown to reduce DNN complexity for a smaller footprint in portable devices, such as ASR on smart phones. The method proposed here is fundamentally different as it exploits the low-rank property in applying LRR on DNN outputs to structure the posterior probability space such that the mismatch between the training and testing posteriors are reduced. To the best of the author knowledge, this is a novel method for hierarchical posterior processing towards optimal acoustic models.

It was found that using kNN classification followed by LRR to reduce the rank of posterior subspaces leads to the best ASR performance. This can be explained by the fact that kNN effectively uses the training posterior variability to align the testing posteriors. On the other hand, if no labeled data is available, unsupervised clustering is also beneficial to enhance the posteriors according to their similarities. The unsupervised method can be interesting when limited or no resource is available to develop speech recognition for a new language, and it enables rapid adaptation. In this context, clustering methods that can exploit the underlying subspace of the posteriors [61] can be considered for LRR enhancement, although the k-means algorithm which relies on low-rank decomposition [6] is shown to be an efficient alternative.

The main feature of using LRR is that miss classification of the posteriors has low influence on enhancement due to its multi-subspace modeling capability. As suggested by [30] the dictionary used for LRR is the most important parameter. A well-chosen dictionary characterizing the space of training posteriors can lead to a better enhancement. In the present work, the PCA-based dictionary was found a promising method if the underlying subspaces are known. However, further analysis has to be conducted on multi-subspace dictionary learning. In addition, the regularization parameter can be tuned according to maximum likelihood speech recognition performance. Similarly, the cost function of low-rank representation can be tailored to maximize the discrimination between posterior subspaces or optimize the likelihood of the best speech transcription path.

This study was mainly focused on ASR performance in clean condition. It is also interesting to study the problem of ASR in mismatch condition, in particular, multi-lingual and non-native speech recognition [26, 62] where the testing posteriors are adapted to the training posteriors using low-rank

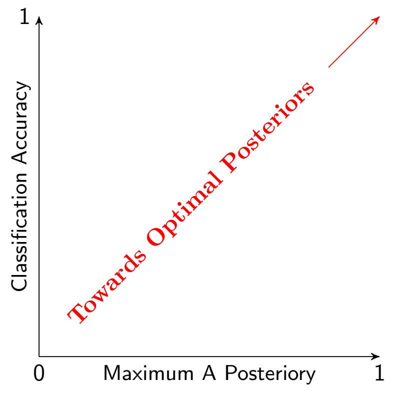


**Figure 9.1:** *Vision towards optimal speech acoustic modeling: Posterior probabilities are more accurate as they are confined towards the right side of the plot. The sparse / binary posterior vectors results in low-rank subspaces. Hence, enhanced acoustic modeling requires hierarchical processing that leads to sparser posteriors and accordingly lower rank of the individual class subspaces.*

representation. Some preliminary work demonstrated the potential of the proposed approach for query detection, and fast search over posterior exemplars. It is an interesting subject of future research to see the potential of this method for hierarchical exemplar-based speech recognition without HMM. In particular, the proposed hashing technique designed for posteriors can enable fast and effective search among a huge number of exemplars. In theory, assuming an “infinite” amount of suitable exemplars, “optimal” recognizers could be sought [14].

This project was a preliminary step towards the realization of highly accurate acoustic modeling that enable speech recognition without HMM. This vision is illustrated in Figure 9.1. The diagonal plot of MAP vs. classification accuracy was found a practical method to quantify the posteriors quality [10]. This project also proposed the underlying rank of the class-specific posterior subspaces as a strong indicator of their quality. Indeed, more accurate acoustic models are the result of posteriors which are more binary and sparser, thus, lower rank of the posterior subspaces is an immediate consequence.

Beyond ASR, other applications that rely on the estimation of DNN posteriors can also benefit from the proposed approach, such as query detection [22], speech coding [4] and linguistic parsing [11]. Through experiments on large speech corpora for a broad range of applications is planned for future research.

# Bibliography

[1] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 2015.

[2] Guillermo Aradilla. Acoustic models for posterior features in speech recognition. 2008.

[3] Afsaneh Asaei, Hervé Bourlard, and Benjamin Picart. Investigation of knn classifier on posterior features towards application in automatic speech recognition. Technical report, Idiap, 2010.

[4] Afsaneh Asaei, Milos Cernak, and Hervé Bourlard. On compressibility of neural network phonological features for low bit rate speech coding. In *Proceeding of Interspeech*, number EPFL-CONF-214698, 2015.

[5] Afsaneh Asaei, Benjamin Picart, and Hervé Bourlard. Analysis of phone posterior feature space exploiting class-specific sparsity and mlp-based similarity measure. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4886–4889. IEEE, 2010.

[6] Christian Bauckhage. k-means clustering is matrix factorization. *arXiv preprint arXiv:1512.07548*, 2015.

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[8] Nitin Bhatia et al. Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*, 2010.

[9] Hervé Bourlard and Christian J Welleken. Links between markov models and multilayer perceptrons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(12):1167–1178, 1990.

[10] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

[11] Milos Cernak, Afsaneh Asaei, and Hervé Bourlard. On structured sparsity of phonological posteriors for linguistic parsing. *arXiv preprintarXiv:1601.05647*, 2016.

[12] Ronald A Cole, Mike Noel, Terri Lander, and Terry Durham. New telephone speech corpora at cslu. In *Eurospeech*. Citeseer, 1995.

[13] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[14] Pierre A Devijver and Josef Kittler. *Pattern recognition: A statistical approach*, volume 761. Prentice-Hall London, 1982.

[15] Pranay Dighe, Afsaneh Asaei, and Hervé Bourlard. Sparse modeling of neural network posterior probabilities for exemplar-based speech recognition. *Speech Communication*, 2015.

[16] Pranay Dighe, Gil Luyet, Afsaneh Asaei, and Hervé Bourlard. Exploiting low-dimensional structures to enhance dnn based acoustic modeling in speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[17] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2765–2781, 2013.

[18] G David Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[19] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[20] Sadaoki Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(1):52–59, 1986.

[21] Laurence Gillick and Stephen J Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 532–535. IEEE, 1989.

[22] Timothy J Hazen, Wade Shen, and Christopher White. Query-by-example spoken term detection using phonetic posteriorgram templates. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 421–426. IEEE, 2009.

[23] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[24] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[25] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[26] David Imseng, Herve Bourlard, John Dines, Philip N Garner, and Mathew Magimai-Doss. Applying multi-and cross-lingual stochastic phone space transformations to non-native speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(8):1713–1726, 2013.

[27] Martin Krzywinski and Naomi Altman. Points of significance: Significance, p values and t-tests. *Nature methods*, 10(11):1041–1042, 2013.

[28] Matthew Lai. Deep learning for medical image segmentation. *arXiv preprint arXiv:1505.02000*, 2015.

[29] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[30] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):171–184, 2013.

[31] Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1615–1622. IEEE, 2011.

[32] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[35] Abdel-rahman Mohamed, Geoffrey Hinton, and Gerald Penn. Understanding how deep belief networks perform acoustic modelling. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4273–4276. IEEE, 2012.

[36] Tood K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[37] Preetum Nakkiran, Raziel Alvarez, Rohit Prabhavalkar, and Carolina Parada. Compressing deep neural networks using a rank-constrained topology. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[38] Hermann Ney. On the probabilistic interpretation of neural network classifiers and discriminative training criteria. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(2):107–119, 1995.

[39] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[40] Michael Nielsen. Neural networks and deep learning. `http://neuralnetworksanddeeplearning.com`. Last access: March 25, 2016.

[41] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[42] Lawrence Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. 1993.

[43] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[44] Dhananjay Ram, Afsaneh Asaei, and Hervé Bourlard. Sparse subspace modeling for query by example spoken term detection. Technical report, Idiap, 2016.

[45] Dhananjay Ram, Afsaneh Asaei, Pranay Dighe, and Hervé Bourlard. Sparse modeling of posterior exemplars for keyword detection. In *Proceedings of Interspeech*, number EPFL-CONF-209088, 2015.

[46] Michael D Richard and Richard P Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483, 1991.

[47] Jeff Robble, Brian Renzenbrink, and Doug Roberts. Nearest neighbor rule.

[48] Luis Javier Rodriguez-Fuentes, Amparo Varona, Mike Penagarikano, Germán Bordel, and Mireia Diez. High-performance query-by-example spoken term detection on the sws 2013 evaluation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7819–7823, 2014.

[49] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

[50] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE, 2013.

[51] Kristie Seymore, Andrew McCallum, and Roni Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[52] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[53] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[54] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3070–3078, 2015.

[55] Serena Soldo, Mathew Magimai Doss, Joel Pinto, and Hervé Bourlard. Posterior features for template-based asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4864–4867. IEEE, 2011.

[56] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[57] Vikrant Singh Tomar and Richard C Rose. Efficient manifold learning for speech recognition using locality sensitive hashing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6995–6999. IEEE, 2013.

[58] Hung-Wei Tseng, Mingyi Hong, and Zhi-Quan Luo. Combining sparse nmf with deep neural network: A new classification-based approach for speech enhancement. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2145–2149. IEEE, 2015.

[59] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[60] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-discriminative training of deep neural networks. In *INTERSPEECH*, pages 2345–2349, 2013.

[61] R Vidal. Subspace clustering. *Signal Processing Magazine, IEEE*, 28(2):52–68, 2011.

[62] Ngoc Thang Vu, David Imseng, Daniel Povey, Petr Motlicek, Tanja Schultz, and Hervé Bourlard. Multilingual deep neural network based acoustic modeling for rapid language adaptation. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[63] Ngoc Thang Vu, Jochen Weiner, and Tanja Schultz. Investigating the learning effect of multilingual bottle-neck features for asr. In *INTERSPEECH*, pages 825–829, 2014.

[64] Ye-Yi Wang, Alex Acero, and Ciprian Chelba. Is word error rate a good indicator for spoken language understanding accuracy. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 577–582. IEEE, 2003.

[65] Halbert White. *Multilayer feedforward networks can learn arbitrary mappings: Connectionist nonparametric regression with automatic and semi-automatic determination of network complexity.* 1988.

[66] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[67] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *INTERSPEECH*, pages 2365–2369, 2013.

[68] Steve J Young, Julian J Odell, and Philip C Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, pages 307–312. Association for Computational Linguistics, 1994.

[69] Dong Yu, Michael L Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide. Feature learning in deep neural networks-studies on speech recognition tasks. *arXiv preprint arXiv:1301.3605*, 2013.

[70] Hui Zhang, Wotao Yin, and Lizhi Cheng. Necessary and sufficient conditions of solution uniqueness in 1-norm minimization. *Journal of Optimization Theory and Applications*, 164(1):109–122, 2015.

[71] Yangmuzi Zhang, Zhuolin Jiang, and Larry S Davis. Learning structured low-rank representations for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 676–683. IEEE, 2013.

[72] Qifeng Zhu, Barry Y Chen, Nelson Morgan, Andreas Stolcke, et al. On using mlp features in lvcsr. In *INTERSPEECH*, 2004.

# Appendix A

# High-dimensional Posterior Grouping Methods

The goal of this project is to exploit the hidden low-dimensional space underlying a large amount of posterior probabilities in order to enhance the local estimates of the acoustic models. To that end, unsupervised clustering, as well as supervised classification techniques, are applied to cluster a group of similar posteriors. Grouping posteriors probabilities according to their underlying own subspace distributions leads to better results while applying low-rank representation methods.

## A.1 k-Nearest Neighbors Classification

k-Nearest Neighbors (kNN) is an algorithm which allows assigning a label or a class to a given query point [8]. The prior knowledge needed is a set of labeled reference points. The size of this set depends on the desired precision as well as their underlying quality. Generally, the variation of the parameter $k$ also influences the accuracy of the labeling obtained.

The method takes a set of labeled points as represented in Figure A.1. Here the query point is represented as a red diamond. A certain number is chosen for the parameter $k$. Here, $k = 10$ was chosen. The value of $k$ is usually tuned over a cross-validation set. Its influence on the precision is also important. Then a computation of a distance matrix between the query point and all the know points is performed. The k first reference points with the smallest distance are chosen as shown in Figure A.2. The query point is labeled with the most common label among those selected reference points. As usual many possible variations of the algorithm are possible. Techniques such as weighted kNN or adaptations for memory consumption are detailed in [8]. The most important observation about kNN is that if a sufficient number of samples is provided as exemplars then the convergence to the true label guaranteed [47].

In this project, kNN is used to cluster the posteriors belonging to the same class. The fact that kNN can exploit a large amount of labeled training posteriors makes this method powerful. However, it is also interesting to study the performance of unsupervised techniques which do not require labeled data.

## A.2 k-Means Unsupervised Clustering

k-Means is an algorithm which allows unsupervised segmentation of a given data set [66]. It makes use of centroid re-estimation using a distance metric between the points queried $I = \{x_i, i \in [1, n]\} \subseteq \mathbb{R}^d$. Figure A.3, based on the code of Kardi Teknomo[1], pictures the algorithmic sequence needed to estimate the clusters using k-Means. The first step is to initialize $k$ cluster centroids $\{c_i, i \in [1, k]\}$ either by giving a precomputed estimate or a random guess. It is then possible to compute for each point a temporary label according to the closer centroid using any metric needed. In general, the

[1]`http://people.revoledu.com/kardi/tutorial/kMean/index.html`

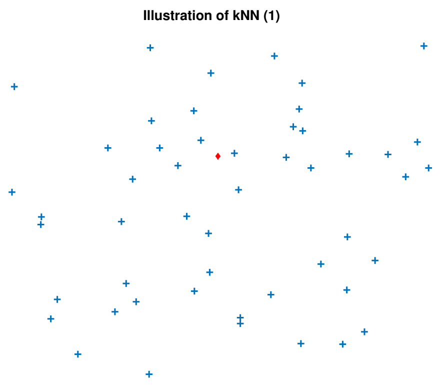


**Figure A.1:** *Random selected point in* $[0,1]^2$. *All points with blue cross are reference points for kNN. The point represented by the red diamond is the query point.*

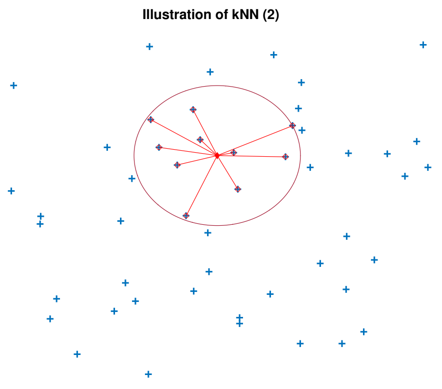


**Figure A.2:** *$k$ Nearest Neighbors of the query point. $k$ is chosen as 10 and the red circled points represent the 10 Nearest Neighbors of the query point.*

Euclidean distance is used. This leads to $k \times n$ computations to get all the labels as shown by (A.1).

$$\min_j ||x_i - c_j||_2^2, \forall i. \tag{A.1}$$

The following step is to re-estimate the centroids as the center (mean) of all the temporary clusters. If the points are associated with weights it is also possible to compute a weighted mean. This sequence is repeated until convergence of the centroids. It is important to note that if the initialization is random then the results are non-deterministic.

## A.3 Sparse Subspace Clustering

Sparse Subspace Clustering (SSC) is an unsupervised algorithmic method which allows clustering a given data set according to their subspaces [17]. It can be summarized into the following steps. Given a set of points $P = \{p_i, i \in [1, N]\}$

1. Compute a sparse representation (Section 4.4.2) for each $p_i$ given $P \setminus \{p_i\}$ as dictionary.
2. Form a graph using the sparse representation. If two points are not used for a common sparse representation then no edge connect them. The nodes of the graph are representing the $p_i$ and the weights are the coefficients of the sparse representation.
3. Make use of spectral clustering algorithm [39] to extract a segmentation of the graph which can be transformed to a segmentation of $P$. This requires an expected used specified number of clusters.

The details of this algorithm are out of the scope of this report and then omitted but available in [39].

## A.4 Summary

Three clustering methods have been presented in this section, more specifically, one supervised (kNN) and two unsupervised methods (k-Means and SSC). kNN, while used with a large enough set of exemplars, allows getting more accurate labels than unsupervised methods. On the other hand, unsupervised methods such as k-Means and SSC allows faster classification while achieving relative lower clustering accuracy. By making use of those clustering methods, posteriors probabilities will be grouped according to their class-membership. Combined with previous low-rank representation

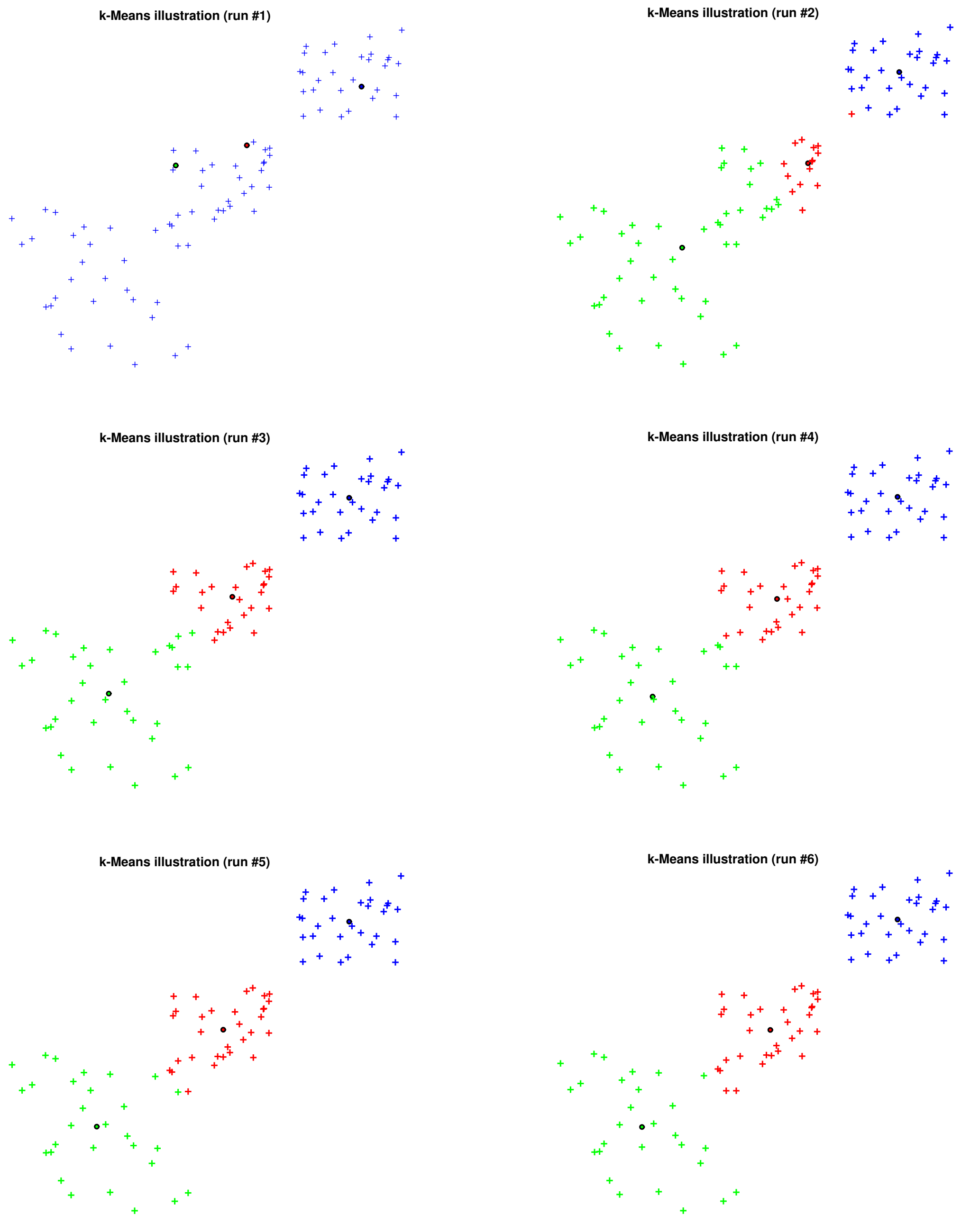


**Figure A.3:** *k-Means algorithm illustration. k-Means performed over a set of points. Each cluster is represented by a color and the centroids are represented by circles.*

methods, a multi-subspace enhancement as suggested by Section 3.5 will be possible. The proposed method can be described as a two-step procedure:

1. Cluster the posteriors according to class membership.
2. Apply low-rank representation methods to each class.

The details of this method are presented in Section 7.

# Appendix B

# McNemar's Significance Test

While developing algorithms it is important to see if minor change observed by comparison to a baseline is the results of chance or it is a significant improvement. To do so, statistical significance tests were developed [27]. Given a threshold described by a percentage of certainty needed $\alpha$ a significance test allows to reject or accept a given hypothesis.

The common hypothesis used is to claim that the two methods are equivalent or equal. This hypothesis is called the null hypothesis and its notation is $H_0$. Probability distribution functions are then modeled given observations and compared. According to the given threshold they might be equivalent or not leading to acceptance or rejection of the null hypothesis.

If the errors made are independent events then McNemar's test can be used [21]. For the case of speech recognition, independent errors mean generally that the methods that are compared are run over isolated words. This is often the case when no language model is used in the methods and when the word sequence is not relevant.

McNemar's test is described in details by [21] and can be summarized as the following procedure. If two different systems $A$ and $B$ with error-rates $e_A$ and $e_B$ respectively, the null hypothesis $N_0$ is defined as

$$e_A = e_B = e \tag{B.1}$$

Let $N_{00}$ be the number of utterances which $A$ and $B$ classify correctly; $N_{01}$ the number of utterance which $A$ classify correctly and $B$ not; $N_{10}$ the number of utterance which $B$ classify correctly and $A$ not, and finally, $N_{11}$ the number of utterance where both $A$ and $B$ results are wrong. Accordingly, the probabilities $q_{00}$, $q_{10}$, $q_{01}$ and $q_{11}$ are defined by normalizing the number of corresponding utterances. Hence, it is possible to prove [21] that

$$\begin{aligned} e_A &= q_{10} + q_{00} \\ e_B &= q_{01} + q_{00} \end{aligned} \tag{B.2}$$

and thus $H_0$ is equivalent to $q_{01} = q_{10}$ which is also equivalent to $q = \frac{q_{10}}{(q_{01}+q_{10})} = \frac{1}{2}$. Given an observation $k = N_{01} + N_{10}$, it is possible to model the probability distribution of $N_{10}$ as a Binomial $\mathcal{B}(k, q)$. The null hypothesis $N_0$ asserts that $N_{10}$ follows the binomial distribution $\mathcal{B}(k, \frac{1}{2})$. This last affirmation will be tested using a two-tailed test described in details by [21]. Given a great number of utterance it is also possible to model the probability density function by a Normal law $\mathcal{N}(0, 1)$.

# Appendix C

# t-SNE Visualization

Visualization of high-dimensional data is a complex task. The main problem is to reduce the number of dimensions while keeping the global structure of the data. An elegant method called t-SNE is proposed in [59] that provides a good trade-off being able to reduce the dimensionality while preserving the and the global structure by making use of probabilities. Stochastic Neighbor Embedding (SNE) is a way to transform the Euclidean distance between two vectors into a conditional probability.

$$p(j|i) = \frac{e^{(-\|x_i - x_j\|^2/2\sigma_i^2)}}{\sum_{k \neq i} e^{(-\|x_i - x_k\|^2/2\sigma_i^2)}} \tag{C.1}$$

where $\sigma_i^2$ is the variance of the Gaussian centered at $x_i$ which will be set to $\frac{1}{\sqrt{2}}$ for ease of representation. Setting the variance only implies that the model will be less accurate. This probability model a similarity between the two vectors $x_i$ and $x_j$. The idea is to find $y_i$ and $y_j$ (with lower dimension) that has the same similarity as $x_i$ and $x_j$ denoted by $q_{j|i}$.

$$q(j|i) = \frac{e^{(-\|y_i - y_j\|^2)}}{\sum_{k \neq i} e^{(-\|y_i - y_k\|^2)}} \tag{C.2}$$

where $q_{i|i} = 0$ as the modeling is done on similarities. Finding those $y_i$ and $y_j$ will be done by gradient descent as a cost function defined by the Kullback-Leibler divergence.

$$C = \sum_i \sum_j p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right) \tag{C.3}$$

t-SNE (t-Stochastic Neighbor Embedding) is a slightly different variant of SNE using Student t-distribution with 1 degree of freedom and the symmetric variant of SNE (further precision can be found on [59]). Regarding the case of posteriors, this visualization technique enables interpretation and act as an approximate visualization of the feature space [35]. The clustering present in the real space is also kept in the dimensional reduction plot as stated by [59]. Implementation of t-SNE used has been done by Laurens van der Maaten, copyright: Laurens van der Maaten, 2010, University of California, San Diego.[1].

[1]`http://lvdmaaten.github.io/tsne/`