University of Fribourg
Department of Mathematics

Idiap
Research Institut

Master's Thesis

Sandrine Revaz
Milos Cernak (supervisor at Idiap)
Christian Mazza (supervisor at University of Fribourg)

Statistical Models in Automatic Speech Recognition

June 2015

Abstract

The general subject of this work is to present mathematical methods encountered in automatic speech recognition (ASR).

Learning, evaluation and decoding problems are important parts in ASR and need hidden Markov models to solve them. These processes are explained in the first chapter after some basic definitions. Because ASR model has roots in machine learning, we present it in the second part of this chapter. What is important in this field of study is the learning in its various forms; the supervised and unsupervised learning are explained with their mathematical links. Artificial neural network is touched on in the last part of this section being widely used in supervised or unsupervised learning.

In the second chapter explanations of the organisation of ASR systems are given, together with mathematical formulation. Therefore acoustic features, language model, pronounciation dictionary, acoustic model and decoder are presented with details of their respective methods. At the end of this chapter, we speak about the performance evaluation of the ASR system.

This performance can be improved by unsupervised learning with confidence measures (CM) introduced in the third chapter. Acoustic CM, utterance verification, confidence estimation methods for neural network and CM for hybrid HMM/ANN are explained.

Finally, we present experiments done in 2013 during a three months trainee work at Idiap Research Institut. It is on development of German automatic speech recognition (ASR) system using a German part of multilingual database Mediaparl. Results confirm that supervised hybrid HMM/ANN system performs the best.

Contents

Chapter 1

Introduction to Machine Learning in Mathematics

1.1 Basic Definitions

Let Ω be the state space, $\omega \in \Omega$, and $\mathcal{P}(\Omega)$ the power set of Ω, i.e. the set of all subsets of Ω, here are some basic definitions:

Definition 1. $\mathcal{A} \in \mathcal{P}(\Omega)$ is an **algebra** if

1. $\Omega \in \mathcal{A}$
2. $\forall A \in \mathcal{A} \Rightarrow A^C \in \mathcal{A}$
3. $\forall A, B \in \mathcal{A} \Rightarrow A \cup B \in \mathcal{A}$

An algebra is a **σ-algebra**, $\mathcal{F}$, if it is closed for countable intersections and unions. $\sigma(\mathcal{A})$ is the σ-algebra generated by $\mathcal{A}$, i.e. the smallest σ-algebra containing $\mathcal{A}$.

Definition 2. A set function $P : \mathcal{A} \longrightarrow [0, 1]$, where $\mathcal{A}$ is an algebra, is a **probability** if

1. $0 \leq P(A) \leq 1, \ \forall A \in \mathcal{A}$
2. $P(\emptyset) = 0$, $P(\Omega) = 1$
3. Let $(A_i)_{i \in I}$ be a family of countable elements of $\mathcal{A}$ s.t. $A_i \cap A_j = \emptyset$ if $i \neq j$, $\bigcup\limits_{i \in I} A_i \in \mathcal{A}$ then $P(\bigcup\limits_{i \in I} A_i) = \sum\limits_{i \in I} P(A_i)$ (additivity).

Theorem 1. (Caratheodory's theorem) Let P be a probability on $(\Omega, \mathcal{A})$, where $\mathcal{A}$ is an algebra, then $\exists! Q$ on $\sigma(\mathcal{A})$ which coincides with P on $\mathcal{A}$, i.e. $Q(A) = P(A) \ \forall A \in \mathcal{A}$.

Thanks to the Caratheodory's theorem, we extend the definition of the probability on a σ-algebra; we note P the extended probability.

To work in $\mathbb{R}$ in the following, we need to define measurable functions named also random variable,

Definition 3. Let $\mathcal{F} \in \mathcal{P}(\Omega)$ be a σ-algebra, E a set and $\mathcal{E} \in \mathcal{P}(E)$ a σ-algebra, $X : \Omega \longrightarrow \mathcal{E}$ is a **random variable** if it is $(\mathcal{F}, \mathcal{E})$-measurable, i.e. $X^{-1}(\mathcal{E}) \in \mathcal{F}$.

In our situation, X is a real random variable so that $E = \mathbb{R}$ and $\mathcal{F} = \mathbb{B}$ where $\mathbb{B}$ is the σ-algebra of Borel.

In the following chapter, we will consider families of random values that have the Markov property which permits us to define hidden Markov models used in speech recognition.

1.2 Hidden Markov Models (HMMs) and Speech Recognition

1.2.1 Hidden Markov Models

Before introduce hidden Markov model, we need the definition of the Markov chain,

Definition 4. A family $(X_n)_{n\in\mathbb{R}}$ of random variables is a **Markov chain** with values in E if $\forall n \geq 1$, $\forall i,\ j,\ i_0,\ i_1, \dots,\ i_{n-1} \in E$, we have:

$$P(X_{n+1} = j|X_n = i, X_{n-1} = i_{n-1}, \dots, X_1 = i_1, X_0 = i_0) = P(X_{n+1} = j|X_n = i)$$

when $P(X_n = i) > 0$ and $P(X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) > 0$.

A hidden Markov model is a stochastic finite-state automaton which generates a sequence of observable symbols. The sequence of states is a Markov chain (see definition 4), i.e. there exists transition probability. Each state has an associated probability function to generate an observable symbol. This is a hidden Markov chain because only the sequence of observations is visible and the sequence of states is not observable, i.e. hidden.

Definition 5. A **hidden Markov model** (HMM) $\mathcal{M} = (A, B, \Pi)$ is defined by

- A state space $\Omega = \{1, 2, \dots, N\}$;
- An output observation alphabet $O = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_M\}$, where M is the number of observable symbols;
- A probability distribution of transitions between states, defined as the matrix $A = \{a_{ij}\}$, where
$$a_{ij} = P(X_t = j|X_{t-1} = i), \ \ 1 \leq i, j \leq N$$
where X_t is the state at time t;
- An output probability distribution $B = \{b_i(\boldsymbol{o}_k)|i \in \Omega\}$, where
$$b_i(\boldsymbol{o}_k) = P(\boldsymbol{o}_k|X_t = i);$$
- An initial state distribution $\Pi = \{\pi_i := P(X_0 = i)|i \in \Omega\}$.

Hidden Markov models are classified according to the properties of their Markov chain. There exist two types of HMM:

- An HMM is **ergotic** (see figure 1.1) if its Markov chain is ergotic, i.e. positive recurrent, irreductible and aperiodic.

  Definition 6. $i \in \Omega$ is **recurrent** if $P(\inf\{n \geq 1; X_n = i\} < +\infty) = 1$.

  Definition 7. An Markov chain is **irreductible** if $|\Omega/\mathcal{R}| = 1$ where $\mathcal{R}$ is the relation
$$i\mathcal{R}j \Leftrightarrow \text{ either } i = j \text{ or } i \leftrightarrow j,$$
  Where $i \rightarrow j$ means $\exists n \geq 1$ s.t. $P(X_n = j|X_0 = i) > 0$ and $i \leftrightarrow j$ means $i \rightarrow j$ and $j \rightarrow i$.

  Definition 8. The **period** of a state i is the greatest common divisor of the set $\{n \geq 0|P(X_n = j|X_0 = i) > 0\}$. If two states communicate, they have the same period; one can thus speak of the period of a state class. If the period is 1, the class is called **aperiodic**.

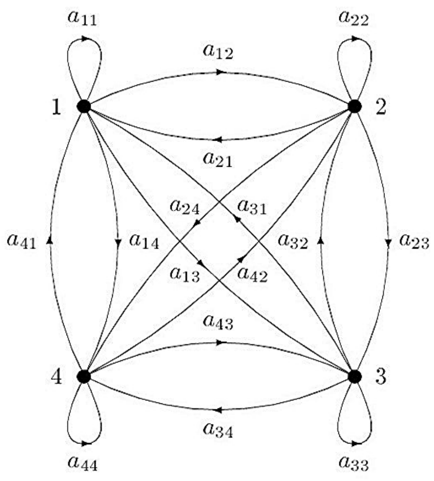

Figure 1.1: Architecture of an ergotic HMM with 4 states.

- A **left-to-right HMMs** are HMMs for which the transition matrix $\boldsymbol{A}$ is uppertriangular, i.e.

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ 0 & a_{22} & \cdots & a_{2N} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{NN} \end{pmatrix}$$

  and the initial state has probability one, i.e. $\pi_1 = 1$ (see figure 1.2).

  Left-to-right HMMs are particulary well suited to modeling stochastic transitory processes like speech processing model.

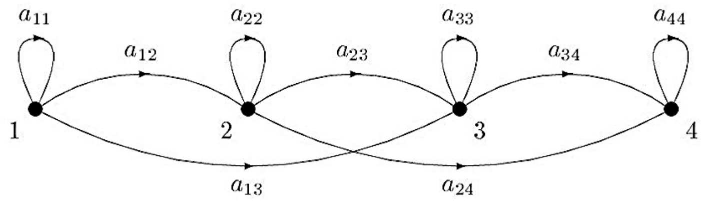

Figure 1.2: Architecture of a left-to-right HMM with 4 states.

Given an observation sequence $\boldsymbol{O} = \boldsymbol{o}_1\boldsymbol{o}_2\cdots\boldsymbol{o}_T$ and a hidden Markov model $\mathcal{M} = (A, B, \Pi)$, we meet three main problems in speech recognition:

- **Learning problem**: How to estimate the model parameters to maximize $P(\boldsymbol{O}|\mathcal{M})$, the probability of the observation sequence given the model?
  (This problem is encountered in the acoustic model in the training part (see section 2.5).)

- **Evaluation problem**: How $P(\boldsymbol{O}|\mathcal{M})$ is efficiently computed?
  (This problem is encountered in the acoustic model in the recognition part (see section 2.5).)

- **Decoding problem**: How to choose the corresponding state sequence $X = X_1X_2\cdots X_T$ that is optimal in some sense?
  (This problem is encountered in the decoding part (see section 2.6).)

1.2.2 Learning Problem

The parameters of the model $\mathcal{M}$ can be estimated by maximizing $P(\boldsymbol{O}|\mathcal{M})$ locally using an iterative algorithm, such as the Baum-Welch algorithm, also called forward-backward algorithm.

Baum-Welch Algorithm (see figure 1.3)

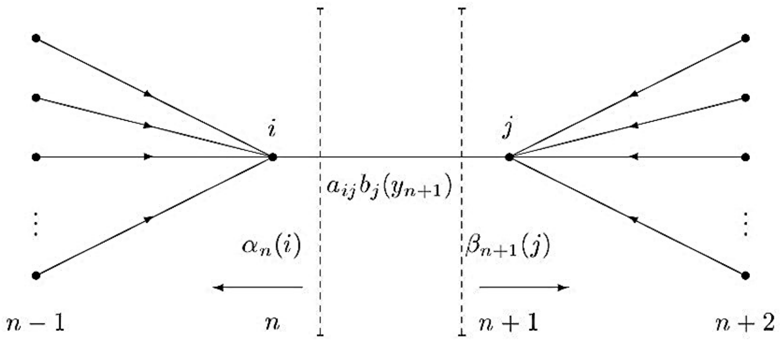

Figure 1.3: Illustration of the procedure of the Baum-Welch algorithm.

First we solved inductively the forward probability $\alpha_t(i)$ and the backward probability $\beta_t(i)$.

- **Definition**,
$$\alpha_t(i) = P(\boldsymbol{o}_1\boldsymbol{o}_2\cdots\boldsymbol{o}_t, X_t = i|\mathcal{M})$$
i.e. $\alpha_t(i)$ is the probability of the partial observation sequence in state i at time t, given the model $\mathcal{M}$;
$$\beta_t(i) = P(\boldsymbol{o}_{t+1}\boldsymbol{o}_{t+2}\cdots\boldsymbol{o}_T|X_t = i, \mathcal{M})$$
i.e. $\beta_t(i)$ is the probability of seeing the partial observation sequence from time $t+1$ to the end in sate i at time t given the model $\mathcal{M}$.

- **Initialization**, for $1 \leq i \leq N$:
$$\alpha_t(i) = \pi_i b_i(\boldsymbol{o}_1);$$
$$\beta_T(i) = \frac{1}{N};$$

- **Induction**, for $1 \leq j \leq N$:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right], \quad 1 \leq t \leq T-1;$$
$$\beta_t(j) = \left[\sum_{i=1}^{N} a_{ij}b_j(\boldsymbol{o}_{j+1})\beta_{t+1}(j)\right], \quad t = T-1, T-2, \dots, 1.$$

- **Update**,
Before reestimation procedure, we need to define two auxilary variables,
$$\xi_t(i,j) := P(X_t = i, X_{t+1} = j|\boldsymbol{O}, \mathcal{M})$$
which is the probability of being in state i at time t, and state j at time $t+1$ and
$$\gamma_t(i) := \sum_{j=1}^{N} \xi_t(i,j)$$

which is the probability of being in state i at time t and every state at time $t+1$.

We can rewrite $\xi_t(i,j)$ as

$$\xi_t(i,j) = \frac{P(X_t = i, X_{t+1} = j, \boldsymbol{O}|\mathcal{M})}{P(\boldsymbol{O}|\mathcal{M})} = \frac{\alpha_t(i)a_{ij}b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\boldsymbol{o}_{t+1})\beta_{t+1}(j)}$$

Using the above formulas, the reestimation procedure of the HMM parameters is defined as

$$\tilde{\pi}_i = \gamma_1(i)$$
$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$$
$$\tilde{b}_j(\boldsymbol{o}_k) = \frac{\sum_{t=1,\boldsymbol{o}_t=\boldsymbol{o}_k}^{T}\xi_t(i,j)}{\sum_{t=1}^{T}\gamma_t(i)}$$

Thus we obtain an updated model $\tilde{\mathcal{M}} = (\tilde{A}, \tilde{B}, \tilde{\pi})$ with $P(\boldsymbol{O}|\tilde{\mathcal{M}}) > P(\boldsymbol{O}|\mathcal{M})$.

This process is repeated until it reachs some limiting point.
Ultimately we obtain the model $\tilde{\mathcal{M}} = (\tilde{A}, \tilde{B}, \tilde{\pi})$ which maximizes $P(\boldsymbol{O}|\boldsymbol{\mathcal{M}})$.

1.2.3 Evaluation Problem

A simple way to compute $P(\boldsymbol{O}|\mathcal{M})$ is to use the forward algorithm which evaluates state by state the probability of being at that state given the observation sequence

$$P(\boldsymbol{O}|\mathcal{M}) = \sum_{i=1}^{N}\alpha_T(i)$$

where $\alpha_T(i)$ is solved by induction:

- **Initialization**:
$$\alpha_t(i) = \pi_i b_i(\boldsymbol{o}_1), \ \ 1 \le i \le N;$$
- **Induction**:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N}\alpha_t(i)a_{ij}\right], \ \ 1 \le t \le T-1, \ 1 \le j \le N;$$

The forward algorithm has a complexity of $\mathcal{O}(N^2T)$.

1.2.4 Decoding Problem

This problem is solved with the Viterbi algorithm (see figure 1.4), for more explanation about decoding have a look at decoder on section 2.6.

Viterbi algorithm:
Suppose we are given a Hidden Markov Model (HMM) with state space Ω, initial probabilities π_i of being in state i and transition probabilities a_{ij}. Say we observe outputs $\boldsymbol{o}_1 \cdots \boldsymbol{o}_T$. We define $\psi_t(X_t)$ as a saving back pointer that remember which state X_t was used at time t. The most likely state sequence $X_1 \cdots X_T$ that produces the observations is given by this processus:

- **Initialization**
$$\delta_1(i) = \pi_i b_i(\boldsymbol{o}_1), \ \ 1 \le i \le N$$
$$\psi_1(i) = 0$$

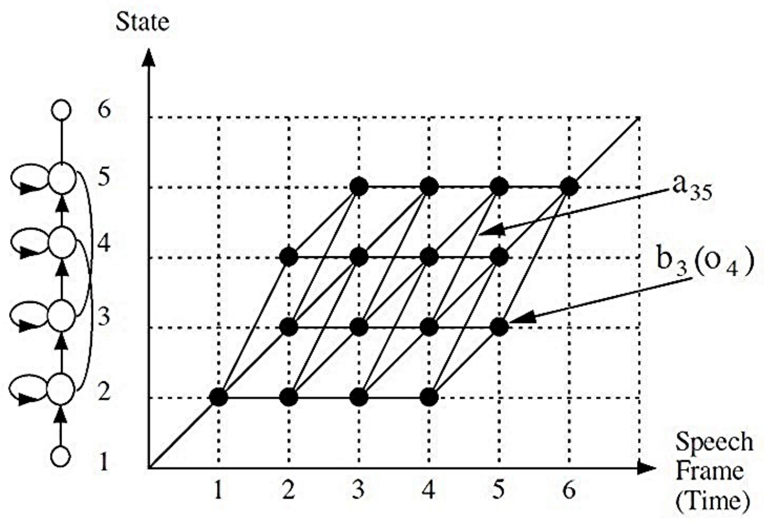

Figure 1.4: Illustration of the Viterbi algorithm for an isolated word.

- **Recursion**

$$\delta_t(j) = \max_{1\leq i\leq N}(\delta_{t-1}(i)a_{ij})b_j(\boldsymbol{o}_t),$$

$$\psi_t(j) = \arg\max_{1\leq i\leq N}(\delta_{t-1}(i)a_{ij}), \quad 2 \leq t \leq T, 1 \leq j \leq N$$

- **Termination**

$$P^* = \max_{1\leq i\leq N}(\delta_T(i)),$$

$$X_t^* = \arg\max_{1\leq i\leq N}(\delta_T(i)).$$

- **Path backtracking**

$$X_t^* = \psi_{t+1}(X_{t+1}^*), \quad t = T-1, T-2, \ldots, 1$$

Speech recognition, the main part of the subject, is a particular context of **machine learning** that we will define in the following chapter.

1.3 Machine Learning

Arthur Samuel (1959) gives this definition of machine learning:

Definition 9. Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998) completes with

Definition 10. Well-posed learning problem: a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Here is a comparison of different denominations between statisticians and computer scientists:

Statistics / Data Mining Dictionary		
Statistics	**Computer Science**	**Meaning**
Estimation	Learning	Using data to estimate an unkown quantity
Classification	Supervised learning	Predicting a discrete Y from X
Clustering	Unsupervised learning	Putting data into groups
Data	Training sample	$(X_1, Y_1), \dots, (X_l, Y_l)$
Covariates	Features	The X_i's
Classifier	Hypothesis	A map from covariates to outcomes
Hypothesis	-	Subset of a parameter space Θ
Large deviation bounds	PAC learning	Uniform bounds on probability of errors

Table 1.1: Comparison of different namings between statisticians and computer scientists, and their meaning.

1.3.1 Learning Models

There exist several learning algorithms, we will speak about supervised, unsupervised and semi-supervised one.

- **Supervised learning** algorithms are trained on labelled examples, i.e. inputs associated with known outputs. In the context of speech recognition an input is some spoken sentences and its associated output is their text transcriptions. Formally, we have a data set $\{(X_1, Y_1), \dots, (X_l, Y_l)\}$, called training set, where l is the number of training examples, X_i are input features and Y_i their corresponding outputs.

  The aim of supervised training is to generalise a function from this data set which can then be used to speculatively generate an output for previously unseen inputs; this problem is called in mathematics a **regression problem**. In other words, the algorithm creates a function which can predict new outputs linked to new inputs thanks to training with labelled examples. We will see later that artificial neural networks are a technique used in regression models (see section 1.3.4).

  If outputs are discrete values (generaly 0 or 1), we speak about **classification problems**.

- On the other side, **unsupervised learning** operates on unlabelled data, i.e. the associated outputs is unkown. To us, it means that we only have spoken sentences and no text transcriptions.

  Thus the objective is to find structure in the data to predict best text transcriptions, i.e. 'decoded' label; in statistics this is called **clustering problem**. Artificial neural networks are also a technique used in cluster analysis.

- **Semi-supervised learning** is a combination of supervised and unsupervised learning because it combines both labelled and unlabelled data.

  The principle is the same as supervised learning but the difference lies in the data. Supervised learning uses only 'true' labelled data while semi-supervised uses 'true' plus 'decoded' labelled data the last one obtained with unsupervised learning.

These problems, supervised and unsupervised, are explained in the next two sections, together with artificial neural networks which can be used in both methods.

1.3.2 Supervised Learning - Regression Problem

Linear Regression

In this subsection, we assume that r is linear.

Aim. Predict a real-valued output, $Y = r_\theta(x)$, estimated from a data set of the form $\{(X_1, Y_1), \dots, (X_l, Y_l)\} \sim F_{X,Y}$.

Thus the regression is a method for studying the relationship between a response variable Y and a covariate X, also called feature. The regression function is a good way to summarize the relationship between X and Y.

Definition 11. The **regression function** is defined by

$$r(x) = \mathbb{E}(Y|X = x) = \int_\Omega y f(y|x) dy. \tag{1.1}$$

For best understanding we will first explain the process for one-dimensional variable, i.e. simple linear regression, and then multiple.

- **Simple Linear Regression**

  In this part, we assume that X_i is one-dimensional, thus we obtain the simple linear regression model:
  $$r(x) = \beta_0 + \beta_1 x.$$
  We add the assumption that $Var(Y|X = x) = \sigma^2$ does not depend on x. We can thus write the linear regression model as follows.

  Definition 12. The simple linear regression model is defined by
  $$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \tag{1.2}$$
  where $\mathbb{E}(\epsilon_i|X_i) = 0$ and $Var(\epsilon_i|X_i) = \sigma^2$.

  The unknow parameters in the model are the intercept β_0, the slope β_1 and the variance σ^2. Let $\hat\beta_0$ and $\hat\beta_1$ denote estimates of β_0 and β_1 respectively. The predicted values are
  $$\hat{Y}_i := \hat{r}(X_i) = \hat\beta_0 + \hat\beta_1 X_i \tag{1.3}$$
  and the residuals are defined to be
  $$\hat\epsilon_i := Y_i - \hat{Y}_i = Y_i - \left(\hat\beta_0 + \hat\beta_1 X_i\right). \tag{1.4}$$

  To measure how well the line fits the data, we use the residual sums of squares, $\sum_{i=1}^l \hat\epsilon_i^2$. Thus the **least squares estimates** are the values $\hat\beta_0$ and $\hat\beta_1$ that minimize $\sum_{i=1}^l \hat\epsilon_i^2$. For finding them, we have this theorem

  Theorem 2. The least squares estimates are given by
  $$\hat\beta_1 = \frac{\sum_{i=1}^l (X_i - \bar{X}_l)(Y_i - \bar{Y}_l)}{(X_i - \bar{X}_l)^2} \tag{1.5}$$
  $$\hat\beta_0 = \bar{Y}_l - \hat\beta_1 \bar{X}_l, \tag{1.6}$$
  where $\bar{X}_l$ is the sample mean, $\bar{X}_l = \frac{1}{l}\sum_{i=1}^l X_i$.

  And an unbiased estimate of σ^2 is
  $$\hat\sigma^2 = \left(\frac{1}{l-2}\right) \sum_{i=1}^l \hat\epsilon_i^2. \tag{1.7}$$

- **Multiple Regression**

  Now suppose that the feature $\mathbf{X}_i$ is a vector of length k. The data set is of the form
  $$\{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_i, Y_i), \dots, (\mathbf{X}_l, Y_l)\}$$
  where
  $$\mathbf{X}_i = \begin{pmatrix} X_{i1} \\ X_{i2} \\ \vdots \\ X_{ik} \end{pmatrix}.$$

  Definition 13. The linear regression model is
  $$Y_i = \sum_{j=1}^{k} \beta_j X_{ij} + \epsilon_i \tag{1.8}$$
  where $\mathbb{E}(\epsilon_i | X_{i1}, \dots, X_{ik}) = 0,\ \forall i = 1, \dots, l$.

  Instead of adding the intercept β_0 like in the simple linear regression model, see (1.2), we set $X_{i0} = 1,\ \forall i = 1, \dots, l$.

  By convenience, we express the model in matrix notation. The outcomes will be denoted by
  $$\boldsymbol{Y} = \begin{pmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_l \end{pmatrix}$$
  and the covariates will be denoted by
  $$\boldsymbol{X} = \begin{pmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1k} \\ 1 & X_{21} & X_{22} & \cdots & X_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{l1} & X_{l2} & \cdots & X_{lk} \end{pmatrix}.$$
  Thus $\boldsymbol{X}$ is a $(l+1) \times k$ matrix where each row is an observation and each columns correspond to the covariates.

  Let
  $$\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix} \text{ and } \boldsymbol{\epsilon} = \begin{pmatrix} 0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_l \end{pmatrix},$$
  then we can write (1.8) as
  $$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{1.9}$$
  The least squares estimate are given in the following theorem

  Theorem 3. Assuming that the $(k \times k)$ matrix $\boldsymbol{X}^\top \boldsymbol{X}$ is invertible,
  $$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y} \tag{1.10}$$
  $$\mathrm{Var}(\hat{\boldsymbol{\beta}} | \boldsymbol{X}^l) = \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \tag{1.11}$$
  $$\hat{\boldsymbol{\beta}} \approx \mathcal{N}(\boldsymbol{\beta}, \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}). \tag{1.12}$$

The estimate regression function is

$$\hat{r}(x) = \sum_{j=1}^{k} \hat{\beta}_j x$$

and an unbiased estimate of σ^2 is

$$\hat{\sigma}^2 = \left(\frac{1}{l-k}\right) \sum_{i=1}^{l} \hat{\epsilon}_i^2$$

where $\hat{\epsilon} = \boldsymbol{X}\hat{\boldsymbol{\beta}} - \boldsymbol{Y}$ is the vector of residuals.

Classification Problem

The problem of predicting a discrete random variable Y from another random variable X is called **classification**, **supervised learning**, discrimination or pattern recognition. Let $(X_1, Y_1), \ldots, (X_l, Y_l)$ be i.i.d. data where

$$\mathbf{X}_i = \begin{pmatrix} X_{i1} \\ X_{i2} \\ \vdots \\ X_{ik} \end{pmatrix} \in \mathcal{X} \subset \mathbb{R}^k$$

is a k-dimensional vector and Y_i takes values in some finite set $\mathcal{Y}$. A **classification rule** is a function $h : \mathcal{X} \to \mathcal{Y}$; when we observe a new X, we predict Y to be $h(X)$.

Aim. Find a classification rule h that makes accurate predictions.

First, some definitions:

Definition 14. The Fisher information, $I(\cdot)$, is a way of measuring the amount of information that an observable random variable X carries about an unknown parameter θ upon which the probability of X depends. I is defined as the second moment of the score which is the partial derivative with respect to θ of the natural logarithm of the likelihood function, i.e.

$$I(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log P(X|\theta)\right)^2 \middle| \theta\right]$$

Definition 15. The **true error rate** of a classifier h is

$$L(h) = P(\{h(X) \neq Y\}) \tag{1.13}$$

and the **empirical error rate** or **training error rate** is

$$\hat{L}_l(h) = \frac{1}{l}\sum_{i=1}^{l} I(h(X_i) \neq Y_i) \tag{1.14}$$

where $I(\cdot)$ is the Fisher information.

Firstly we consider the special case $\mathcal{Y} = \{0, 1\}$ and secondly a generalisation.

- **Special case:** $\mathcal{Y} = \{0, 1\}$
  Let
  $$r(x) = \mathbb{E}(Y|X = x) = P(Y = 1|X = x)$$
  denote the regression function. From Bayes' theorem, we have that
  $$\begin{aligned} r(x) &= P(Y = 1|X = x) \\ &= \frac{f(x|Y = 1)P(Y = 1)}{f(x|Y = 1)P(Y = 1) + f(x|Y = 0)P(Y = 0)} \\ &= \frac{\pi f_1(x)}{\pi f_1(x) + (1 - \pi)f_0(x)} \end{aligned} \tag{1.15}$$
  where $f_r(x) = f(x|Y = r)$, $r = 0, 1$ and $\pi = P(Y = 1)$.

  Definition 16. The **Bayes classification rule** h^* is
  $$h^*(X) = \begin{cases} 1 & \text{if } r(x) > \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.16}$$
  The set $\mathcal{D}(h) := \{x|P(Y = 1|X = x) = P(Y = 0|X = x)\}$ is called the **decision boundary.**

  Theorem 4. The Bayes rule is optimal, that is, if h is any other classification rule then $L(h^*) \leq L(h)$.

  We need to use the data to find some approximation to the Bayes rule as it depends on unknown quantities. We use the regression model to estimate $\hat{h}$:
  $$\hat{h}(X) := \begin{cases} 1 & \text{if } \hat{r}(x) > \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.17}$$
  where $\hat{r}$ is an estimate of the regression function r.

- **Generalization**
  Now, let us generalize to the case where Y takes more than two values as follows.

  Theorem 5. Suppose that $Y \in \mathcal{Y} = \{1, \dots, K\}$. The optimal rule is
  $$\begin{aligned} h(X) &= \arg\max_k P(Y = k|X = x) && (1.18) \\ &= \arg\max_k \pi_k f_k(x) && (1.19) \end{aligned}$$
  where
  $$P(Y = k|X = x) = \frac{f_k(x)\pi_k}{\sum_r f_r(x)\pi_r}, \tag{1.20}$$
  $\pi_r = P(Y = r)$, $f_r(x) = f(x|Y = r)$ and $\arg\max_k$ means 'the value of k that maximizes that expression'.

1.3.3 Unsupervised Learning - Cluster Analysis

Remember that the problem of unsupervised learning is that of trying to find hidden structure in unlabeled data. In mathematics we call this problem cluster analysis.

Aim. Grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.

Cluster analysis can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. So we can have connectivity models like **hierarchical clustering** based on distance connectivity or centroid models like k-**means algorithm** which represents each cluster by a single mean vector, or others. These two kinds of clustering, explained above, are used for unsupervised learning in the context of speech recognition.

Hierarchical Clustering

Aim. The aim of the hierarchical cluster analysis is to build a hierarchy of clusters.

To reach this objective there exist two different strategies:

- **Agglomerative**: each observation starts in its own cluster and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: all observations start in one cluster and splits are performed recursively as one moves down the hierarchy.

A measure of dissimilarity between sets of observations is required to decide respectively which clusters should be combined (for agglomerative), or when a cluster should be split (for divisive). This is achieved by use of an appropriate **metric** and a **linkage criterion** which specifies the dissimilarity of sets.

The choice of an appropriate metric will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. For text or other non-numeric data, metric such as the Hamming distance is often used.

Definition 17. The **Hamming distance**, $d(\cdot,\cdot)$, between two strings of equal length is the number of positions at which the corresponding symbols are different. Formally, let A be an alphabet and F the set of sequences of length n to value A. The Hamming distance between two elements $a = (a_i)_{i\in[1,n]}$ and $b = (b_j)_{j\in[1,n]} \in F$ is the number of elements of the set of images of a which is different from b,

$$d(a,b) := \#\{i|a_i \neq b_i\}.$$

Examples 1. The Hamming distance between:

- "words" and "world" is 2.
- "instance" and "distance" is 2.
- "two" and "too" is 1.

The problem is that entries must have the same lentgh. The generalisation of the Hamming distance for different length is the **Levenshtein distance**.

Definition 18. The Levenshtein distance between two words is the minimum number of single-character edits, i.e. insertions, deletions or substitutions. Formally let $a = (a_i)_{i\in[1,n]}$ and $b = (b_j)_{j\in[1,n]} \in F$ as in the Hamming distance definition, $lev_{a,b}(|a|,|b|)$ is given by

$$lev_{a,b}(i := |a|, j := |b|) = \begin{cases} \max(i,j), & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j)+1, \\ lev_{a,b}(i,j-1)+1, \\ lev_{a,b}(i-1,j-1)+\mathbf{1}_{a_i\neq b_j}, \end{cases} & otherwise, \end{cases}$$

where $\mathbf{1}_{a_i\neq b_j}$ is the indicator function equal to 0 when $a_i = b_j$. Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

Examples 2. The Levenshtein distance between:

- "sing" and "singing" is 3 because we add three letters.
- "homework" and "home" is 4 because we delete four letters.

- "lettre" and "letter" is 2 because there are two substitutions.
- "definition" and "decision" is 6 because there are 4 substitutions and 2 additions.

Note 1. With this definition, we see that the letters additions can not be done in the middle of the word. For example between "decision" and "definition" we could do this changes:

1. "decision" $\rightarrow$ "defision", one substitution;
2. "defision" $\rightarrow$ "definion", one substitution;
3. "definion" $\rightarrow$ "definiton", one addition in the middle;
4. "definiton" $\rightarrow$ "definition", one addition in the middle.

So we obtain a result of 4 instead of 6.

The linkage criterion determines the distance between sets of observations as a function of the pairwise distances between observations. For example let A, B being two sets of observations and $d(\cdot,\cdot)$ the distance, the single-linkage clustering is $\min\{d(a,b) | a \in A, b \in B\}$.

k-**Mean Algorithm**

Given a set of observations, $\{X_1, X_2, \ldots, X_n\}$, here is the aim of the k-mean clustering,

Aim. Partition the n observations into k $(\leq n)$ sets $\mathbf{S} = \{S_1, \ldots, S_k\}$ in which each observation belongs to the cluster with the nearest mean. The objective is to minimize the within-cluster sum of squares, i.e. $\underset{\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{X \in S_i} \|X - \mu_i\|^2$ where μ_i is the mean of observations in S_i and $\|\cdot\|$ the Euclidean distance.

To achieve this aim, there is a standard algorithm which proceeds by alternating between two steps:
Given an initial set of k means $m_1^{(0)}, m_2^{(0)}, \ldots, m_k^{(0)}$ and let $t \in \mathbb{N}$ be the mutable state,

- **Assignment step**: Each observation is assigned to the cluster, $S_i^{(t)}$, whose mean gives the least within-cluster sum of squares.

$$S_i^{(t)} = \{X_l \mid \|X_l - m_i^{(t)}\|^2 \leq \|X_l - m_j^{(t)}\|^2, \ \forall j \text{ s.t. } 1 \leq j \leq k\},$$

  where each X_l is assigned to exactly one $S_i^{(t)}$.

- **Update step**: Update the new means

$$m_i^{(t+1)} = \frac{1}{S_i^{(t)}} \sum_{X_l \in S_i^{(t)}} X_l$$

  whose represent the centroids of the observations in the new clusters.
  This also minimizes the within-cluster sum of squares objective because arithmetic mean is a least-squares estimator.

The algorithm converges, to a local optimum, when the assignments no longer change; this can happen because there only exists a finite number of partitionings. There is no guarantee that it will converge to the global optimum. The result may depend on the initial clusters so it is common to run it multiple times with different starting conditions as the algorithm is usually very fast.

The number of clusters k is an input parameter, an inappropriate choice of k may yield poor results. The correct choice of k is often ambiguous, with interpretations depending on the shape and scale of the distribution of the observations and the desired clustering resolution. One simple manner is the rule of thumb which sets the number to

$$k \approx \sqrt{n/2}$$

where n is the number of observations. There are other methods like for example Elbow method not explained in this paper.

1.3.4 Artificial Neural Networks (ANN)

Artificial Neural Networks are computational models inspired by the brain; they are capable of machine learning and pattern recognition. The architecture are presented as systems of interconnected artificial neurons which can compute values from input by feeding information through the network.

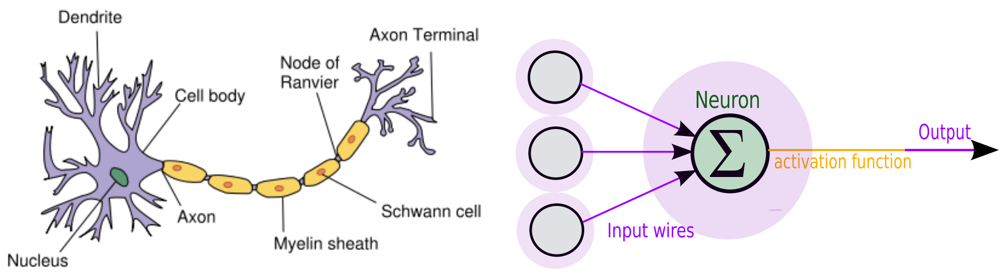

Figure 1.5: Comparison of biological (left) and artificial (right) neuron.

An artificial neuron (see figure 1.5) receives one or more 'inputs wires', representing the dendrites, and sums them to produce output, representing an axon. The sums of each node are weighted, representing the importance of the information i.e. the transmission speed (depending in part on myelin), and passed through an activation function, representing the rate of action potential firing in the cell.

An ANN is defined by three main components

1. The **activation function** that converts input to output,
2. The **architecture** pattern between different layers of neurons,
3. The **learning process** for updating the weights of the interconnections.

Activation Function

The activation function, denoted $g(z)$, is a monotically increasing, continuous, differentiable and bounded function like for example Heaviside step function or sigmoid function.

Definition 19. The **Heaviside function** $H(\cdot)$ is the integral of the Dirac delta function $\delta(\cdot)$

$$H(t) = \int_{-\infty}^{t} \delta(s) ds$$

where $\forall t \in \mathbb{R}$

$$\delta(t) = \begin{cases} +\infty, & t = 0 \\ 0, & t \neq 0 \end{cases} \tag{1.21}$$

with constraint

$$\int_{-\infty}^{\infty} \delta(s)ds = 1.$$

Definition 20. A **sigmoid function** (see figure 1.6), a function with a 'S' shape, is defined by the formula,

$$S(t) := \frac{1}{1+e^{-t}}, \ \forall t \in \mathbb{R}$$

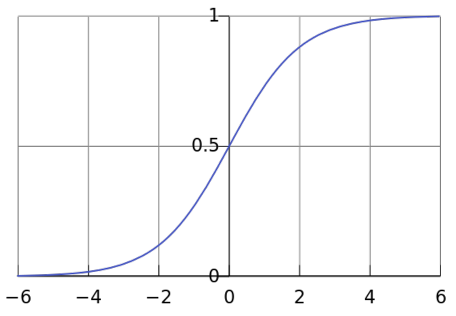

Figure 1.6: Graph of the sigmoid function.

The sigmoid function is often generalised by

$$S_\theta(t) = \frac{1}{1+e^{-\theta t}}$$

where θ is a constant in $\mathbb{R}$.
For $\boldsymbol{t} \in \mathbb{R}^n$ and a constant $\boldsymbol{\theta} \in \mathbb{R}^n$, we have

$$S_{\boldsymbol{\theta}}(\boldsymbol{t}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \boldsymbol{t}}}$$

ANN Architecture

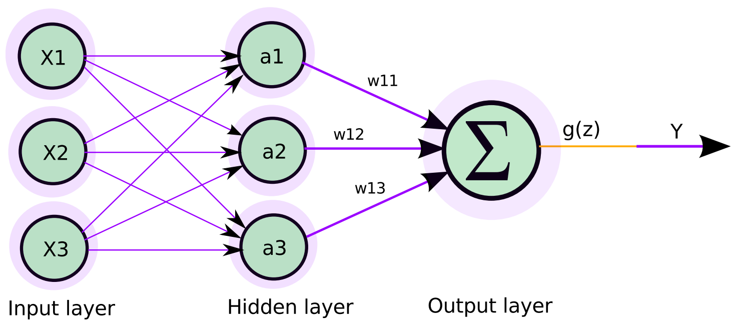

Figure 1.7: Architecture of a simple ANN (3 layers).

The different layers of an ANN architecture (see figure 1.7) are

- the first layer called **the input layer**,
- the middles called **hidden layers**,

- and the last called **the output layer**.

More middle layers there are, the more complex non-linear functions can be learn.

Here are the corresponding notations:

- $g : \Omega \longmapsto \Omega$ the activation function;
- $\boldsymbol{X} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{l_1} \end{pmatrix} \in \Omega^{l_1+1}$ is the input layer with l_1 units plus $X_0 = 1$, the bias unit;
- $\boldsymbol{a}^{(j)} = \begin{pmatrix} a_0^{(j)} \\ a_1^{(j)} \\ a_2^{(j)} \\ \vdots \\ a_{l_j^{(j)}} \end{pmatrix} \in \Omega^{l_j+1}$ is the hidden layer j with l_j neurons plus the bias unit $a_0^{(j)}$ where $j \in \mathbb{N}$ and $\boldsymbol{a}^{(0)} := \boldsymbol{X}$;
- $W^{(j)} = \{w_{ik}^{(j)}\} \in M([0,1],\ l_{j+1},\ l_j + 1)$ where $w_{ik}^{(j)}$ is the weight between the i^{th} and k^{th} element of $\boldsymbol{a}^{(j+1)}$ and $\boldsymbol{a}^{(j)}$ respectively;
- and $\boldsymbol{Y} := \boldsymbol{a}^{(m)} \in \Omega^{l_m}$ the output where we assume that m is the output layer.

Learning Process

In classification problem, there are two kinds of propagation, explained below, first a forward propagation to calculate $\boldsymbol{Y}$ and then a backward one to calculate the errors of the cost function for every unit.

Forward propagation: For $1 \le j \le m$, we define for practical notations $z^{(j)} = \begin{pmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \vdots \\ z_{l_j^{(j)}} \end{pmatrix}$ by

$$z_i^{(j)} := w_{i0}^{(j-1)} a_0^{(j-1)} + w_{i1}^{(j-1)} a_1^{(j-1)} + \cdots + w_{il_{j-1}}^{(j-1)} a_{l_{j-1}}^{(j-1)},\ \forall 1 \le i \le l_j.$$

Thus, every layer $1 \le j \le m$ is calculable by this reccurent relation

$$a^{(j)} = g(z^{(j)})$$

where

$$g(z^{(j)}) := \begin{pmatrix} g(z_1^{(j)}) \\ g(z_2^{(j)}) \\ \vdots \\ g(z_{l_j^{(j)}}) \end{pmatrix}$$

At every step, a bias unit $a_0^{(j)} = 1$ is added to the layer, for $1 \le j \le m - 1$.

Backward propagation: Suppose we have a training set $\{(X_1, Y_1), \dots, (X_{l_1}, Y_{l_1})\}$. Let $a_i^{(1)} = X_i, \ \forall i$ and we set

$$\delta_i^{(m)} = a_i^{(m)} - Y_i,$$

and we have this recursive formula

$$\delta^{(j)} = (W^{(j)})^T \delta^{(j+1)} g'(z^{(j)}),$$

where $\delta_i^{(j)}$ represents the error of node i in layer j.

The most interesting possibility in neural network is the learning process. Learning means, using a set of observations, to find a solution which solves the task in some optimal sense.

Definition 21. Let F be a set of functions, a **cost function** is a function $C : F \longrightarrow \mathbb{R}$ such that, for the optimal solution f^*,

$$C(f^*) \leq C(f), \ \forall f \in F,$$

i.e. no solution has a cost less than the cost of the optimal solution.

The cost function C is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Entropy or mean squared error can be examples of cost functions. Learning algorithms search through the solution space to find a function that has the smallest possible cost.

The weights are adaptatives, i.e. they are tuned by a learning algorithm.

Chapter 2

Components of Automatic Speech Recognition Systems

2.1 Fundamentals of Automatic Speech Recognition

2.1.1 Main Components

The automatic speech recognition (ASR) is simply the translation of spoken words into text. In this section, the key components of an ASR system are broadly presented.

- **Acoustic Features**: A raw audio signal, for example as received from a microphone, needs to be converted into a more manageable form because it is too complex to deal with when it comes to the task of speech recognition.

  First the incoming audio is treated as a sequence of frames at regular time intervals, 10 [ms] for example.

  These frames are then analysed such that some key (perceptually important) data can be extracted; this process is driven by the perception of speech and elimination of redundant data.

  For each audio frame, we obtain a representative feature vector.

- **Language Model** (LM): A language model describes how words can be combined.

- **Pronunciation Dictionary**: The pronunciation dictionary contains the set of words and their pronunciations using a common set of phonemes as for the acoustic model. Multiple entries can appear for a word depending on pronunciations. It can also contain homonyms.

- **Acoustic Model** (AM): An acoustic model contains the data describing the acoustic nature of all the phonemes understood by the system. Acoustic models are built through a training process using large quantities of transcribed audio. An acoustic model is usually specific to one language; and could be adjusted to a particular accent.

  One difficulty is that phonemes are context-dependent; they tend to sound different depending on what the previous and next phonemes are. Each context-dependent phoneme, also called triphone, is represented by a hidden Markov model (HMM). The HMM states permit to describe how the sound of the phonemes progresses in time. The number of states used can vary depending on the type of model, but five is typical. Thus HMM represents the phonemes.

  The acoustic model assigns parameters to each triphone state which describes the probability that the triphone state matches a feature vector, called probability density function (pdf).

- **Decoder**: The decoder is the reason of the ASR system; its job is to decode a sequence of speech signal to reveal what words were spoken.

  For each audio frame, there is a process of pattern matching; the decoder evaluates the received features against all active HMMs to determine probabilities of staying/changing states or HMMs. At any given time the decoder is keeping the trace of any possible HMM states in a decoding space; this is necessary because a state that has low local score can finally be the best match when more frames are processed or when the language model is considered. Thus the decoder can start building hypotheses of the most likely sequence of words as the process evolves.

  There exists a measure, called confidence measure (see chapter 3), to discard unlikely hypotheses.

These components are explained more precisely in the following sections. But first, let's look more closely at the mathematical formulation of the recognition process:

2.1.2 Mathematical Formulation of Recognition Process

The continuous speech waveform is first divided into frames with constant length (~ 25 ms). Each frame gives features represented by discrete parameter vectors, by supposing that for the duration of a single vector, i.e. one frame, the speech waveform can be regarded as being stationary. Although this is not strictly true but it is a reasonable approximation.

For each spoken word, let $\boldsymbol{O} = \boldsymbol{o}_1 \boldsymbol{o}_2 \cdots \boldsymbol{o}_\tau$ be a sequence of parameters vectors, where $\boldsymbol{o}_t$ is observed at time $t \in \{1, \ldots, \tau\}$. Given a dictionary $\mathcal{D}$ with words $w_i \in \mathcal{D}$, the recognition problem is summerized by

$$\tilde{W} = \arg \max_W P(W|\boldsymbol{O}) \tag{2.1}$$

where $\tilde{W}$ is the recognised word, P is the probability measure and $W = w_1 \cdots w_k$ a word sequence.

Bayes' Rule permits to transform (2.1) in a suitable calculable form:

$$P(W|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|W)P(W)}{P(\boldsymbol{O})} \tag{2.2}$$

where $P(\boldsymbol{O}|W)$ represents the acoustic model and $P(W)$ the language model; $P(\boldsymbol{O})$ can be unheeded. Thus, for a given set of prior probabilities $P(W)$, the most probable spoken word depends only on the likelihood $P(\boldsymbol{O}|w_i)$.

It is important to note that the combination of the acoustic model probability and the language model probability is weighted. The problem is that HMM acoustic models usually underestimate the acoustic probability (due to independence assumption) giving to the language model little weight. A solution for such problem is to add a weight to raise the language model probability. Thus the LM is often scaled by an empirically determined constant, s, called language model scale factor (LMSF). LMSF is determined empirically to optimize the recognition performance. This weighting has a side effect as a penalty for inserting new words. That's why we add a scaling factor, p, that penalizes word insertions called word insertion penalty (WIP) also calculated empirically. Thus (2.1) becomes

$$\tilde{W} = \arg \max_W P(\boldsymbol{O}|W)P(W)^s|W|^p \tag{2.3}$$

In the log domain, the total likelihood is calculated as

$$\log_{|W|} \tilde{W} = \log_{|W|} P(\boldsymbol{O}|W) + s \ \log_{|W|} P(W) + p \tag{2.4}$$

where $|W|$ is the length of the word sequence W, s is the language model scale factor (LMSF) (typically in the range of 5 to 15) and p the word insertion penalty (WIP) (typically in the range of 0 to -20).

Thus, if the language model probability decreases (large penalty), the decoder will prefer fewer longer words. On the contrary if the language model probability increases (small penalty), the decoder will prefer a greater number of shorter words. These weightings are important in the experiment part to create the development data (see section 4.2.1).

Note 2. Different notations can be used depending on whether we speak about local or global posterior probability. Here is the explanation of the difference between the **local** and **global posterior probability**:

- Global posterior probability is $P(\mathcal{M}|\boldsymbol{O}, \theta)$ such that $\mathcal{M}$ is the model given the acoustic data $\boldsymbol{O}$ and the parameters θ.
- It is possible to express the global posterior probability in terms of local posteriors $P(q_l^n|q_k^{n-1}, \boldsymbol{o}_n, \theta)$ (where q_k^n denotes the specific state q_k of $\mathcal{M}$ at time n) and language model priors.

We have:

$$P(\mathcal{M}|\boldsymbol{O}) = \sum_{l_1=1}^{L} \sum_{l_N=1}^{L} P(q_{l_1}^1, \ldots, q_{l_N}^N, \mathcal{M}|\boldsymbol{O})$$

here the posterior probability of the state sequence and the modal can be decomposed into the product of an acoustic model and priors over models (language models):

$$P(q_{l_1}^1, \ldots, q_{l_N}^N, \mathcal{M}|\boldsymbol{O}) = P(q_{l_1}^1, \ldots, q_{l_N}^N|\boldsymbol{O}) P(\mathcal{M}|\boldsymbol{O}, q_{l_1}^1, \ldots, q_{l_N}^N) \quad (2.5)$$

$$\simeq \underbrace{P(q_{l_1}^1, \ldots, q_{l_N}^N|\boldsymbol{O})}_{\text{ac. model}} \underbrace{P(\mathcal{M}|q_{l_1}^1, \ldots, q_{l_N}^N)}_{\text{lang. model}}. \quad (2.6)$$

Note 3. (Explanation of **Hybrid HMM/ANN**)
With some modifications of equation (2.6), we arrive at

$$P(\mathcal{M}|\boldsymbol{O}) \simeq \sum_{l_1,\ldots,l_N} \left[\prod_{n=1}^{N} P(q_{l_n}^n|\boldsymbol{o}_{n-c}^{n+d}) \frac{P(q_{l_n}^n|\mathcal{M})}{P(q_{l_n}^n)} \right] P(\mathcal{M}). \quad (2.7)$$

Then the hybrid HMM is based on local posterior probability, where $\boldsymbol{o}_{n-c}^{n+d}$ is limited to local context.
With the Bayes rule, we can show that:

$$\frac{P(\boldsymbol{o}_{n-c}^{n+d}|q_{l_n}^n)}{P(\boldsymbol{o}_{n-c}^{n+d})} = \frac{P(q_{l_n}^n|\boldsymbol{o}_{n-c}^{n+d})}{P(q_{l_n}^n)}$$

Then we also have:

$$P(\mathcal{M}|\boldsymbol{O}) \simeq \sum_{l_1,\ldots,l_N} \left[\prod_{n=1}^{N} P(\boldsymbol{o}_{n-c}^{n+d}|q_{l_n}^n) \frac{P(q_{l_n}^n|\mathcal{M})}{P(\boldsymbol{o}_{n-c}^{n+d})} \right] P(\mathcal{M})$$

based on local likelihood, this is the standart HMM.

The difference between the hybrid and the likelihood approches lies at the local level. The hybrid system estimates local posteriors and is then discriminant at the frame level. The likelihood system estimates local probability density functions. Both system can give us an estimate of the global posterior.

2.2 Features Extraction

Features extraction is done by a speech signal processing. Its aim is to provide a compact encoding of the speech waveform. This encoding should minimize the loss information and provide a good match with the distributional assumptions made by the acoustic models. The final result is a feature vector in $\mathbb{R}^n$ whose dimensionality is typically around 40. Feature vectors are typically computed every 10 [ms] using an overlapping analysis window of around 25 [ms].

Many features extraction techniques are available, these include:

- Linear predictive cepstral coefficients (LPCC)
- Mel-frequency cepstral coefficients (MFCCs)
- Perceptual linear predictive coefficients (PLP)
- etc.

Thus the kind of feature depends on the technique, for example PLP features. PLP features are used in the experiment part (see section 4) that is why more informations are given in the next subsection.

There exists an other kind of features, called posterior features, which is a transformation of feature vectors by multilayer perceptron (MLP) (see section 2.2.2). They are presented in the second subsection.

2.2.1 Perceptual Linear Predictive Coefficients (PLP)

The goal of the PLP model is to describe the psychophysics of human hearing more accurately in the features extraction process. PLP is based on the short-term spectrum of speech. In contrast to pure linear predictive analysis of speech, perceptual linear prediction (PLP) modifies the short-term spectrum of the speech by several psychophysically based transformations.

First the signal is broken in regular frames ($\sim$10 ms) to obtain short time stationary. After the speech signal is weighted by a window functions which are functions with zero-valued outside of some chosen interval for example the Hamming window.

Definition 22. Generalized Hamming windows are of the form

$$h(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right)$$

where α and β are parameters and N the number of samples. The particular Hamming window, proposed by Richard W. Hamming, has coefficients $\alpha = 0.54$ and $\beta = 1 - \alpha = 0.46$.

Then discrete Fourier transform (DFT) changes the windowed speech segment into frequency domain. We obtain the short-term power spectrum:

$$p(h) = \mathrm{Re}[S(h)]^2 + \mathrm{Im}[S(h)]^2$$

where $S(h)$ is the short-term speech spectrum and h the angular frequency in [rad/s].
Finally the PLP technique uses three psychophysically concepts whose we do not explain in details:

- the critical-band spectral resolution,
- the equal-loudness curve,
- and the intensity-loudness power law.

2.2.2 Posterior Features

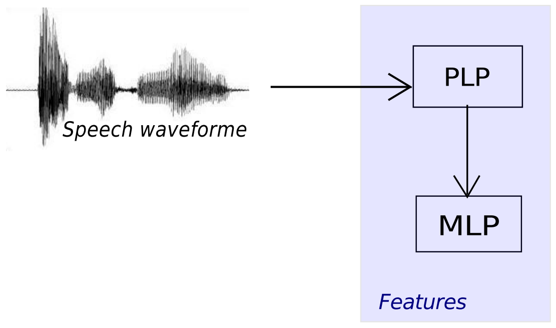

Figure 2.1: Graphical representation of PLP and MLP features.

A multilayer perceptron (MLP) is a feedforward artificial neural network (see section 1.3.4) model that maps sets of input data onto a set of appropriate outputs. Recall the three mains components of an ANN: the activation function, the architecture and the learning process. MLP has the same properties except that the information always moves in one direction (property of feedforward), thus the learning process is carried out through backpropagation.

The architecture of multilayer perceptron consists generally of three layers. The input layer is PLP feature and the output layer gives the MLP features (see figure 2.1), which are in our case phoneme scores.

2.3 N-Gram Language Models (LM)

Remember (equation 2.2) that language model is described mathematically by $P(W)$, where $W = w_1 \cdots w_K$ is a sequence of words. This priors are given by

$$P(W) = P(w_1 \cdots w_K) = \prod_{i=1}^{K} P(w_i | w_{i-1} \cdots w_1). \tag{2.8}$$

The conditioning word history is usually truncated to $N - 1$ words, where N is generally in the range of 2 to 4, to form a N-gram language model

$$P(W) = \prod_{i=1}^{K} P(w_i | w_{i-1} \cdots w_{i-N+1}). \tag{2.9}$$

These conditional probabilities can be estimated from N-gram frequency counts, i.e. maximum likelihood parameter estimates,

$$P(w_i | w_{i-1} \cdots w_{i-N+1}) \approx \frac{C(w_i \cdots w_{i-N+1})}{C(w_{i-1} \cdots w_{i-N+1})}$$

where $C(\cdot)$ represents the number of occurences of the word sequence.

Thus statistical language model (LM) are known as N-grams, where a 1-gram is an individual word, a 2-gram is a sequence of two words, and so on. Typically a statistical LM will contain probabilities for 1-grams, 2-grams and 3-grams, the probabilities of appearance of words and their dependence.

2.4 Pronunciation Dictionary / Lexicon

Pronunciation dictionaries play an important role in guiding the predictive powers of the ASR. A pronunciation dictionary or lexicon created for a specific language is a mapping of words to their corresponding pronunciation forms in terms of the phoneme sequences.

Consonants and vowels are used as the phonemes set for the creation of the dictionary. A sorted wordlist of the data is creating; then to each word from the wordlist is assigned a pronunciation. Afterward the audio version of the pronunciation is played back and the dictionary developer acts as a verifier to provide a verdict in the context of the accuracy of the word-pronunciation pair (correct, invalid, ambiguous or unsure). The developer removes or replaces phonemes in the predicted pronunciation until all word-pronunciation pairs are declared correct. For each adjustement, the system's learning algorithm extracts set of rules which would help to predict the next pronunciation. More there are rules, more accurate is the system.

2.5 Acoustic Models' Architectures

There exists different structures of acoustic model (AM). We are going to use hidden Markov model/Gaussian mixture model (HMM/GMM) and HMM tandem, both based on HMM (see section 1.2.1). The only difference between HMM/GMM and HMM tandem model is the input, i.e. the features, which is combined for the HMM tandem (see figure 2.2).

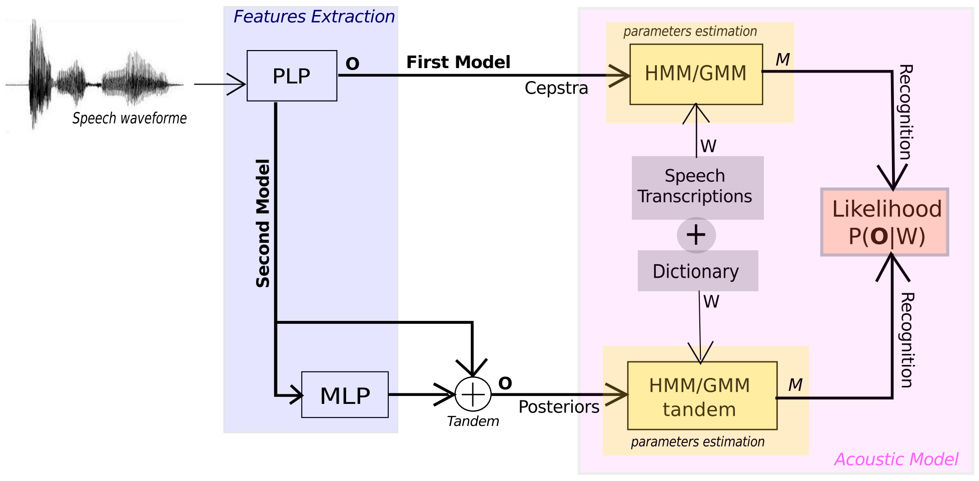

Figure 2.2: Architecture of the acoustic model (AM).

Acoustic Model (AM) is represented by the likelihood $P(\boldsymbol{O}|W)$ as explained in equation (2.2). This probability is obtained by compiling audio speech and text transcriptions to establish statistical representations of the sounds of each word (process called **training**). Then the best representation is chosen to identify the searched word (process called **recognition**). But before explaining HMM process, we need some tools.

Tools

Thanks to the dictionary, we have a list of words found in our data. Each word is decomposed into a sequence of basic sounds called phonemes to create different pronouciations. Thus the likelihood $P(\boldsymbol{O}|W)$ can be computed over multiple pronounciations

$$P(\boldsymbol{O}|W) = \sum_{Q} P(\boldsymbol{O}|Q)P(Q|W), \tag{2.10}$$

where Q is a sequence of word pronounciatons $Q_1 \cdots Q_K$. Therefore each pronunciation can be decoded as if they were alternative word hypothese.

Each pronouciation Q_i is a sequence of independant phonemes $Q_i = q_1^i q_2^i \dots$. Whence

$$P(Q|W) = \prod_{i=1}^{K} P(Q_i|w_i), \tag{2.11}$$

where $P(Q_i|w_i)$ is the probability that word w_i is pronounced by the phonemes sequence Q_i. In practice, there is a very small number of possible Q_i for each w_i it implies that (2.10) is easily tractable.

Each phoneme q is represented by a continuous HMM (see section 1.2.1) with transition parameter a_{ij}, $i, j \in \mathbb{N}$, and output observation distributions $b_j(\cdot)$, $j \in \mathbb{N}$, (see figure 2.3). Thus we have context-dependent phonemes, i.e. triphones.

Since there is a large number of triphones, and each triphone has several HMM states, and each state has many parameters, there ends up being too many parameters to efficiently work with. However, many triphone states represent a similar sound enough that they can be tied together to share the same set of parameters. These are called tied-triphone states.

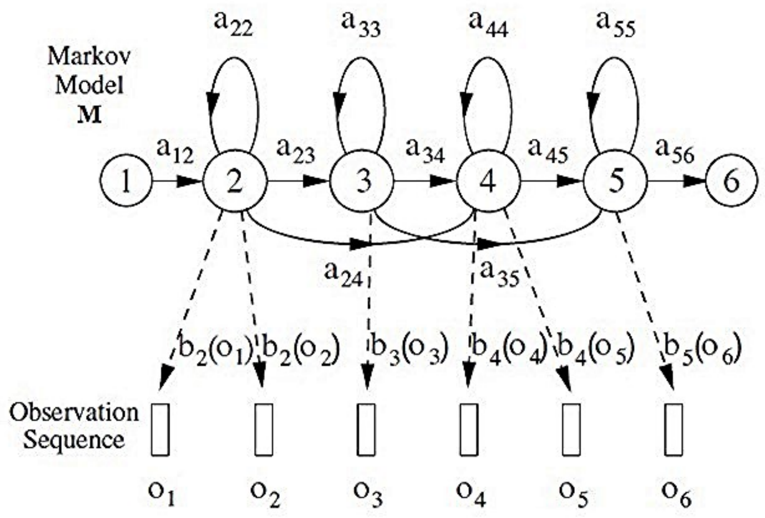

Figure 2.3: The illustration of the form of the hidden Markov model (HMM).

The output distributions (probability density function), $b_j(\cdot)$, are represented by Gaussian Mixture densities

$$b_j(\boldsymbol{o}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\boldsymbol{o}; \mu_{jm}, \Sigma_{jm}), \tag{2.12}$$

where c_{jm} is the weight of the m^{th} component, M the number of components (typically in the range of 10 to 20) and $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, that is

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}-\boldsymbol{\mu})} \tag{2.13}$$

where n is the dimensionality of $\boldsymbol{o}$. The parameters can be calculated by applying the Baum-Welch algorithm (see section 1.2.2).

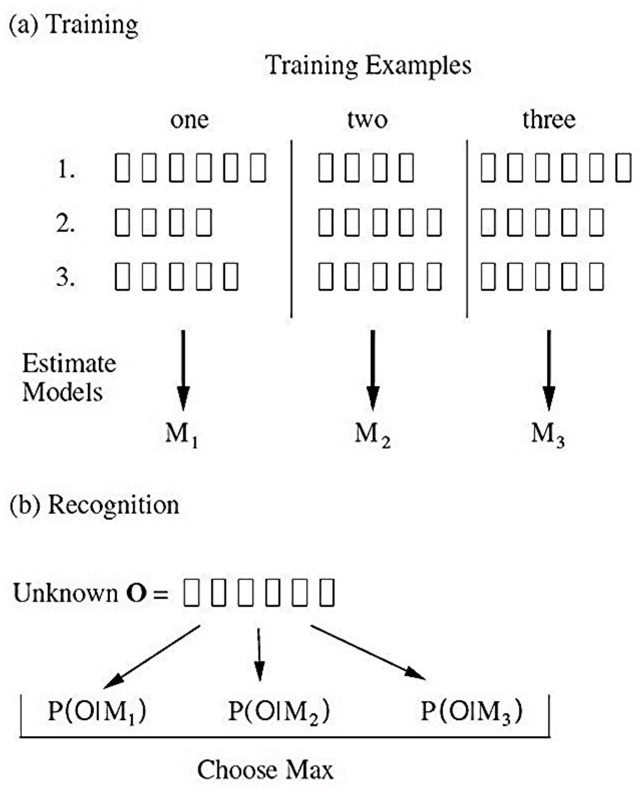

Figure 2.4: Illustration of the training and recognition processes in the AM.

Training (see figure 2.4)

An HMM $\mathcal{M}_i$ is created for each word using a number of examples of that word, thanks to the speech transcriptions and dictionary. Thereby, each word w_i has a corresponding model $\mathcal{M}_i$ which contains all triphones of w_i. We speak about training because the speech transcriptions represent training examples used to derive the best models.

Recognition (see figure 2.4)

The joint probability that $\boldsymbol{O}$ is generated by the model $\mathcal{M}$, moving through the state sequence $X = X_1 \cdots X_T$ is calculated simply as the product of the transition probabilities and the output probabilities

$$P(\boldsymbol{O}|\mathcal{M}) = \sum_X P(\boldsymbol{O}, X|\mathcal{M}) = \sum_X P(\boldsymbol{O}|X, \mathcal{M})P(X|\mathcal{M}) \tag{2.14}$$

where

$$P(\boldsymbol{O}|X, \mathcal{M}) = \prod_{t=1}^{T} P(\boldsymbol{o}_t|X_t, \mathcal{M}) = b_{X_1}(\boldsymbol{o}_1), \ldots, b_{X_T}(\boldsymbol{o}_T)$$

and

$$P(X|\mathcal{M}) = P(X_0 = X_1)a_{X_1X_2}a_{X_2X_3} \ldots a_{X_{T-1}X_T}.$$

However, only the observation sequence $\boldsymbol{O}$ is know and the underlying state sequence X is hidden (this is why it is called a hidden Markov model). Given that X is unknown, the required likelihood is computed either by summing over all possible state sequences $X = X_1X_2 \cdots X_T$, i.e.

$$P(\boldsymbol{O}|\mathcal{M}) = \sum_X a_{X_0X_1} \prod_{t=1}^{T} b_{X_t}(\boldsymbol{o}_t)a_{X_tX_{t+1}}, \tag{2.15}$$

or by the most likely state sequence

$$\hat{P}(\boldsymbol{O}|\mathcal{M}) := \max_X \left\{ a_{X_0X_1} \prod_{t=1}^{T} b_{X_t}(\boldsymbol{o}_t) a_{X_tX_{t+1}} \right\} \tag{2.16}$$

where we fix that X_0 is the entry state and X_{T+1} the exit one.

Equations (2.15) and (2.16) are not directly tractable because they require $(2T-1)N^T$ multiplications and $N^T - 1$ additions but simple recursive procedures, called forward algorithm (see section 1.2.3), exist to calculate efficiently.

Finally given a set of models $\mathcal{M}_i$ corresponding to words w_i, the equation (2.2) is solved assuming that

$$P(\boldsymbol{O}|w_i) = P(\boldsymbol{O}|\mathcal{M}_i). \tag{2.17}$$

2.6 Decoder

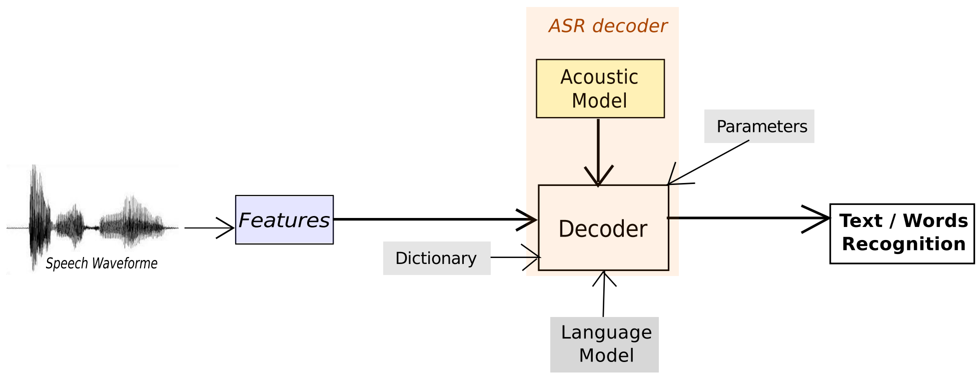

Figure 2.5: Overview of the ASR system.

Remember (see section 2.1.2) that the goal of the decoder is to search, given some observed acoustic audio $\boldsymbol{O}$, the most likely words sequence W so that

$$\hat{W} = \arg\max_W P(\boldsymbol{O}|W)P(W), \tag{2.18}$$

transformed as

$$\hat{W} = \arg\max_W P(\boldsymbol{O}|W)P(W)^s|W|^p \tag{2.19}$$

where s is the language model scaling factor and p the word insertion penalty (see equation (2.4)).

To do it, we define a search space which is defined by a finite state automata where the states are the phonemes and the transitions are defined in the acoustic models. The combination of the language model with the acoustic models produces an HMM that models all acceptable sequence of words (see figure 2.5). Given the search space, the most likely word sequence can be found by using the Viterbi algorithm (given in section 1.2.4).

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states, noted $\delta_t(i)$, which depend on a sequence of observed events, i.e.

$$\delta_t(i) = \max_{X_1X_2\cdots X_{t-1}} P(X_1X_2\cdots X_{t-1}, X_t = i, \boldsymbol{o}_1\boldsymbol{o}_2\cdots\boldsymbol{o}_t|\mathcal{M})$$

All paths through the search space are followed in parallel and are gradually pruned away using threshold such as the best path (minimum cost) emerges. For every time t, all the states are updated by the best score from all states in time $t-1$ for having only one best predecessor. By tracking back the best predecessors until the end of the search, the algorithm can determine the best state sequence for the entire search.

The complexity of the decoder is highly dependent on the complexity of the search space. In the case of phonemes, each word is obtained by concatenating sub-word models. So the search span can go from thousands of states to millions of states. The problem with Viterbi algorithm is when the search space contains a massive number of states because it becomes computationally infeasible. The complexity of the Viterbi algorithm is $\mathcal{O}(N^2T)$ with the assumption that every state can transition to every state at each step, where N is the number of state and T the length of the sequence. A solution, called N-best or multipass search, is to divide the decoding process into stages.

Multipass Search and N-best Paradigm

As the decoding progresses through the stages, the set of hypotheses is reduced so that more refined and computationally demanding knowledge sources can be used to produce the most likely sequence. This processing strategy is referred to as multipass search (i.e., cascaded decoding).

N-best paradigm is the most known multipass search strategy. The basic idea is to use computationally inexpensive knowledge sources to find N alternative sentence hypotheses. Then, each of these hypotheses is rescored with more expensive and more accurate knowledge sources in order to determine the most likely utterance.

The N hypotheses can be represented by a word graph, which edges are labeled by words, because it provides an explicit specification of word connections. Edges can also carry score information such as the acoustic and language model scores.

2.7 Evaluation of the Performance

A commonly metric used to evaluate the performance of ASR systems on the test data, i.e., not seen by the training process, is the word error rate (WER). For simple recognition systems, i.e. isolated words, the performance is simply the percentage of misrecognized words. However, in continuous speech recognition systems, such measure is not efficient because the sequence of recognized words can contain this three kinds of errors:

- the word substitution which happens when an incorrect word is recognised in place of the correct spoken one,
- the word deletion which corresponds to a spoken word not recognised,
- and the word insertion which occurs when extra words are estimated by the recognizer.

To estimate the word error rate (WER), the correct and the recognized sentence must be first aligned. Then the number of substitutions S, deletions D and insertions I can be estimated. The WER is defined as

$$WER = \frac{S + D + I}{|W|} \tag{2.20}$$

where $|W|$ is the number of words in the sequence of word W. Multiplied by 100, this gives the percentage of success.

Chapter 3

Confidence Measures (CM)

Since every speech recognizer will inevitably make some mistakes during recognition, the outputs from any automatic speech recognition (ASR) systems have some variety of errors. Thus, in any real-world application, it is extremely important to be able to make an appropriate and reliable judgement based on the error of ASR results. This problem requires to automatically assess reliability or probability of correctness for every decision made by the systems. This indicative score (preferably between 0 and 1) is called **confidence measure** (CM).

To resume we define confidence measures (CM) as a function which quantifies reliability of ASR results. It can be implement in different ASR processing levels as a basis for deciding the acceptance or rejection of specific hypotheses. These measures are based on one or more criterions.

These confidence measures (CM) are presented below:

- Acoustic CM derived exclusively from acoustic model (AM);
- CM as statistical model, here utterance verification is presented;
- Confidence estimation methods for neural network;
- Confidence Measures for Hybrid HMM/ANN.

3.1 Acoustic CM

Our approach to generate CM is based upon local estimates of posterior probabilities produced by a HMM system (see section 1.2.1).

The frame-based entropy is calculated with the results of the lattices with the function

$$H(\mathbf{X}, \mathcal{M}_L) = -\sum_{i=1}^{N_L} P(q_i|\mathbf{X}, \mathcal{M}_L) \log_{N_L}(P(q_i|\mathbf{X}, \mathcal{M}_L))$$

where N_L is the number of phonemes for language L, $\mathbf{X}$=observation, q_i = phonemes and $\mathcal{M}_L$=model. The logarithm in basis N_L is choosen to ensure that $H(X, \mathcal{M}_L)$ ranges from 0 to 1 regardless of the size of the phonemic inventory language L.
The higher the entropy, the higher the uncertainty of the model in describing the observation $\mathbf{X}$.

To estimated CM per utterance, we calculate the average of this entropy per utterance without the silent results.

3.2 Utterance Verification

Both out-of-vocabulary (OOV) word and unclear acoustics (accented or noisy speech) are a major source of recognizer error and may be detected by employing a confidence measure (CM) as a statistical hypothesis testing problem. Utterance verification is a post-processing stage to examine the reliability of the recognised hypothesis.

As before, we denote $\boldsymbol{O}$ a speech segment and we assume that the ASR recognizes it as a word $\tilde{W}$ which is represented by an HMM $\mathcal{M}_{\tilde{W}}$. The null hypothesis $H_0 = \{\theta_0\}$ and the alternative hypothesis $H_1 = \{\theta_1\}$ are defined as follows

$$H_0 : \boldsymbol{O} \text{ is correctly recognized and comes from model } \mathcal{M}_{\tilde{W}}. \tag{3.1}$$

$$H_1 : \boldsymbol{O} \text{ is wrongly classified and is not from model } \mathcal{M}_{\tilde{W}}. \tag{3.2}$$

$\boldsymbol{O}$ is distributed according to P_{θ_0} or P_{θ_1}, so that the parameters space is shrinked to $\Theta = \{\theta_0, \theta_1\}$. H_0 is tested against H_1 to determine whether or not we should accept the recognition result. According to Neyman-Pearson Lemma, the optimal solution to the testing (3.1) and (3.2) is based on a likelihood ratio testing (LRT), i.e.

$$\text{LRT} = \frac{P(\boldsymbol{O}|H_0)}{P(\boldsymbol{O}|H_1)} \underset{H_1}{\overset{H_0}{\gtrless}} \alpha \tag{3.3}$$

where α is the critical decision threshold.

Theorem 6. (Neyman-Pearson Lemma)
Let $\alpha \in (0, 1)$ be given. Then

1. We can find constant k, γ such that the following test ϕ^* has the required level $\alpha = \mathbb{E}_{\theta_0}(\phi^*(\boldsymbol{O}))$. This test has the form

$$\phi^*(\boldsymbol{O}) = \begin{cases} 1 & \text{when} \quad P_{\theta_1}(\boldsymbol{O}) \geq kP_{\theta_0}(\boldsymbol{O}), \\ \gamma & \text{when} \quad P_{\theta_1}(\boldsymbol{O}) = kP_{\theta_0}(\boldsymbol{O}), \\ 0 & \text{when} \quad P_{\theta_1}(\boldsymbol{O}) \leq kP_{\theta_0}(\boldsymbol{O}). \end{cases} \tag{3.4}$$

2. ϕ^* is the most powerful among all tests of level smaller that α.

3. Let ϕ be a test of level $\leq \alpha$, if ϕ has maximal power, it is necessary of the form given by 1., $\forall \boldsymbol{O} \in \{\boldsymbol{O}|P_{\theta_1}(\boldsymbol{O}) \neq kP_{\theta_0}(\boldsymbol{O})\}$. Moreover, ϕ is of level α unless there exists a test of level $< \alpha$ and of power 1.

3.3 Confidence Estimation Methods for Neural Network

Neural Network predictions suffer from uncertainty sources due to

- inaccuracies in the training data,
- limitation of the model and training algorithm.

This uncertainties involve two different noises: data noise variance, σ_v^2, and model uncertainty variance, σ_m^2. This two are independent but confidence estimation takes into account both sources. The total prediction variance is $\sigma_{\text{TOTAL}}^2 = \sigma_v^2 + \sigma_m^2$.

The problem is to estimate an unknow function $f(\mathbf{X})$ given a set of input-target pairs $D = \{X_n, t_n\}$, $\forall n = 1, \ldots N$. The targets are corrupted by additive noise $t_n = f(X_n) + \varepsilon_n$ where ε_n are modelled as Gaussian i.i.d. with zero mean and variance σ_v^2.

Two main approaches to predict **the data noise variance**, σ_v^2:

Maximum Likelihood (ML) Approach:

Aim. Predict σ_v^2.

Method: the traditional network architecture is extended by a new split-hidden-unit architecture. Only inputs and structure are in common.

We have

$$s^2(X_n) = \exp\left[\sum_k u_k h_k^{s^2}(X_n) + \beta\right],$$

where s^2 is the approximation of σ_v^2, β is the bias of the s^2 unit and $h_k^{s^2}(X_n)$ is the activation funtion of hidden unit k for input X_n in the hidden layer feeding directly into the s^2 unit.

All initial weights are drawn from an uniform random distribution $\in [-1, 1]$ scaled by the reciprocal of the number of incoming connections.
Warning: we do not update the weights, $\mathbf{u}$, of the variance network until $y(\mathbf{X})$ (the linear activation function which approximates $f(\mathbf{X})$) is somewhat close to $f(\mathbf{X})$.
Thus, after each pattern n is presented, all the weights in the network are adapted to minimize some cost function C (see definition 21). We obtain a form for C by maximize $\sum_n \ln P(t_n|X_n, \mathcal{N})$ where $\mathcal{N}$ is the network.
The least-squares regression techniques can be as maximum likelihood with an underlying Gaussian error model. We have:

$$P(t_n|X_n, \mathcal{N}) = \frac{1}{\sqrt{2\pi\sigma_v^2(X_n)}} \exp\left(\frac{-[t_n - y(X_n)]^2}{2\sigma_v^2(X_n)}\right).$$

After taking the natural log of both sides, some approximation and adding penalty terms, we obtain this cost function

$$C(\boldsymbol{w}, \mathbf{u}) = \frac{1}{2}\sum_{n=1}^{N}\left(\frac{[t_n - y(X_n, \boldsymbol{w})]^2}{\sigma_v^2(X_n, \mathbf{u})} + \ln[\sigma_v^2(X_n, \mathbf{u})]\right) + \frac{\alpha_w}{2}\sum_{i=1}^{W} w_i^2 + \frac{\alpha_u}{2}\sum_{j=1}^{U} u_j^2$$

where α_w and α_u are the regularisation parameters for weight $\boldsymbol{w}$ (the regression hidden layer connections) and $\mathbf{u}$ (the variance connections) respectively. Using this equation, we obtain our weight-update equations.

Bayesian Approach

Aim. Predict σ_v^2.

Method: we also extend split-hidden network but we obtain regression and variance estimates by using the bayesian approach with Gaussian approximation to the posterior.

First level: The starting point is the likelihood function $P(D|\boldsymbol{w}, \mathbf{u})$ which we combine with prior $P(\boldsymbol{w}|\alpha_w)$ using Bayes' rule to obtain:

$$P(\boldsymbol{w}|D, \mathbf{u}, \alpha_w) = \frac{P(D|\boldsymbol{w}, \mathbf{u})P(\boldsymbol{w}|\alpha_w)}{P(D|\mathbf{u}, \alpha_w)}.$$

By using generalised linear regression (GLR), this likelihood is a Gaussian with mean centred at the most probable weights $\tilde{\boldsymbol{w}}$ which can be found from minimising the error:

$$C_r(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\frac{[t_n - y(X_n, \boldsymbol{w})]^2}{\sigma_v^2(X_n, \mathbf{u})} + \frac{\alpha_w}{2}\sum_{i=1}^{W} w_i^2,$$

Second level: **Aim:** Define $P(\boldsymbol{w}|D, \tilde{\mathbf{u}}, \tilde{\alpha_w})$ which requires the knowledge of $\tilde{\mathbf{u}}$ and $\tilde{\alpha_w}$.

In this level, we want to find $\tilde{\mathbf{u}}$. We begin from:

$$P(\mathbf{u}|D, \alpha_w, \alpha_u) = \frac{P(D|\mathbf{u}, \alpha_w)P(\mathbf{u}|\alpha_u)}{P(D|\alpha_w, \alpha_u)}.$$

Like for first level, we finally arrive to find $\tilde{\mathbf{u}}$ by minimising:

$$C_v(\mathbf{u}) = \frac{1}{2}\sum_n \left[\frac{[t_n - y(X_n, \boldsymbol{w})]^2}{\sigma_v^2(X_n, \mathbf{u})} + \ln[\sigma_v^2(X_n), \mathbf{u}]\right] + \frac{1}{2}\ln|\mathbf{H}| + \frac{\alpha_u}{2}\sum_{j=1}^{U} u_j^2,$$

where $\mathbf{H}$ is the exact Hessian of error $C_r(\boldsymbol{w})$.

Therefore the network is trained in two phases by minimising $C_r(\boldsymbol{w})$ and $C_v(\mathbf{u})$ alternatively.

If either ML or Bayesian approach is used the **model uncertainty variance** σ_m^2 can be estimated using the delta estimator, a variant of the linearization method which is used for constructing confidence regions and confidence intervals. This method assumes that the nonlinear function can be adequately approximated by an affine or linear approximation to the function at the solution. The delta estimator gives

$$\sigma_m^2(\mathbf{X}) = \mathbf{g}^T(\mathbf{X})\mathbf{V}^{-1}\mathbf{g}(\mathbf{X})$$

where $\mathbf{g}(\mathbf{X})$ is a vector whose k^{th} element is $\partial\hat{y}(\mathbf{X})/\partial\hat{w}_k$ and $\mathbf{V}$ is the covariance matrix of weights $\boldsymbol{w}$. We can estimate the covariance matrix by the exact Hessian matrix

$$\mathbf{H} = \sum_{n=1}^{N} \frac{1}{\sigma_v^2(X_n; \hat{\mathbf{u}})} \frac{\partial^2 E_n}{\partial \boldsymbol{w}^2} + \alpha_w \mathbf{I}_n,$$

where $\mathbf{I}_n$ is the unitary matrix and $E_n = 1/2(t_n - \hat{y}_n)^2$ is the least-square error for data point X_n.

Or the outer product approximation to the Hessian

$$\tilde{H} = \sum_{n=1}^{N} \frac{1}{\sigma_v^2(X_n; \hat{\mathbf{u}})} \mathbf{g}(X_n)\mathbf{g}^T(X_n) + \alpha_w \mathbf{I}_n.$$

3.4 Confidence Measures for Hybrid HMM/ANN

See the note 3 in section 2.1.2 to have the explanation of an hybrid HMM/ANN. In the same section, recall the equation (2.7) by taking the maximum depending of the state sequence instead of the sum over it, we obtain

$$P(\mathcal{M}|\mathbf{X}) \simeq \max_{stat-seq}\left[\prod_n P(q_k^n|X_n)\frac{P(q_k^n|\mathcal{M})}{P(q_k)}\right] P(\mathcal{M}). \qquad (3.5)$$

We present three confidence measures that can be derived from equation (3.5).:

- **Scaled likelihood**:

$$CM_{sl}(q_k) = \sum_{n=n_s}^{n_e} \log\left(\frac{P(q_k|X_n)}{P(q_k)}\right) = \sum_{n=n_s}^{n_e} \log\left(\frac{P(X_n|q_k)}{P(X_n)}\right),$$

  where phone q_k has a hypothesised start time n_s and end time n_e.

- **Posterior**:

  $$\mathrm{CM}_{\mathrm{post}}(q_k) = \sum_{n=n_s}^{n_e} \log(P(q_k|X_n)).$$

  $\mathrm{CM}_{\mathrm{post}}(q_k)$ is computed by rescoring the optimal state sequence using the local posterior probability estimates.

  $\mathrm{CM}_{\mathrm{post}}(q_k)$ can be normalised by the duration of the phone in frames:

  $$\mathrm{CM}_{\mathrm{npost}}(q_k) = \frac{1}{n_e - n_s} \sum_{n=n_s}^{n_e} \log(P(q_k|X_n)).$$

- **Entropy**:

  $$\mathrm{CM}_{\mathrm{ent}}(q_k) = -\frac{1}{n_e - n_s} \sum_{n=n_s}^{n_e} \sum_{k=1}^{K} P(q_k) \log(P(q_k)).$$

  $\mathrm{CM}_{\mathrm{ent}}(q_k)$ is the entropy (see below) of K posterior phone probability estimates output by the ANN averaged over the duration of the phone.

  Definition 23. The entropy $H(X)$ of a discrete random variable X is defined by

  $$H(X) = -\sum_{X} P(X) \log(P(X)).$$

All these CMs may be applied at both the phone and word levels. The scaled likelihood and posterior CM are based on the most probable state sequence obtained by the Viterbi algorithm (see section 1.2.4).

Chapter 4

Experiments

This chapter presents three months trainee work in 2013 on development of German automatic speech recognition (ASR) system using a German part of multilingual database Mediaparl. The purpose of this chapter is exemplify the theory by training and testing the ASR with specific database. Both presented methods (see section 2.5), HMM/GMM and HMM tandem, are used in this experiment.

4.1 Mediaparl's Database

Mediaparl is a project developed by Idiap Research Institute, based in Martigny, which is a non-profit foundation that conducts basic research and development in the area of multimedia information management and it is affiliated with Ecole Polytechnique Fédérale de Lausanne. The goal of Mediaparl is to develop in parallel two distinct tools designed to help the Valaisan parliament in his process towards a parliament without paper. On one hand, an indexing and retrieval system that uses automatic speech recognition to index words pronounced by deputies during political debates. On the other side, an advanced automatic speech recognition transcription system that is able to identify spoken language and automatically transcribe French and German speech.

Mediaparl databases are french and german deputies interventions in the Valaisan parliament taken from Canal 9, a local TV company. There are now 4 years of raw data. Each sound file is associated with a .txt and .lab files. For training, we used the .lab files as it as been prepared using Gwenole's scripts for French and German tokenisation and normalisation. During data preparation, segments of less than three seconds or more than 50 seconds have been discarded. In addition end silence has been trimmed to not have more than one second at the end of the wav file. Finally, language identification has been done automatically, because speakers can switch language at any time. Here are some additional statistics:

- French:
  - 76'414 audio files or 170hours 38min 21sec,
  - 239 speakers.
- German:
  - 29'354 audio files or 73hours 30min 56sec,
  - 150 speakers.

Database's folder: The folder is divided into languages and years. Each folder name (speaker intervention) contains some meta data in it: i.e. 283_truetime2010september07_416_614F:

- 283 is the speaker id,
- truetime2010september07 map to ../database/sessions/2010/09/07,
- 416 start time in seconds,
- 614 end time in seconds,
- F is for French.

Folder content includes .wav (audio files), .txt (transcription files) and .lab files (prepared files using Gwenole's scripts). The .lab is normalised and ready for training (it is generated with the Mediaparl decoder and it's accuracy is around 90% until 2012, as sentences are included in the language model, and then 80%).

4.1.1 Preparation of Data

Lists: For this experiment, we used the german part of the Mediapal's database that is 150 speakers. The list of speakers has been split up into 3 files: training list, development list and test list (see section 2.7). The choosen distribution is 80% for training, 6% for development and 14% for testing.

Dictionaries: The first step in building a dictionnary is to create a sorted list of the required words. Each word must be associated with its own phonetic, for this step we used the German dictionary. Phonemes in the dictionaries are represented using the speech assessment methods phonetic alphabet (SAMPA) which is based on the international phonetic alphabet, but features onlay ASCII characters.

Two dictionaries are required for training, flat (basic) and main (alternative entries with sil and sp final phones) for alignment. These contains three columns, first is the word, second in bracets is the written word and the last column is the phonetic transcritption.

The entire data is divided into 3:

- 80% of **training data** (see section 2.5, the training part) for the models' creation.
- 6% of **development data** to estimate parameters (see section 2.1.2 about language model scale factor (LMSF) and word insertion penalty (WIP))
- and 14% of **test data** to evaluate the performance.

The distribution varies according to the amount of data, here is the one that we used for experiment (see section 4).

4.2 HMM/GMM Method

Summarize of supervised, unsupervised and semi-supervised models (see section 1.3):

- Supervised models:
    1. Train supervised acoustic models with transcribed audio;
    2. Tuning of the supervised models with the development list;
    3. Test of the supervised models with the test list.
- Unsupervised models: decode new untranscribed audio with supervised AM for obtaining decoded transcription.
- Semi-supervised:

1. Train semi-supervised acoustic models with “truth” transcribed audio and decoded transcribed audio;
2. Pruned and test the semi-supervised models.

4.2.1 Supervised Models

The data used for supervised models is ’true’ manual transcription with their corresponding audio.

Training Part

Training of acoustic models (AMs) (see section 2.5) was performed using the HTS tools, wrapped into Idiap Speech Scripts [1]. We used perceptual linear prediction (PLP) features (see section 2.2) which we extracted from the database.

The training program first aligns the words and its pronunciations, it’s mean that a matching is create between the word and its phonemes. Then, an estimation of each phoneme (monophone) is given by a single Gaussian. After a list of triphones is created and a reestimation is done, having still single Gaussian distributions. The HMM is trained to have in each state a statistical distribution that is a mixture of 16 Gaussians, which will give a likelihood for each observed vector. Finally to tie rare states, the program apples a minimum description length (MDL) decision trees. The set of Gaussian’s means and variances in each HMM state forms the AMs.

Tuning Part

Aim. Find out, using the development list, optimal decoding parameters like word transition penalties and language model scales (see section 2.1.2).

The effect of the word transition penalty is to modify the language model log probability according to the equation

$$\widetilde{P}(W)' = s\widetilde{P}(W) + p$$

where p is the word transition penalty when it is negative, and word confidence when it is positive (in the logatithmic domain), s is the language model probability scale factor and $\widetilde{P}(W) = \log(P(W))$. In general, when unknown speech is given as input to the recogniser, the number of insertions tend to increase and the number of deletions decrease, as p is increased (more positive and less negative) and s decreased. Alternatively, the number of insertions tend to decrease and the number of deletions increase, as p decreases (more negative and less positive) and s increases. This can be explained due to the fact that the token log-likelihood is decreased when p is a negative number, penalising transitions between words (phonemes), resulting in a decrease in the number of insertions and an increase in the number of deletions. The optimal values for s and p are usually found by maximising the recognition accuracy on a development set, for both of these parameters.

Tuning Results

We have

$$\text{\% of Correct Words} = \frac{\#\text{phoneme correcty recognised}}{\#\text{total labels}},$$

$$\text{Accuracy} = \frac{\#\text{phoneme correcty recognised} - \#\text{insertions}}{\#\text{total labels}}.$$

[1] github.com/idiap/ssp

- Pruning of LM Scale, s, :

Development List				
Word Transition Penalty	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
0	19	15.00	81.55	76.15
0	**17**	16.36	82.27	**76.51**
0	15	16.14	82.44	76.22

Table 4.1: Different results obtained in decoding the development list by varying the language model scale.

- Pruning of Word Transition Penalty, p,:

Development List				
Word Transition Penalty	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
0	17	16.36	82.27	76.51
-3	17	15.91	81.98	76.50
-7	17	16.14	81.94	76.92
-9	17	16.14	81.93	77.13
-11	**17**	16.36	81.86	**77.19**

Table 4.2: Different results obtained in decoding the development list by varying the word transition penalty.

Thus the parameters are worth 17 for LM scale and -11 for word transition penalty (see tables 4.1 and 4.2).

Testing Part

After the language model (LM) (see section 2.3) was trained from all transcriptions belonging to the training part of AMs, we can test the ASR with adjusted parameters.

Test List				
Word Transition Penalty	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
-11	17	17.29	78.54	75.11

Table 4.3: Results obtained for the test list with the pruned LM scale and word transition penalty.

We obtained a result of **75.11 %** of accuracy for the ASR with supervised learning (see table 4.3).

4.2.2 Unsupervised Models

The aim of unsupervised model is to decode new audio (here 52hours08) which don't have transcriptions files. This model uses the AM of the supervised models obtained in section 4.2.1 to get decoded transcriptions. We created a new dictionary and language model depending on the new audio data. We had to restrict the dictionary to 60'000 words because of HTK capacities. Unsupervised learning allows to expand the amount of data which will be use for semi-supervised model.

4.2.3 Semi-Supervised Models

Training Part

The training procedures for semi-supervised model is the same as supervised, the only difference resides in the data. For semi-supervised learning, we use merged data which are data used to supervised (in section 4.2.1) and those used to unsupervised (in section 4.2.2). Therefore there are audio with “truth” transcription plus audio with decoded transcriptions thanks to unsupervised model.

Testing Part

After pruning in the same manner as section 4.2.1, i.e. by testing different values, we obtained 19 for the LM scale and -9 for word transition penalty. The same test list as supervised testing is used to test the model.

Test List				
Word Transition Penalty	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
-9	19	16.22	76.78	73.68

Table 4.4: Results obtained for the test of the semi-supervised models with the test list.

The result is **73.68%** of accuracy (see table 4.4) which is less than supervised model. This can be explained by the fact that the transcriptions in supervised model are correct instead of semi-supervised which has decoded transcriptions with errors. So we need to prune decoded transcriptions to sort out bad ones.

Pruned New Audio

Aim. Application of a CM threshold to set ’pruned new audio’ and retraining semi-supervised models.

First HTK calculate the lattices which is a partially ordered set in which every two elements have a unique supremum and a unique infimum. The output is a folder containing each sentences in a file.

Secondly we calculate the frame-based entropy (see section 3.4) using the results of the lattices. The higher the entropy, the higher the uncertainty of the model in describing the observation. The output is also a folder containing each utterances in a file.

Then we estimate CM per utterance, i.e. we do the average of the entropy measure per utterance without the silent results. We remove the silence results; it means we take the length of the frame where the speech start until it ends so we remove the silence of the beginning and the silence of the end. This permits to have a representative average.

Finally we set a threshold which applies between 10% and 30% less sentences. Thus we obtain new pruned references.

Results with pruned new audio

We train the semi-supervised audio with the “truth” transciption and the pruned references following multiple thresholds always by adjusting parameters. Here are the results:

The threshold 0.2 gives 74% of untranscribed audio (whether 39 hours), 0.15 83% (whether 44 hours) and 0.3 54% (whether 28 hours). We increase the “unpruned” semi-supervised result (73.68% of accuracy see table 4.5) but it is always less than the supervised (75.11% of accuracy).

Test List					
Threshold	WTP	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
0.2	-13	17	16.09	77.04	74.16
0.15	-11	17	16.09	77.34	74.29
0.3	-3	17	16.36	77.96	74.33

Table 4.5: Results obtained for the test list of the retrain semi-supervised models with different threshold.

4.3 HMM Tandem

The difference with the HMM/GMM model is that PLP features are transformed with MLP of 3 layers (here we obtained dimension 351x1565x57). The same 3 different lists: 80% for training, 6% for development and 14% for test sets and the same dictionary are used.

4.3.1 Supervised Tandem Models

The output of the training supervised MLP is a .mat file of posteriors. After reducing dimentionality with the principal component analysis (PCA), we reduce the dimension of 57 into 42. The model is train with the same HMM/GMM but with posterior features.

Results

After pruning the parameters we obtain a result of **76.74%** of accuracy (see table 4.6), more than supervised training with PLP features.

Test List				
Word Transition Penalty	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
-11	19	17.82	81.06	76.74

Table 4.6: Results obtained for the test of the supervised tandem model with the test list.

4.3.2 Semi-Supervised Tandem Models

After using unsupervised learning to obtain decoded transcription of the unlabelled audio, we train the semi-supervised model with “truth” and decoded transcription. The same as in section 4.2.3 but with MLP features instead of PLP. The dimension of the 3 layers MLP is 351x3231x57. We observe the same phenomenon as for HMM/GMM model, the result, **75.89%** (see table 4.7), decreases compare to supervised learning.

Test List				
Word Transition Penalty	LM Scale	% of Correct Sentences	% of Correct Words	Accuracy
-11	19	17.15	79.78	75.89

Table 4.7: Results obtained for the test of the semi-supervised tandem model.

4.4 Conclusion of the Experiment

In conclusion, first we use HMM/GMM model for training supervised models with PLP features, used unsupervised learning for more data and we retrain the model with merged data. The semi-supervised result being worse, we try to pruned the decoded transcriptions which doesn't give significant improvement.

Secondly we use the MLP method with 3 layers. With this method, we can see a small improvement (see table 4.8).

Type of Model	**Audio**	**Accuracy**
Supervised	transcribed audio	75.11
Semi-Supervised	transcribed + decoded audio	73.68
Semi-Supervised	transcribed + 83% decoded audio	74.29
Semi-Supervised	transcribed + 74% decoded audio	74.16
Semi-Supervised	transcribed + 54% decoded audio	74.33
Supervised Tandem	transcribed audio	**76.74**
Semi-Supervised Tandem	transcribed + decoded audio	75.89

Table 4.8: Summarize of results obtain in the experiment.

Finally, time doesn't allow us to test MLP method with 5 layers which should promise better performance, and apply CMs of section 3.

Conclusion

Through this work, I discovered mathematics in a practical field such as artificial intelligence. Courses taken at the university allowed me to understand the subject although it was sometimes difficult to link theory to practice. I learned with this work to juggle with different experience papers on which the theory was often summarized.

I enjoyed my 3-month internship at Idiap Research Institute which allowed me to touch the area of research and bring about a practical side. During this internship I was able to improve my computer skills and I realize that without this, the practice goes not very far. One major difficulty I encountered was in the differences between computer scientists' and mathematicians' appellations. Recognize known methods through experiments was not easy. Another difficulty was the time because it taked long to run experiments.

The writing of this work has allowed me to better understand each step of speech processing and do not mix them. This is a subject that I found very interesting by mixing the different areas and the fact that it is current though it is dated 1956.

To conclude, the main thing learned is that there is still much to discover whether mathematics, computer sciences, or neurosciences and with my studies I simply learned the tools to try to get to do it autonomously.

Acknowledges

I would like to thank Idiap to permit me to do this master subject and therefore permit me to discover mathematics in the field of artificial intelligence. It was really interesting to deal with informatic, mathematics and neurosciences.

I thank Milos Cernak, my supervisor at Idiap, which helps me for all my trainee work and for understanding of ASR model. Experiment would not exist without him. Thank you for your time and your help.

I thank Christian Mazza, my supervisor at university of Fribourg, who accepted to follow me for this subject. He was valuable for understanding mathematical concepts.

References

Figures and Tables

- **Figures 1.1, 1.2** and **1.3**: taken from the paper: Couvreur C.; *Hidden Markov Models and their Mixtures*; 1996; Université catholique de Louvain.
- **Figures 1.4, 2.3** and **2.4**: taken from the book: Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D., Valtchev V. and Woodland P.; *The HTK Book (for HTK Version 3.4)*; 1999; Cambridge University.
- **Figure 1.5**: taken from `https://en.wikipedia.org/wiki/File:Neuron_Hand-tuned.svg`.
- **Table 1.1**: taken from the book: Wasserman Larry; *All of Statistics: A Concise Course in Statistical Inference*; 2005; Springer.

Books

- Wasserman Larry; *All of Statistics: A Concise Course in Statistical Inference*; 2005; Springer.
- Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D., Valtchev V. and Woodland P.; *The HTK Book (for HTK Version 3.4)*; 1999; Cambridge University.
- Huang X., Acero A. and Hon H.; *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*; 2001; Prentice Hall.
- Rabiner L. and Juang B.-H.; *Fundamentals of Speech Recognition*; 1993; Prentice Hall.
- Young S.; *HMMs and Related Speech Recognition Technologies.*; 2007; Springer Handbook of Speech Processing. J. Benesty, M. Sondhi and Y. Huang, Springer.
- Cover T.M. and Thomas J.A.; *Elements of Information Theory*; 1991; John Wiley & Sons.

Papers

- Hwang J.T.G. and Ding A.A.; *Prediction Intervals for Artificial Neural Networks*; 2012; Journal of the American Statistical Association; DOI: 10.1080/01621459.1997.10474027
- Mazza C.; *Decision Theory*; Course Summer Semestre 2008, Department of Mathematics; University of Fribourg.
- Revaz S. and Cernak M.; *Baseline System for Automatic Speech Recognition with French GlobalPhone Database*; 2012; Idiap Research Institute.

- Faria A.; *An Investigation of Tandem MLP Features for ASR*; 2007; TR-07-003.
- Couvreur C.; *Hidden Markov Models and their Mixtures*; 1996; Université catholique de Louvain.
- Nkosi M.C., Manamela M.J.D and Gasela N.; *Creating a Pronunciation Dictionary for Automatic Speech Recognition: a Morphological Approach*; 2003; Private Bag X1106.
- Adami A.G.; *Automatic Speech Recognition: From the Beginning to the Portuguese Language*; RS 95070-560.
- Gandrabur S., Foster G. and Lapalme G.; *Confidence Estimation for NLP Applications*; published in ACM Transactions on Speech and Language Processing (TSLP); 2006.
- Hermansky H., Ellis D.P.W. and Sharma S.; *Tandem Connectionist Feature Extraction for Conventional HMM Systems*; 2000; ICASSP-2000.
- Akamine M. and Ajmera J.; *Decision Tree-Based Acoustic Models for Speech*; EURASIP Journal on Audio, Speech, and Music Processing 2012; Springer.
- Bernardis G. and Bourlard H.; *Improving Posterior Based Confidence Measures in Hybrid HMM/ANN Speech Recognition Systems*; 1998; IDIAP-RR 98-11.
- Jiang H.; *Confidence Measures for Speech Recognition: A Survey*; 2005; doi:10.1016.
- Tibor F.; *Confidence Measurement Techniques in Automatic Speech Recognition and Dialog Management*; 2008; Technische Universität München.
- Willett D., Worm A., Neukirchen C. and Rigoll G.; *Confidence Measures for HMM-Based Speech Recognition*; Gerhard-Mercator-University Duisburg, Germany.
- Papadopoulos G., Edwards P.J. and Murray A.F.; *Confidence Estimation Methods for Neural Networks: A Practical Comparison*; 2000; ISBN 2-930307-00-5.
- Williams G. and Renals S.; *Confidence Measures Derived from an Acceptor HMM*; University of Sheffield.
- Williams G. and Renals S.; *Confidence Measures from Local Posterior Probability Estimates*; University of Sheffield.
- Nix D.A. and Weigend A.S.; *Estimating the Mean and Variance of the Target Probability Distribution*; 1994 IEEE; 0-7803-1901-X/94.
- Williams D.A.G.; *Knowing What You Don't Know: Roles for Confidence Measures in Automatic Speech Recognition*; 1999; University of Sheffield.
- Wang D., Tejedor J., Frankel J., King S. and Colás J.; *Posterior-Based Confidence Measures for Spoken Term Detection*; 2005; TIN2005-06885.
- Wessel F., Macherey K. and Ney H.; *A Comparison of Word Graph and N-Best List Based Confidence Measures*; University of Technology Aachen, Germany.
- Hennebert J., Ris C., Bourlard H., Renals S. and Morgan N., *Estimation of Global Posteriors and Forward-Backward Training of Hybrid HMM/ANN Systems*; 2007.
- Qazar C.S.; *Bayesian Error Bars for Regression*; 1996; PHD thesis, Aston University.
- Hermansky H.; *Perceptual Linear Predictive (PLP) Analysis of Speech*; 1990; Acoustical Society of America.

- Rabiner L.R.; *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*; 1989; IEEE.
- Cheng B. and Titterington D.M.; *Neural Networks: A Review from a Statistical Perspective*; 1994; Statistical Science.
- Wessel F. and Ney H.; *Unsupervised Training of Acoustic Models for Large Vocabulary Continuous Speech Recognition*; 2002; IEEE.
- Weintraub M., Beaufays F., Rivling Z., Konig Y. and Stolcke A.; *Neural-Network Based Measures of Confidence for Word Recognition*.