# Probability and Random Processes Project - Speech Processing

Ananya Sane - 2020102007 Dosapati Sri Anvith - 2020102015
Harish Umasankar - 2020102067 L Lakshmanan - 2020112024

## Abstract

This report is a summary of our findings regarding the usage, applications and importance of probability in the various fields of speech processing. We have mainly read a few papers, which have been referenced in the references section, and have summarised what we understand from their research and their implementations. The first paper discusses the use of probabilistic measures in speech recognition for Cantonese. Blah blah add some more probably…

## Introduction

Speech processing is a very vast field that has seen quite a bit of progress over the few years because of the development in hardware and software that can process speech signals. However, the underlying software relies on methods from probability theory that are used to detect, recognise, analyse and modify the given signals according to our needs and preferences.

The subject of speech processing is indeed a very large topic, and we have tried reading a few specific topics that relate to it and depend on probability theory.

Add more ig, why not

## Paper 1 (A probability decision criterion for speech recognition, C.K. Yu, P.C. Ching)

Paper discusses Cantonese, because each character is monosyllabic and the characters only take 4 phonetic forms, which makes end point location of a discrete word a reasonably certain process. A simple detection algorithm (Lai, Ching & Chan, 1987) based on segmental energy and zero-crossing rate is employed to estimate the end-point locations of an input utterance such that all significant acoustic events within the word are included. The digitized speech samples are sent to a bank of five bandpass filters with passbands of: (i) 15-500 Hz; (ii) 50-850 Hz; (iii) 850 Hz-l.2 kHz; (iv) 1.2-1.8 kHz; and (v) 1.8-3.2 kHz. The outputs of these along with the original wideband signal for 6 different sequences. These 6 sequences are divided into 16 segments with a 50% overlap.

These bands are then processed and computed appropriately to form ENERGY TIME PROFILES (ETPs). The ETP matrix is a 6x16 matrix which can be used to compare with reference patterns and decode/interpret based on minimum distance logic, which is a probabilistic estimate. These ETPs are created by using a normalised measure of energy, calculated by

$$LE_i(q) = log\frac{E_i(q)}{E_max}$$

Here, $E_i(q)$ is the energy of each segment $i$ in the sequence $q \in 1, ..., 6$

To reduce the complexity of the operations in matching the input sequence to a reference sequence, a method proposed by Lai, Ching and Chan(1987) suggests the division of the vocabulary into groups according to the phonetic labelling of their intial regions. The error due to this categorisation has been found to be negligible. Now, every input token will only be compared with the references in its group, which optimises the operation by a large amount.

A probabilistic approach can be adopted here instead of just traditional Euclidean distance used for decoding. Now, each spoken word can be considered to be characterized by 16 ETP vectors and each vector has six segmental energy elements obtained from the six bandpassed sequences. We can use a large database which contains the ETP characteristics of many instances of every word in the vocabulary to determine the input word with more accuracy.

As we initially have many instances of the same word being spoken in our database, representing them as vectors in a matrix could get very tedious due to the large amounts of data (If our vocabulary size is T and the number of examples of each word w have are S, then our distance matrix will have dimensions of $S \cdot TxS \cdot T$). The references for each segment are

created by the K means clustering algorithm from the examples we have in the database. The method then suggests the use of **temporary** codewords from the initial S tokens and then clustering them again to form Q final templates.

These templates are then compared with our input ETP vector, and this comparison will form a 3D probability table, as the initial templates itself formed a 2D matrix. We now take the measurements from this table and try to maximise the total probability of resemblance. The template which maximises this is considered to be the most probable word (say m). The template that gives us the second highest resemblance is taken as the second most probable word (say n).

If the difference measure between these two words is not greater than a specified threshold $\delta$, then the most probable word is taken to be our result. If not, then we perform another comparison which involves the variances of ALL 16 segments of the two words, i.e.

$$\sigma_{u^2} = \sum_{i=1}^{16} \{P_i(u, k_i) - \frac{1}{16} \sum_{l=1}^{16} P_l(u, k_l)\}^2 \qquad u = m, n$$

So the final conditioning is

- If $\sigma_m^2 - \sigma_n^2 \leq \beta$, then the recognised word is m
- Otherwise, no result.

The experiments conducted in this paper have positive results (an accuracy of about 96%), which goes to show that using a probabilistic measure to decode an input signal given a database is a good method for speech recognition.

## Markov Chains and Hidden Markov Models

Markov chains are basically a set of states that are connected to each other by non zero probability arrows like in a weighted directed graph. Except here, the probability distribution of the next state depends purely upon the current state, and nothing before it, i.e.

$$P(x_n|x_{n-1}, x_{n-2}, ...x_1) = P(x_n|x_{n-1})$$

This is called the Markov property. We can represent Markov chains using matrices and can find stationary states using Linear algebra, i.e. taking the given Markov chain as an adjacency matrix $A$ ($A_{i,j}$ represents probability of going from state i to state j) and finding the required eigenvectors.

Now, if we have states that we might never reach again in a certain random walk, we call those states **transient**. If there are states where we will inevitably reach again in a certain random walk, we call them **recurrent**.

A markov chain with no transient states is called an **irreducible** Markov chain. If it only has transient states, then we can divide that Markov chain into separate irreducible Markov chains, which leads to them being called **reducible**.

Example, check Gambler's ruin. We can call those irreducible chains as communicaion classes too.

If we have a Markov chain and we need to find the probability of reaching a state j from a state i after a certain number, say n steps. To do this we need to consider all the possible paths to reach that state in those many steps and add their probabilities (Chapman Kolmogorov Theorem). This is a little tedious, but if we notice, with an adjacency matrix $A$, this problem can be reduced to finding the i, jth element of the $A^n$.

$$P_{ij}(n) = A^n_{i,j}$$

Now, if we take $\lim_{n->\infty} A^\infty$, each element will denote the same value as before, but after infinite steps. This is like our stationary distribution (The eigenvector thing from before) and we will see that every row vector converges to the same row vector, which means that the final probabilities of each state are independent of the starting state. This only happens when certain conditions like irreducibility and aperiodicity are satisfied.

A hidden Markov Model (HMM) is the combination of a Hidden Markov Chain (A Markov Chain with states that we cannot observe) and a set of observed variables which depend on the states attained in our Hidden Markov Chain. The matrix that contains the probabilities of observed variables is called the **emission matrix**. This can be represented by joint and conditional distributions. So, if we represent our Hidden states as X and our observed variables as Y, then we can get an estimate of the order in which transitions took place in the hidden states by

$$Estimate(X) = argmax_{X=x_1,x_2...,x_n} P(X = x_1, x_2..., x_n|Y = y_1, y_2..., y$$

As we cannot find this directly, we can use Bayes' theorem.

$$Estimate(x) = argmax_{X=x_1,x_2...,x_n} \frac{P(Y|X)P(X)}{P(Y)}$$

We can neglect the denominator as it is independent of X. The $P(Y|X)$ term can be simplified into the product of multiple $P(y_i|x_i)$ terms as the observed variable at that time only depends upon the current state and no other state. Same for $P(X)$, which can be a product of $P(x_i|x_{i-1})$ type terms because of the Markov property. For $x_0$, we can use the stationary distribution. This leads us to

$$Estimate(x) = argmax_{X=x_1,x_2...,x_n} \prod_i P(y_i|x_i) \cdot P(x_i|x_i - 1)$$

The above mentioned ideas are for an order 1 HMM. This can be extended to an order n HMM.

Order refers to dependence on history. First order Markov property says that the future state depends only on the current. Second order Markov property says that the future depends on current state and the previous state. Mathematically,

First order: $P(st+1|st, ..., s2, s1) = P(st+1|st)$

Second order: $P(st+1|st, ..., s2, s1) = P(st+1|st, st-1)$

Similarly, the nth order Markov is following:

nth order: $P(st+1|st, ..., s2, s1) = P(st+1|st, st-1, ..., st-(n-1))$

For example, in a typical HMM for language translation, the first order Markov model defines the probability of the next word depending on the current word and the probability is zero for all previous words. Second order Markov model will specify the probability of current word depending on the current word and the previous word. The probability is zero for all previous words. Same applied to the nth order Markov model.

## Chapter 2: Microsoft tutorial : Statistical Speech recognition

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MC_He_Ch02.pdf

Current speech recognition systems use Hidden Markov Models for acoustic modeling. Every input speech signal is assumed to be part of a certain language and vocabulary, and so, every speech signal is first converted into a sequence of "feature vectors" which are equally spaced in time. This feature vector consists of a representation of the data in the chosen time interval of the input signal. All the feature vectors together give us the data corresponding to the input signal.

Using this feature vector sequence, we can now use the maximum likelihood criteria and a database of samples to deduce the exact word sequence that was spoken.

So, using conditional probabilities, we can mathematically define this as

$$S^* = argmax_s P(X|s) \cdot P(s)$$

Here, $P(s)$ is defined by our **language model**, while the conditional probability is determined using the **acoustic model**.

The language model has probabilities defined for each word depending on the type of distribution present in a given vocabulary. It can be calculated using a dataset (called the training text corpus) and counting the frequencies of occurences of all the words. This is in the case of isolated word recognition.

In the case of continuous speech recognition, we can assume that the word sequence is produced by an (N-1)th order Markov model to simplify our computations. This simplifies our computation to

$$P(S) = P(w_1)P(w_2|w_1)...P(w_{N-1}|w_1, ..., w_{N-2}) \cdot \prod_{m=N}^{M} P(w_m|w_{m-N+1},$$

The model can be estimated by using a training text corpus and counting the occurences of each word while also taking into account its position.

## Results and discussion

Code stuff here.

## Conclusion

## References


- A probability decision criterion for speech recognition, C.K. Yu, P.C. Ching

- https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MC_He_Ch02.pdf