**Introduction to Speech Recognition Techniques and Their Implementations**

**Tayfun Adakoğlu**

11253001

Department of Computer Engineering

Pamukkale University

May 20, 2016

*Senior Year Student Research Showcase Report*

Table of Contents

1 [Introduction 3](#_Toc451495498)

2 [Speech Recognition Process 7](#_Toc451495499)

3 [Speech Classification Techniques 8](#_Toc451495500)

3.1 [Dynamic Time Warping (DTW) 8](#_Toc451495501)

3.2 [Hidden Markov Modelling (HMM) 21](#_Toc451495502)

3.3 [Neural Networks 23](#_Toc451495503)

3.3.1 [Preparing the neural network for speech recognition: Signal Processing 26](#_Toc451495504)

3.3.2 [Multilayer Feedforward Neural Network 38](#_Toc451495505)

4 [Closure 42](#_Toc451495506)

# **Introduction**

Speech is possibly the most efficient way to communicate with each other. This also means that speech could be a useful interface to interact with machines. Having long researches on how to improve this type of communication, some remarkable, successful examples have come out. With the advances of in the field of electromagnetism, examples are the invention of megaphone, telephone and etc.

Even in the 18th century, people were doing experiments on speech synthesis. For instance, in the late 18th century, a man named Von Kemplemen developed a machine capable of ‘speaking’ words and phrases. Today, thanks to the evolution of computational power, it has become possible not only to develop, test and implement speech recognition(SR) systems, but also to have systems capable of real-time conversion of text into speech and speech to text. Sadly, despite the fact that good progress made on that field, the speech recognition process is still facing lots of problems, assuring to the fact that speech is not an objective phenomenon yet.

Speech recognition is the inter-disciplinary sub-field of computational linguistics which includes knowledge and research in the linguistics, computer science, and electrical engineering fields to develop methodologies and Technologies that enables the recognition and translation of spoken language into text by computers and computerized devices such as those categorized as smart Technologies and robotics. It’s also knows as “automatic speech recognition (ASR)”, “voice recognition” or just “speech to text(STT)”. Voice recognition is also used for both for the term “speaker recognition” and “speech recognition”

Basically, stages of speech recognition process in a computer environment;

1. User speaks into the microphone
2. Microphone captures the sound waves and generates electrical impulses (analog electrical signal).
3. Sound card converts acoustical signal to digital signal with Analog-To-Digital Converter it includes, typically using pulse-code modulation method. This is also called sampling.
4. Speech recognition engine examines the digital signal coming and either do its job converting the signal to phonemes and then words or directly trying to investigate words in terms of different methodologies developed.

Most common problems are based on the reality of recording samples never produce identical waveforms;

* Speaker variation: in this case exactly the same word is pronounced differently by different people because of age, sex, speed of speech, emotional conditions of the speaker, dialect variations. Even the throat shape or lung size of speaker affect the acoustic signal produced.
* Background noise: a noise environment can add noise to signal. Even the speaker himself could do this by the way he speaks.
* Suprasegmental aspects: as opposed to individual speech sounds, influence of intonation (for example in question sentences) and putting stress on syllables. These all influence the pronunciation of a word.
* Continuity of the speech: when we speak, rarely there is a break between words, this makes it very hard to detect individual words.
* Other external factors are: position of the microphone in respect to the speaker, direction of the microphone and many others.

These all factors make up the characteristics of the sound wave such as amplitude, length. However perceptual information relative to speech remains consistent across wave. Solution is to extract speech-related information.

To mention about human nature, the eardrum gets the pitches (how high or low a sound is) and convey them to the inner ear. Then, in the inner ear, pitch signal’s processed by 20,000 hair-like nerve cells that perform one of the most critical roles in our ability to hear. Each nerve cell has a natural sensitivity to a particular frequency of vibration. After detecting the frequency value, nerve cell will resonate with a larger amplitude of vibration. This increased amplitude causes the nerve cell to release an electrical impulse that passes the auditory nerve towards brain. In a process that is not clearly understood, the brain is capable of interpreting the qualities of the sound upon reception of these electric nerve impulses. So human ear is an astonishing transducer (i.e. signal form converter).

Neural networks are composed of simple computational elements working in parallel. The network function is determined largely by the connections between elements. We can train a neural network so that a particular input leads to a specific target output. Here, I only discuss the usability of two different types of neural networks (i.e. Feedforward neural network with back propagation training method, radial basis function neural network through MATLAB. With all of them, we simply try to classify the input samples to known output words.

Next section will give information about speech recognition process. I will be giving some basic ideas, problems and discussing about challenges of the speech recognition progress. And in third section I will focus on signal processing necessary for extracting relevant information from speech signal (i.e. the speech related ones). That’s called signal pre-processing here. The implementation of the neural network classifiers is a subject of fourth section. The conclusion remarks are given in the last section of the report.

# **Speech Recognition Process**

General structure is that;

Speech 🡪 Signal Preprocessing 🡪

Feature Extraction 🡪 Speech Classification 🡪

Output

**Recording speech**, which consists of the acoustic environment (i.e. a room tuned (so soundproofed), not including any electrical wires, plugs except for the microphones) such as acoustic studios and the transduction equipment (i.e. microphone, preamplifier and Analog-to-Digital(AD) converter have a strong effect on the generated speech representations. For example, we can have additional impact generated from reverberation in room or addictive noise.

**Signal pre-processing** have the purpose to deal with these problems mentioned above as well as deriving acoustic representations that are both good at separate classes of speech sounds (i.e. making us able to obtain the speech sounds) and effective at suppressing irrelevant sources of variation in signal (e.g. the change in amplitude, length or background noise of a speech doesn’t change the speech itself or what is speech about, we should get rid of these effects that causes the same two speech signal to differ).

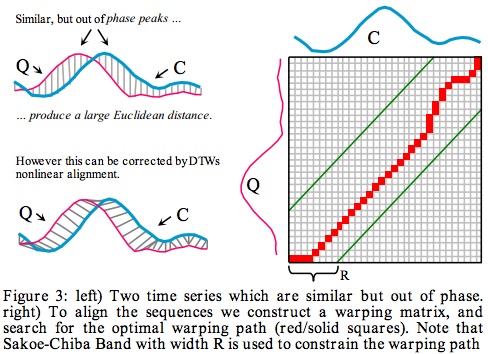
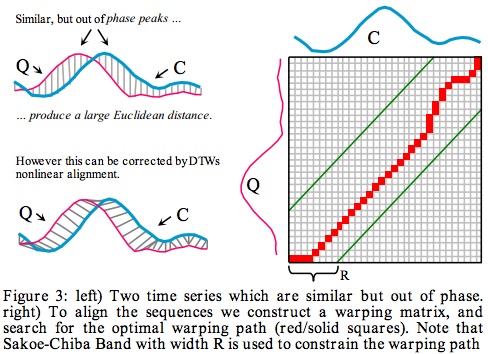
**Feature extraction** is all about extracting speech specific features of the pre-processed clear signal. This can be achieved with a variety of techniques such as cepstrum analysis and spectrogram. These techniques are also used in speaker detection systems.

**Speech classification** system tries to classify the extracted features and relates the input sound to the best fitting sound in a known vocabulary set and presents this as an output to us.

The performance of the speech recognition system finally depends on the classification techniques such as Artificial Neural Networks, Hidden Markov Models(HMMs), Bayesian Networks and Dynamic Time Warping and some hybrids of these techniques. Today, modern HMMs and Bayesian networks is used most in advanced systems and shows a state-of-art performance. I am only going to cover some neural networks and dynamic time warping because of the longness of the topics, so Bayesian networks, and hidden markov models are not included.

# **Speech Classification Techniques**

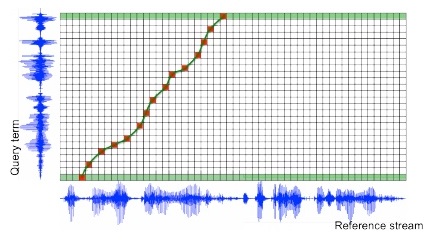
## **Dynamic Time Warping (DTW)**

This technique compares words with reference words. Every reference word has a set of spectra. There not exists distinction between separate sounds in the word. A time normalization is required because a word can be pronounced at different speeds. Dynamic Time Warping is a programming technique where the time dimension of the unknown word is changed (i.e. stretched and shrinked) until there is a similarity with a reference word. So it allows for elastic shifting in time domain and matches the sequences that are similar but out of phase (like shown in picture below). It could apply to time series to find the pattern in them. 

**The pros**: DTW only works with patterns, no need for costly transcriptions or knowledge of the language.

**The cons**: DTW compares patterns given a known start-end position which needs that at least one of the patterns be well bounded. It increases success that having both of the signal patterns well bounded.

However, thankfully, we are going to solve this boundary problem later by rebounding the both patterns with *Entropy Based Start-End Point Detection* Technique.



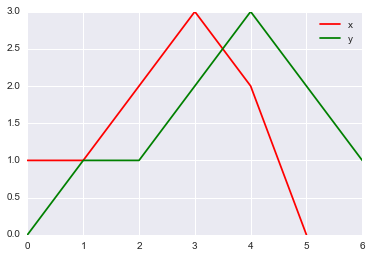
Consider two sequences of feature vectors taken for each second. Features could be anything here are amplitudes.

x = a1, a2, a3, …, ai, …, an

y = b1, b 2, b 3, …, b j, …, b n

Let A be [1, 1, 2, 3, 2, 0]

Let B be [0, 1, 1, 2, 3, 2, 1]



Graph includes feature vectors on y axis and time information on x axis.

Our aim is find a mapping (or distance) between all the points of x and y. For instance, x [3] may be mapped to y [4] and so on. MATLAB did that task showing the matrix of the Squared Euclidean distances between the pairs of the points. We are going to use squared distances. That’s for optimization of algorithm, not need for extra square rooting operation that is really hard and effort-requiring one for a computer (Actually rooting is the most time consuming operation in the computing).

[ [ 1., 1., 4., 9., 4., 0.],

[ 0., 0., 1., 4., 1., 1.],

[ 0., 0., 1., 4., 1., 1.],

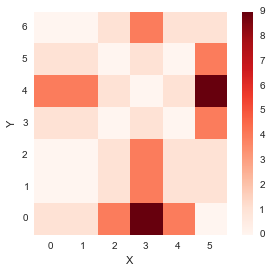
[ 1., 1., 0., 1., 0., 4.],

[ 4., 4., 1., 0., 1., 9.],

[ 1., 1., 0., 1., 0., 4.],

[ 0., 0., 1., 4., 1., 1.]]

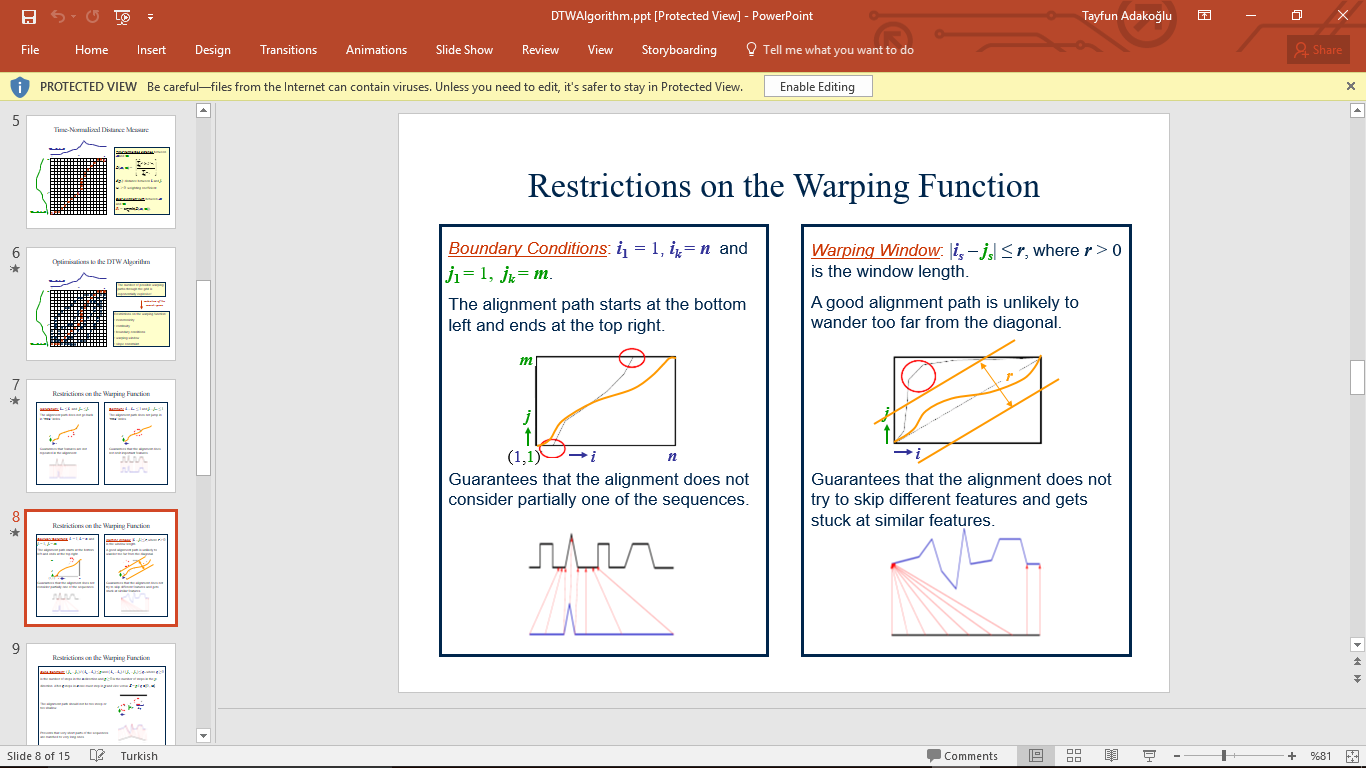
Now, let’s visualize the distance matrix with a function that achieves the job with shades of red.



From the plot above, it seems like the diagonal entries have low distances, which means that the distance between similar index point in x and y is low.

**Warping the path,** in order to create a mapping between the two signals we need to create a path in the above plot (Warping matrix). The path through the grid which minimizes the total distance between them is the best match or alignment between these two sequences. So, our goal is the find the path of minimum distance. The procedure for computing this overall distance involves finding all possible routes through the grid and for each one computing the overall distance. As you understand, that is really tough work to find. Thankfully, I am going to optimize (i.e. decreasing the number of paths we consider) this problem via dynamic programming and back-tracking algorithmic solution by imposing restrictions on the paths we would explore.

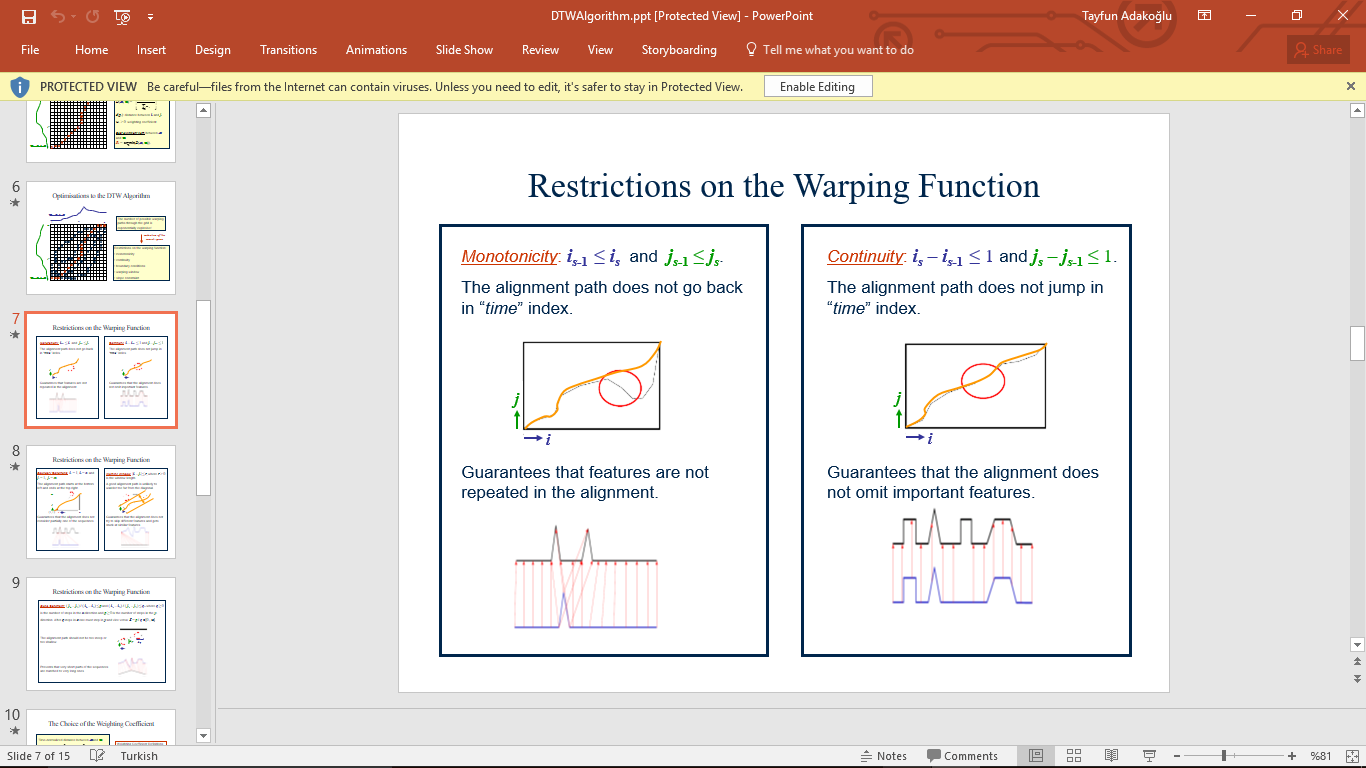
1. **Boundary condition**: The path must start at (0,0) (i.e. bottom-left) and end at (m, n) (i.e. top-right).



Guarantees that the alignment does not consider partially one of the sequences



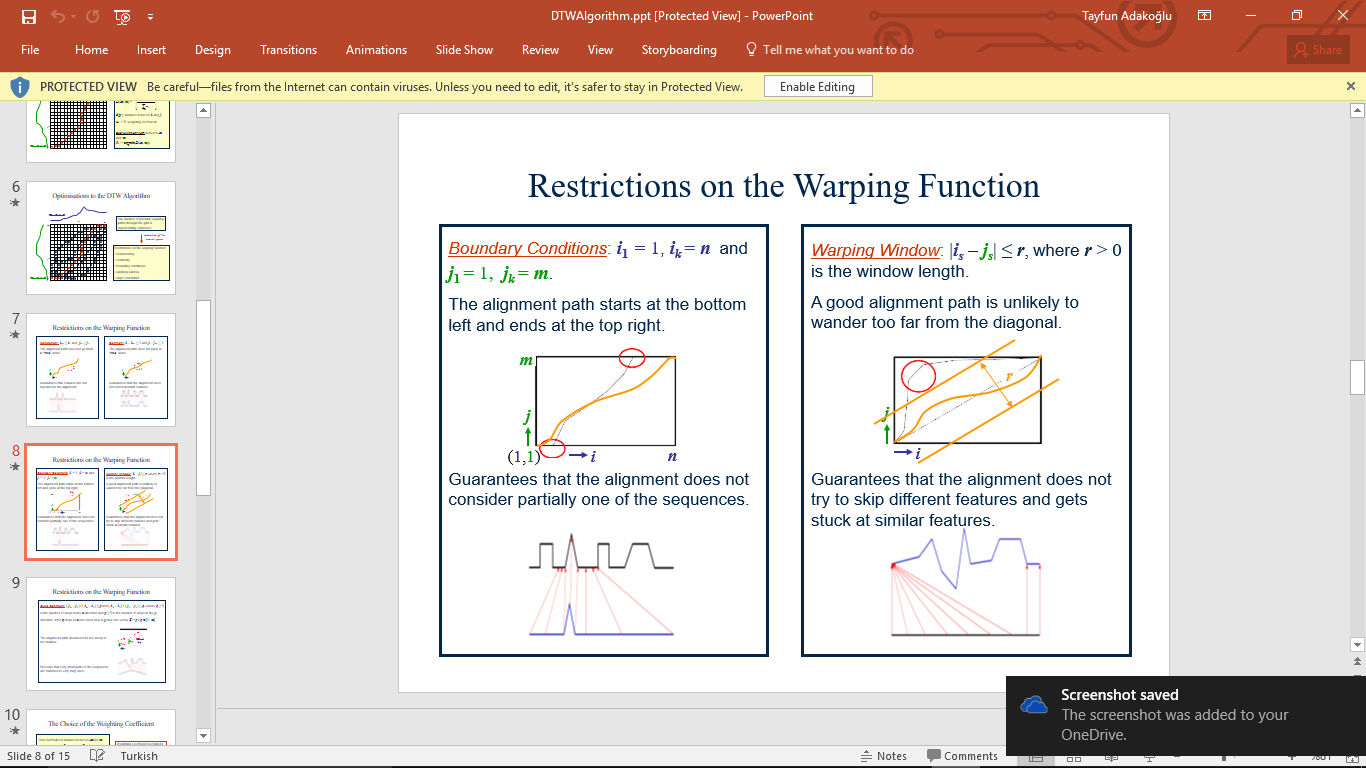
1. **Continuity condition**: The path advances one step at a time. Both i and j can only increase by at most 1 on each step along the path.



Guarantees that the alignment does not omit important features.



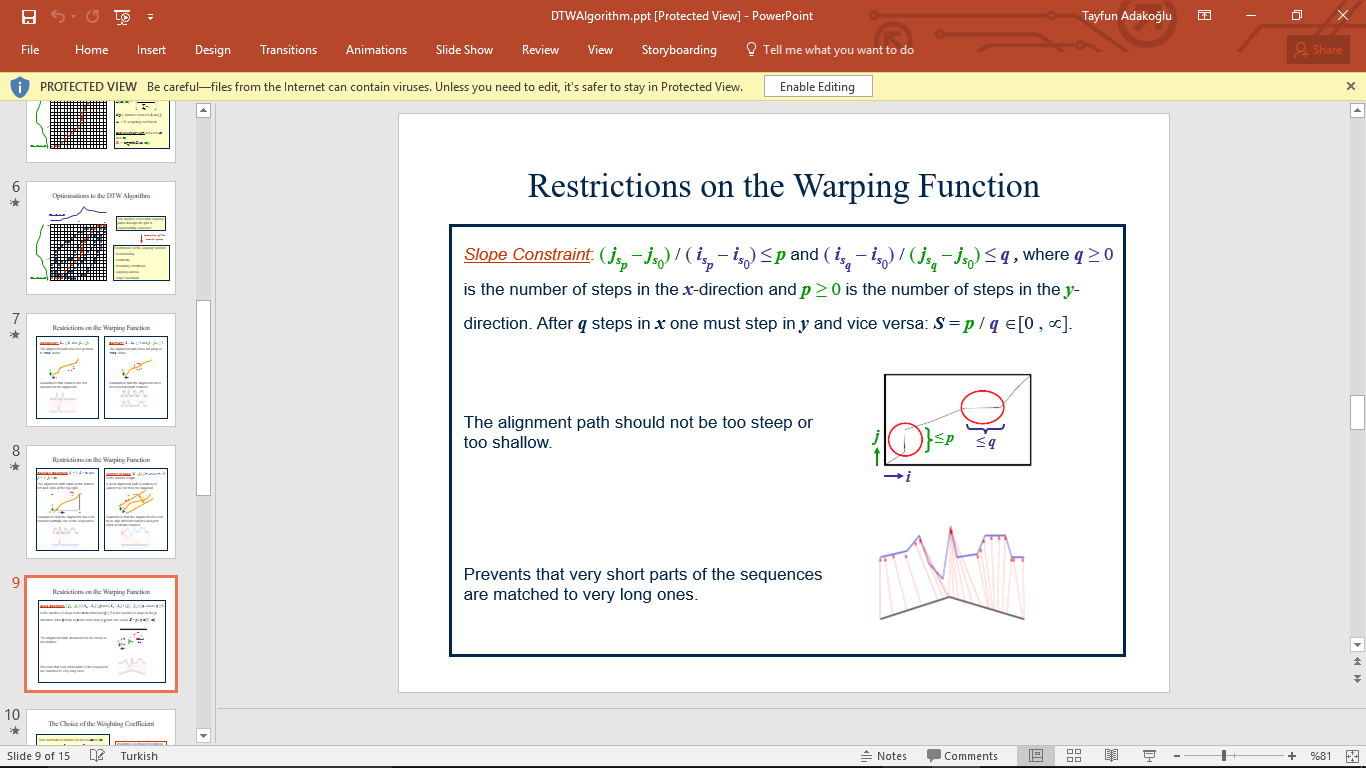
1. **Warping window condition**: A good path is unlikely to wander too far from the diagonal.



Guarantees that the alignment does not try to skip different features and gets stuck at similar features



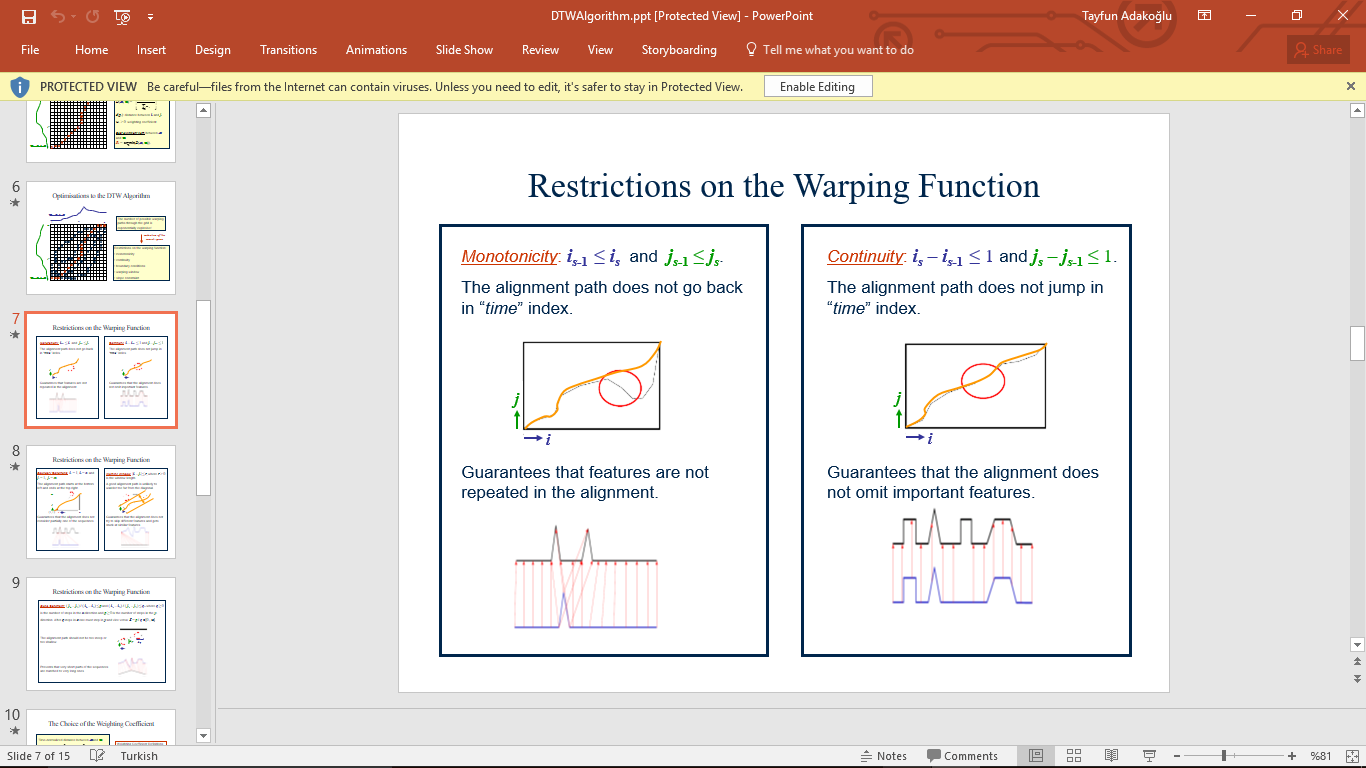
1. **Slope constraint condition**: The alignment path should not be too steep or too shallow according to some variables q and p where ***q*** ≥ 0is the number of steps in the ***x***-direction and ***p*** ≥ 0is the number of steps in the ***y***-direction.



Prevents that very short parts of the sequences are matched to very long ones



1. **Monotonic condition**: We cannot go back in time, so the path only flow forwards, which means that from a point (i, j), we can only right (i+1, j) or upwards (i, j+1) or diagonal (i+1, j+1).



Guarantees that features are not repeated in the alignment.



The power of the Dynamic Time Warping algorithm is in the fact that instead of finding all possible routes through the grid which satisfy the above conditions, it works by keeping track of the cost of the best path to each point in the grid.

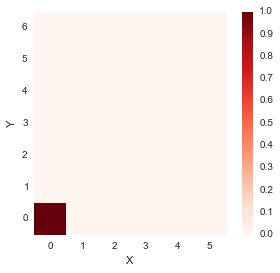
So, these restrictions prevent the combinatorial explosion and convert the problem to a Dynamic Programming problem which can be solved in O(m\*n) Big-Oh time which is faster than classical approach.

During the calculation of the DTW grid, it’s not known which path is minimum overall distance path, but this can be traced back when the end point is reached. So, the back tracking algorithm is used here.

**To start warping using the modern techniques mentioned above**, we build a matrix similar to the distances matrix. This matrix would contain the minimum distances to reach a specific point when starting from point (0,0). We are going to calculate the accumulated cost for each point regarding to (0,0).

Since we start from point (0,0), the accumulated cost at this point is distance (0,0)

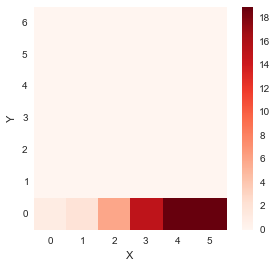
accumulated\_cost[0,0] **=** distances[0,0]



If we moved along the first row, i.e. from (0,0) in the right direction only, one step at a time;

**for** i **in** range(1, len(x)):

accumulated\_cost[0,i] **=** distances[0,i] **+** accumulated\_cost[0, i**-**1]

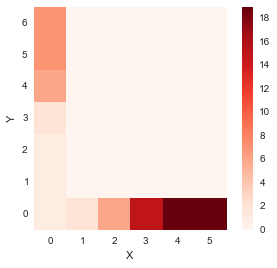


In addition to x axis, if we were to move along the first column, i.e. from (0,0) in the upwards direction only, one step at a time;

**for** i **in** range(1, len(y)):

accumulated\_cost[i,0] **=** distances[i, 0] **+**

accumulated\_cost[i**-**1, 0]



For all other elements we have,

Accumulated Cost (D(i,j))=min{D(i−1,j−1),D(i−1,j),D(i,j−1)}+distance(i,j)

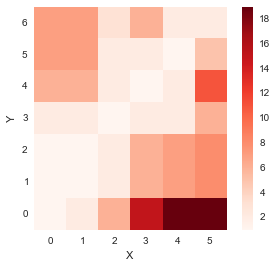
In MATLAB;

**for** i **in** range(1, len(y)):

**for** j **in** range(1, len(x)):

accumulated\_cost[i, j] **=** min(accumulated\_cost[i**-**1, j**-**1], accumulated\_cost[i**-**1, j], accumulated\_cost[i, j**-**1]) **+**distances[i, j]

All we did is the calculating the accumulated cost of each point simply by adding the distance of the point to the minimum of accumulated cost of previous element.



So, now we have obtained the matrix containing cost of all paths starting from (0,0). We now need to find the path minimizing the distance which we do by backtracking.

**Backtracking and finding the optimal warp path**; backtracking procedure is fairly simple and involves trying to move back from the last point (m, n) and finding from which place we would have reached there, by minimizing the cost. It also does this in a repetitive fashion until point (0,0).

In MATLAB;

path **=** [[len(x)**-**1, len(y)**-**1]]

i **=** len(y)**-**1

j **=** len(x)**-**1

**while** i**>**0 **and** j**>**0:

**if** i**==**0:

j **=** j **-** 1

**elif** j**==**0:

i **=** i **-** 1

**else**:

**if** accumulated\_cost[i**-**1, j] **==** min(accumulated\_cost[i**-**1, j**-** 1], accumulated\_cost[i**-**1, j], accumulated\_cost[i, j**-**1]):

i **=** i **–** 1 % meaning we get the min next point on i-1 point

**elif** accumulated\_cost[i, j**-**1] **==** min(accumulated\_cost[i**-**1, j**-**1], accumulated\_cost[i**-**1, j], accumulated\_cost[i, j**-**1]):

j **=** j**-**1 % meaning we get minimal next point on j-1 point

**else**: % meaning we should get down diagonally

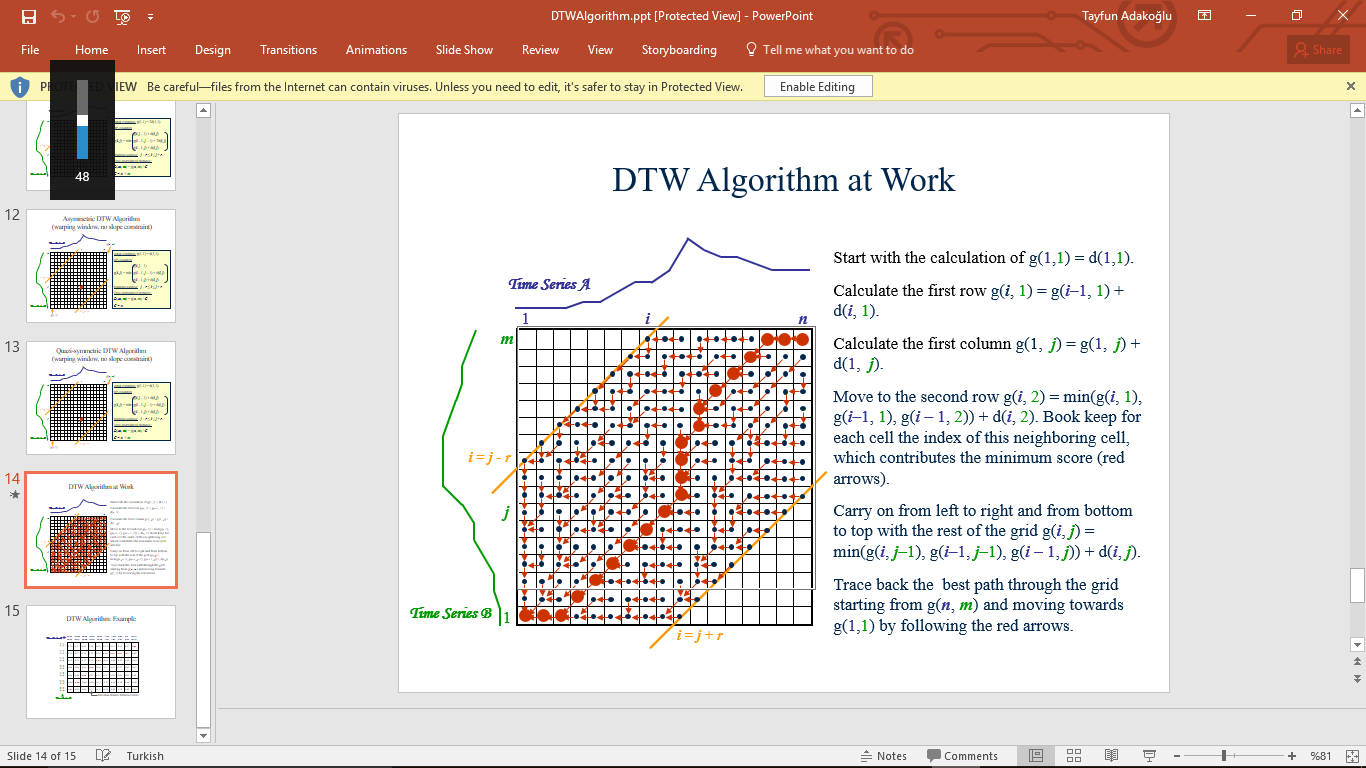
i **=** i **–** 1

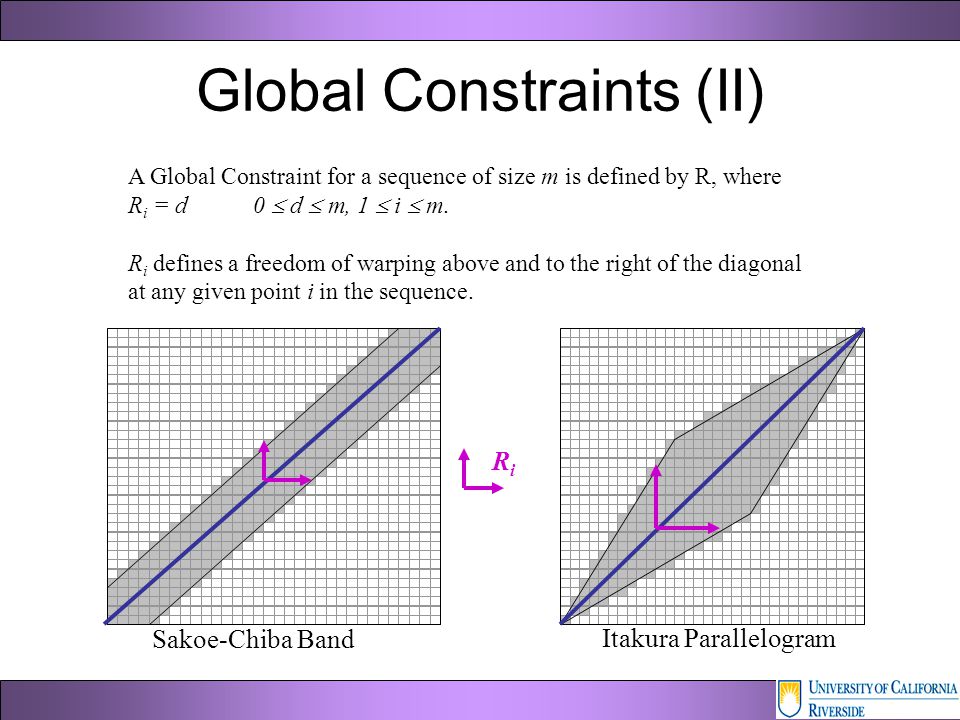
j**=** j**-** 1

path**.**append([j, i])

path**.**append([0,0]) %final step

To explain, the algorithm simply does that it goes downs to the next box where it finds the minimal costed point.



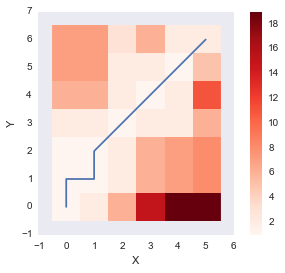


Bound limitation can be done in two ways.

There are limits, so algorithm is enclosed in here. We have explained the restrictions on which Dynamic Time Warping algorithm should depend.

Our output is shown in array and

[[5, 6], [4, 5], [3, 4], [2, 3], [1, 2], [1, 1], [0, 1], [0, 0]]



And the total cost of the path;

**for** [y, x] **in** path:

cost **=** cost **+**distances[x, y]

**return** cost

2 is the answer, i.e. the overall minimal cost.

The above plot shows the optimum warping path which minimizes the sum of distance (Dynamic Time Warping distance) along the path. Let’s finish the function by also incorporating the DTW distance between the two signals as well. **After finding**

[[5, 6], [4, 5], [3, 4], [2, 3], [1, 2], [1, 1], [0, 1], [0, 0]],

We look for all the points like 5 and x(5) vs 6 and y(6) for each one. Resulting;

5 0 : 6 1

4 2 : 5 2

3 3 : 4 3

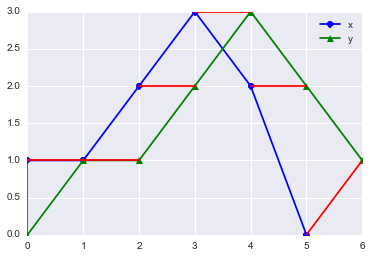
2 2 : 3 2

1 1 : 2 1

1 1 : 1 1

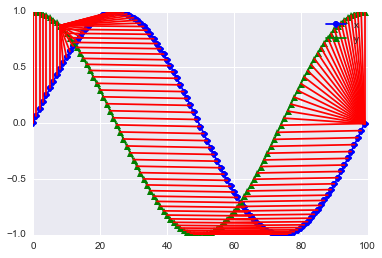
0 1 : 1 1

0 1 : 0 0



Red lines connect the matched points which are given by the DTW algorithm.

The latest situation independent of time info on non-lineer space.



## **Hidden Markov Modelling (HMM)**

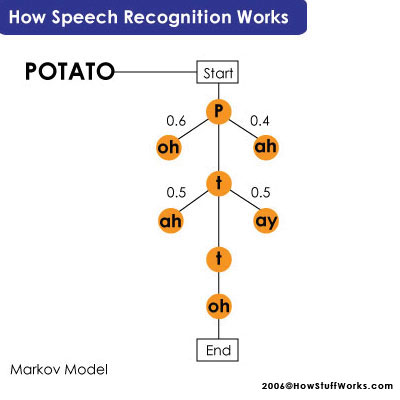
To introduce another technique of speech recognition:

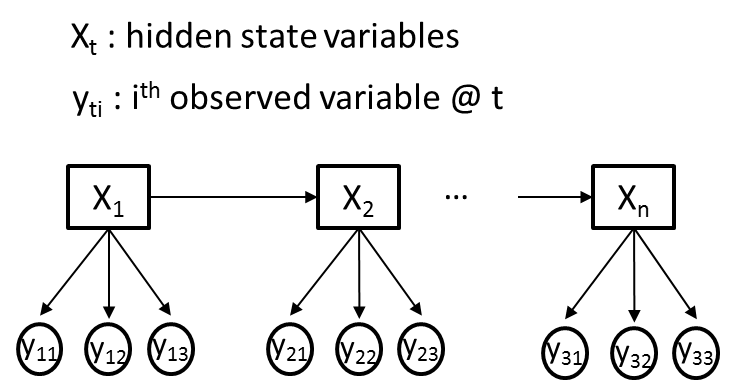
Until now, Hidden Markov Modelling is the most successful and most used pattern recognition method for speech recognition. According to John Garofolo, Speech Group Manager at the Information Technology Laboratory of the National Institute of Standards and Technology, the two models that dominate the field today are the Hidden Markov Model and neural networks.

These systems use probability and mathematical functions to determine the most likely outcome.

Hidden Markov Model is a mathematical model derived from Markov Model [[1]](#footnote-1). Speech is split into the smallest audible entities (i.e. not only vowels and consonants) but also conjugated sounds (i.e. gaining different forms) like ou, ea, eu, …

All these audible entities are represented as states in the Markov Model. This technique takes advantage of Graph Theory and Rules of Language being recognized too. As a word enters the Hidden Markov Model, it is compared to the best suited model. According the transition probabilities there exist a transition from one state to another. A state can also have a transition to its own if the sound repeats itself (e.g. the ‘u’ in word ‘book’). For example, the probability of a word starting with xq is almost zero. It also has to be that the sum of the transition probabilities from one state to others are one. Markov Models seems to perform quite well in noisy environments because every sound entity is treated separately. If a sound entity is lost in the noise, the model might be able to guess that entity based on the probability of going from one sound entity to another.





Here when we are given a lexicon, we could construct separate HMM models for each word in it.

I am not going to implement these model not to exceed some level of sophistication in this research…

## **Neural Networks**

Neural networks have many similarities with Markov Models. Each of them are statistical models which are represented as graphs. Where Markov models use probabilities for state transitions, neural networks use connection strengths and functions. A key difference is that neural networks are fundamentally parallel while Markov chains are serial. Frequencies in speech occur in parallel (i.e. frequency range through a boundary of time), while syllable series and words basically serial. That means that both techniques are very powerful in a different context.

While in the Markov model, the challenge is finding the appropriate transition (i.e. what the mostly next new audible sound could be) and observation probabilities (i.e. what the audible sound recognized should be), the neural network challenge is to set the appropriate weights of the connection.

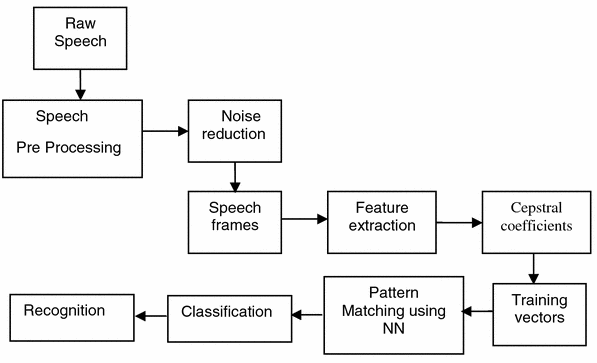
In many recognition systems, both techniques are implemented together and work in a symbiotic relationship.

Neural networks perform quite well at learning phoneme probability from highly parallel audio input (frequencies), as Markov models can use phoneme observation probabilities *that neural networks provide* to produce the likeliest phoneme sequence, word.

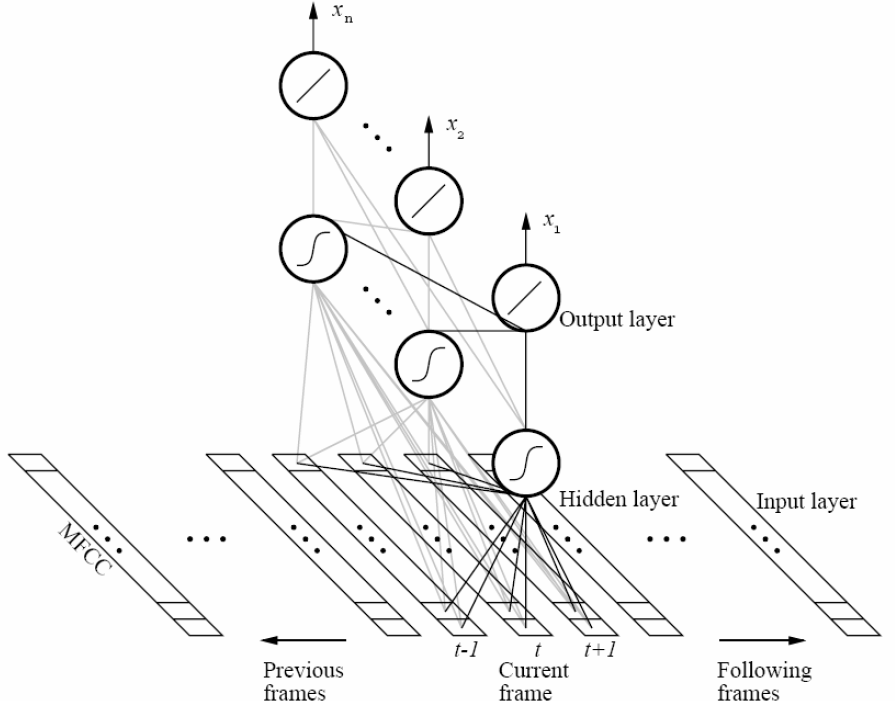
This is at the core of a hybrid approach to natural language understanding.

Here, speech features such as spectrogram and cepstrum will be sequentially presented as neural network inputs and then will be classified at the output of the network.

Here is the system overview of speech recognition based on neural network.



A visualization of classification process in the Neural Network. MFCC is accounted for ‘Mel-frequency Cepstral Coefficients’ which is used for sound modelling as we will explain them in detail later.



### **Preparing the neural network for speech recognition: Signal Processing**

Signal pre-processing has a big impact on the performance of speech classifier. It is important the feed the neural network with normalized input. Recorded samples never produce identical waveforms; the length, amplitude (i.e. a unit of amount of loudness like decibel), background noise might vary. Therefore, we need to perform signal pre-processing to extract only the speech related information. This means that using the right features have an importance of critical degree for successful classification. Good features simplify the design of a classifier whereas weak features (with little discrimination power) can hardly be balanced by any classifier.

The steps to signal pre-processing,

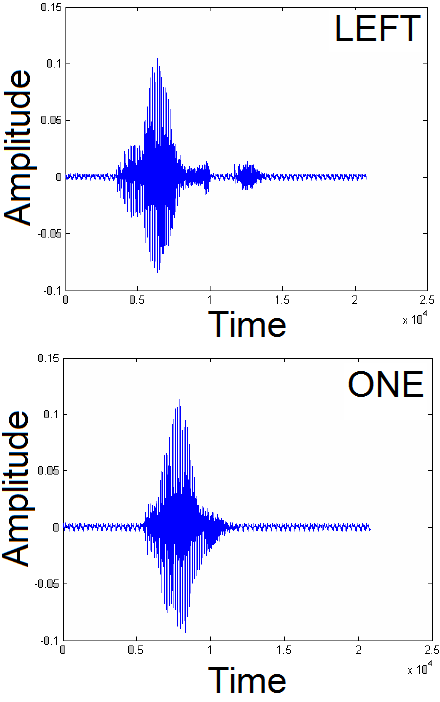
1. Representing the speech

Speech can be represented in different ways. Depending on the situation and the kind of speech information that needs to be present, one representation domain might be more appropriate than the other.

#### Waveform

This is the most general way to represent a signal. Variations of amplitude in time are presented. *The biggest disadvantage* of this method is that it cannot represent speech related information. A time-domain signal as such contains too much irrelevant data (no need for loudness info, remember like said in the beginning, a human ear transform signals from amplitude form to frequency based form and then send the info to brain) to use it directly for classification. No need for loudness information, remember like I said before in the beginning, a human ear transforms signals from amplitude form to frequency based form and then send the information to brain.

Figures below shows the time-domain representation of the words ‘left’ and ‘one’. It is clear that based upon this representation, it would be difficult to extract relevant speech information and thus cannot be used as inputs for the neural network classifier.



#### Spectrogram

There is a better representation domain for speech related information, namely the spectrogram. This representation domain shows the *change in amplitude spectra (i.e. range of amplitudes)* over time. It has three dimensions.

X-axis: Time(ms)

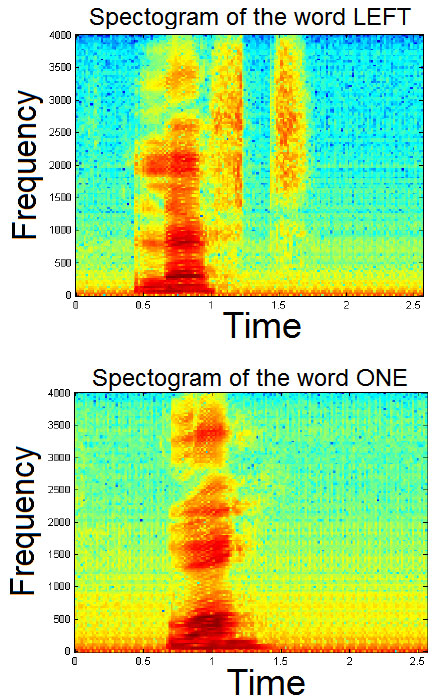
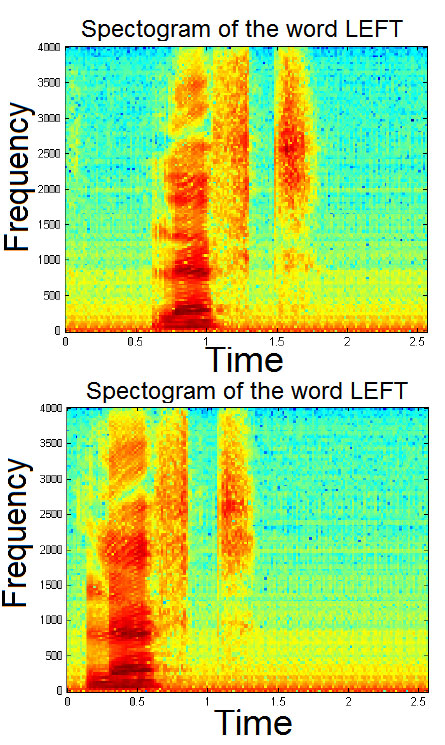
Y-axis: Frequency

Z-axis: Color intensity represents magnitude.

Spectrograms are calculated from waveform of a signal via Fast Fourier Transform(FFT) algorithm. I am not going to explain it in detail here.

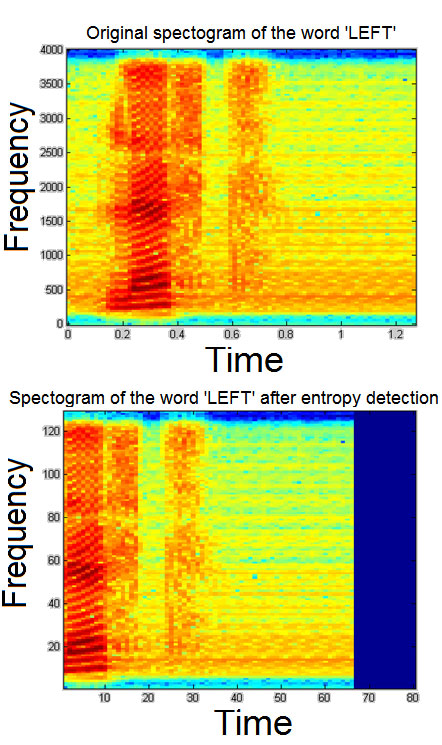
Although the spectrogram provides a good visual representation of speech, it still varies significantly between same samples recorded because samples never start at exactly the same moment, words may be pronounced faster or slower (i.e. causing the spectrogram wider or narrower) and they might have different intensities at different times due to magnitude.

Figures below represents two spectrograms of the word ‘left’ but they are calculated from different samples. As you can see, they both show somewhat the same pattern, but the second sample is shifted in time compared to first sample. As these patterns vary so much, makes them useless as input for the neural network unless some more signal pre-processing performed. I mean this is good representation to use but still not enough.

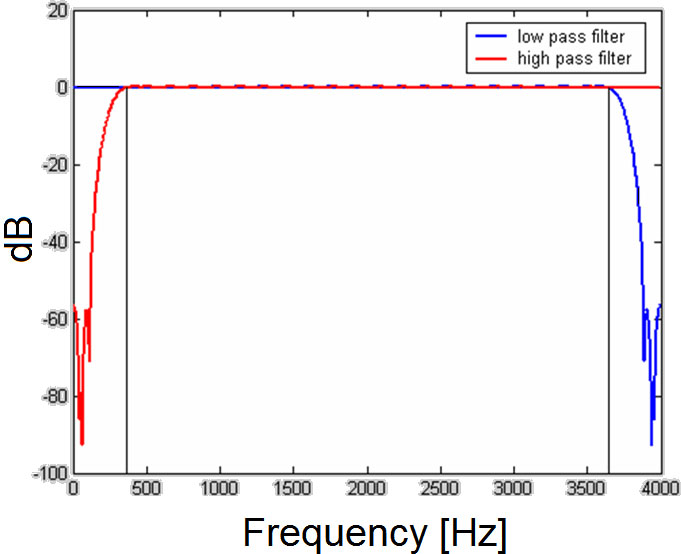
 

The second one here has a quicker start in time and was pronunciated faster a little bit somewhere than the first one, and all these vary.

Nevertheless, we can use spectrogram to train neural networks by force after applying Entropy based start-point detection technique on it. That technique allows the spectrogram to start at an appropriate area.



More filtering stages if required;

If samples are recorded with a standard microphone. So they contain besides speech signals a lot of distortion and noise due to the quality of the microphone or just because of picked up background noise. In this first step we perform some digital filtering to eliminate low and high frequency noise. As speech is situated in the frequency domain between 300 Hz and 3750 Hz, a bandpass filtering is performed on the input signal. This is done by passing the input signal successively trough a FIR low pass filter and then through a FIR high pass filter. An FIR filter has the advantage above an IIR filter that it has a linear phase response. The frequency response of the low pass and high pass filter is shown in figure below.

#### **Cepstrum and Mel Frequency Cepstral Coefficients**

We know that human ears, for frequencies lower than 1kHz, hear tones with a linear scale while for the frequencies higher than 1 kHz, hearing tones with logarithmic scale. The mel-frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. The voice signals have most of their energy in the low frequencies. It is also very natural to use a mel-spaced filter bank showing the above characteristics.

The mel scale is an attempt to find a measure of the human psychological sensation of pitch (i.e. perceptual frequency).

The following approximate formula is used to compute the mels for a given frequency in Hz.

C:\Users\tadak\Desktop\882986977de33cacddcf14e002a51458.png

For each tone with an actual frequency f (in Hz), a subjective pitch is measured on a scale called the ‘mel’ scale.

*Cepstrum* is the Inverse Fourier Transform(IFT) of the logarithm of a spectrum (also means Forward Fourier Transform). It’s also named by the reverse of ‘Spec’ as Cepstrum. It is thus the spectrum of a spectrum and has certain properties that make it useful in many types of signal analysis. One of its more powerful attributes is the fact that any periodicities, or repeated patterns in a spectrum will be sensed as one or two specific components in the cepstrum.

The difference between the cepstrum and the mel-frequency cepstrum is that in the mel-frequency cepstrum, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum.

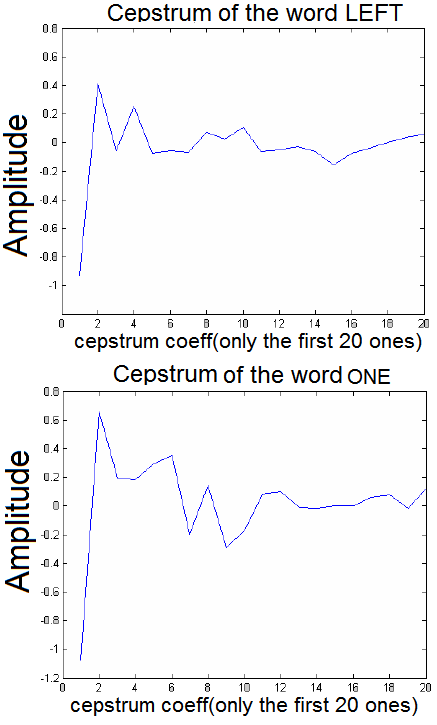
Therefore, mel-frequency cepstral coefficients are better for representing the speech with respect to human sensation.

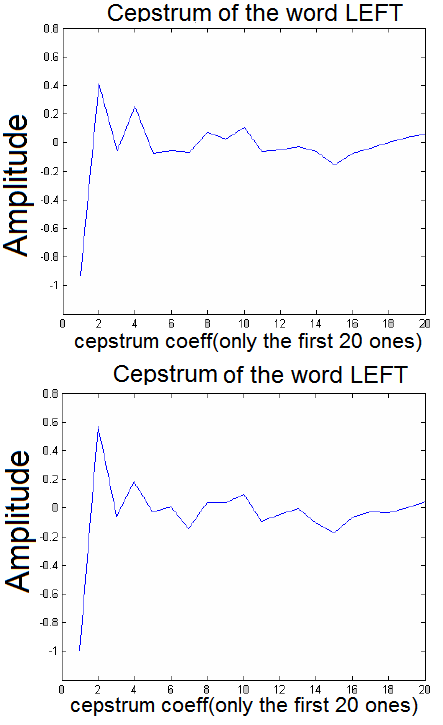
*To recap;*

**Inverse Fourier transform of the log of the Fourier transform of a signal waveform using the Mel Scale results in Mel Frequency Cepstral Coefficients.**

**Here, I am not going to mention about Fourier transforms’ mathematical procedures.**

Figures below the cepstrum of the words ‘left’ and ‘one’ is shown. Both charts show a different shape characteristic for that specific word. We discussed that the spectrogram has time dependent problems and the cepstrum is an ideal method for coping with these problems.

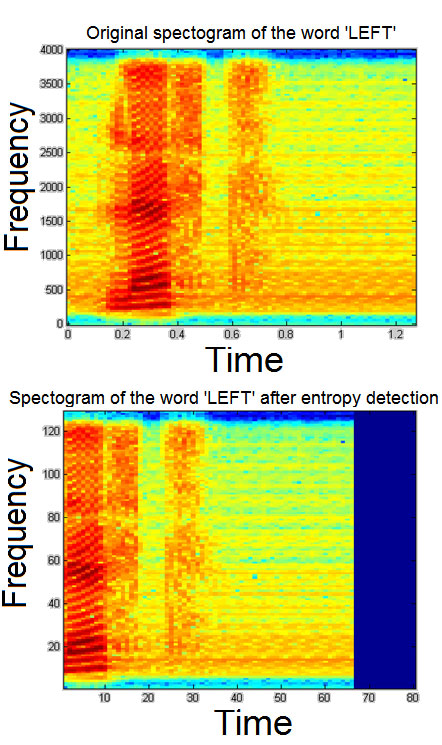




The different samples of same word produced nearly same result though noise, different speaker pronunciations and other different conditions between them.

*A cepstral analysis is a popular method for feature extraction in speech recognition applications, and can be accomplished using the Mel Frequency Cepstrum Coefficients Analysis (MFCC).*

#### **What to Use to Train Neural Networks? (Mel-frequency coefficients vs spectrogram values)**



For example, if we were to use spectrogram to train the neural networks; these spectrogram contain 80 time frames which contain 129 frequencies each. This means a total of 80x129=10320 points which is too large to use them all as input for the neural network. Therefore a selection resulting in a smaller set of points is necessary. Therefore using the Mel Frequency Cepstrum coefficients is a better strategy.

Taking many coefficients yields to a better approximation of the signal (more details), but becomes more sensitive to small variations of the different input samples. Using a fewer coefficients results

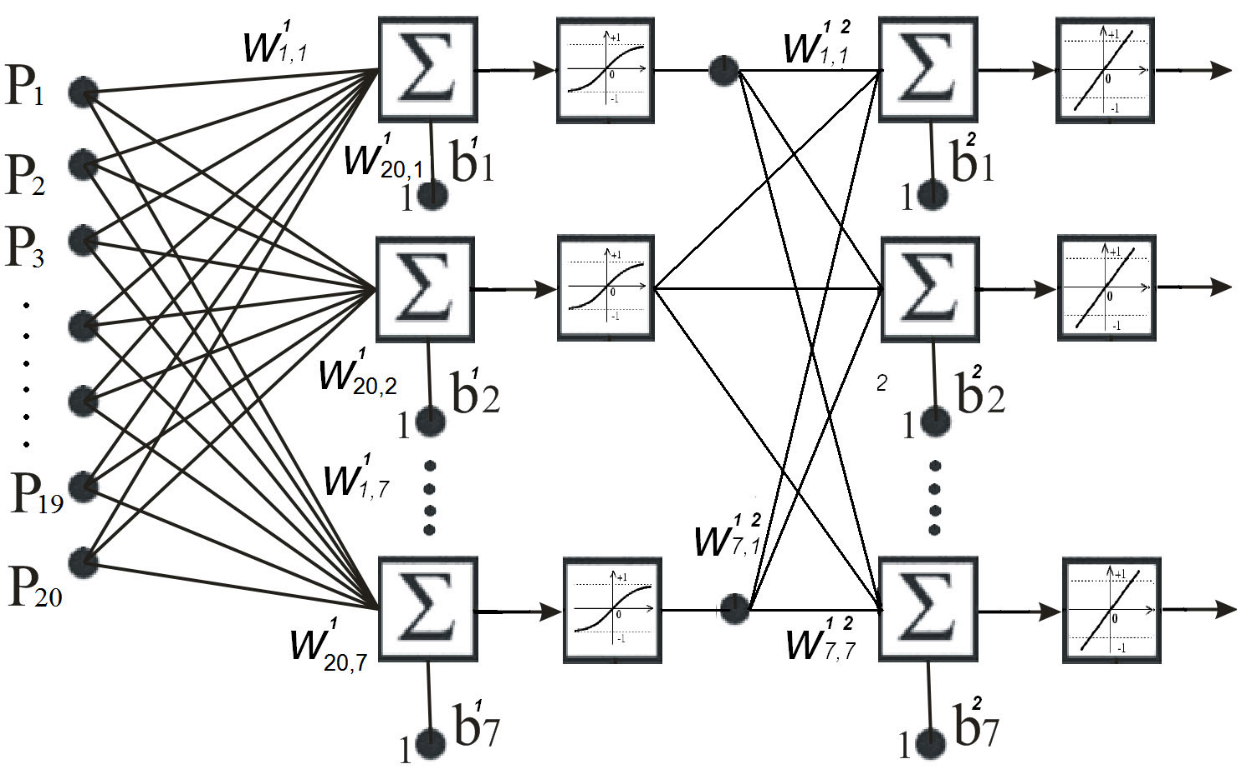
into a rougher approximation of the speech signal. An amount of 10 to 20 coefficients is optimal as input for the Neural Network.

### **Multilayer Feedforward Neural Network**

The fundemantal type of of neural nets used for speech classification is a Multilayer Feedforward Network using Back Propagation alghorithm for training. This type of NN is the most popular NN and is used worldwide in many different types of applications. Our network consists of an input layer, one hidden layer and an output layer.

We already discussed into detail the importance of the consistency of the neural network inputs. Feeding the NN with all data points from the spectrogram would be too much, as the spectrogram consists of 80x129=10320 data points the NN would require the same amount of inputs. Therefore we use a set of Mel Frequency Cepstrum coefficients as input for the neural network. As we only need ten up to twenty of them to represent a word, the neural network will only have 10 to 20 inputs like seen in figure below.

The input values are in a range of -5 up to 1.5. For every input neuron this parameter is set. In our design we put all these input ranges in a 'InputLayer' variable matrix.



Therefore we select a much smaller set of data points from each spectrogram to train NN. For every selected time frame we pick some frequencies. Taking 8 time frames with 10 frequency points each results in a NN input of 80 values, which is still a large input set, but with lesser dimension than the full spectrogram.

The hidden layer consists of **non-linear sigmoidal activation function** neurons. The amount of neurons in hidden layer depends on some factors like the *amount of input data* and *output layer neuron number*, the needed generalization capacity of the network and *the size of the training set*.

First the Oja rule of thumb is applied to make a first guess on how many hidden layer neurons are required.

First the Oja rule of thumb is applied to make a first guess on how many hidden layer neurons are required.

For example if we want to recognize five words with a training set of 100 samples per word. The neural network as 15 inputs (MFCCs) and 5 outputs.

No. of Hidden Layer Neurons = 100\*5/ 5\*(15+5) = 5

Applying the Oja rule of thumb results in 5 neurons in the hidden layer. Tests show that this amount was ideal for recognizing five words. Recognizing more words require more hidden layer units.

The output layer consists of **linear activation function elements**. We used a arbitrary assumption that the amount of output neurons is equal to the amount of words we want to recognize.

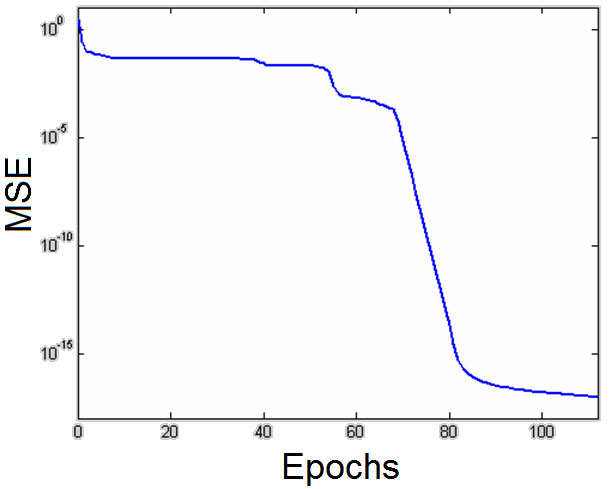
The design of this particular network is show on Fig.11. In this example, the input layer has 20 inputs (MFCCs) and the minimum and maximum values are contained in the InputLayer matrix. The hidden layer contains 7 Hyperbolic tangent sigmoid transfer function neurons. The output also has 7 linear neurons. This network is designed to recognize seven different words.

Once the network is created, it can be trained for a specific problem by presenting training inputs and their corresponding targets (supervised training).

A set of 100 samples of each word is used as training data. The network is trained in batch mode which means that the weights and biases of the network are updated only after the entire training set has been applied to the network.

The gradients calculated at each training example are added together to determine the change in the weights and biases.

In most cases, 100 up to 200 iterations (i.e. epochs) are enough to train the network sufficiently. In the training phase the network error reaches almost zero as can be seen on figure below.

[[2]](#footnote-2)

The trained network was simulated with inputs that were not in the training set. We observed that the trained network performs very well. It is possible to recognize more than ten words. The number of words that have to be recognized increases the number of hidden layer neurons. The amount of neurons needed is almost equal to the amount of words to recognize.

Increasing the number of hidden layer units causes the training time to grow sensitively. The performance of the network is mainly dependent on the quality of the signal preprocessing. The NN doesn't manage to work properly on input data coming from the spectrogram, but performs very well with MFCCs as input having more than 90% successful classification rate.

# **Closure**

Thank you for reading.

This research is an example to huge dedication to the topic. I have tried not to exceed some level of sophistication because this is not a thesis and that is an introduction to the topic. Most of the figures and topics have been carefully researched and chosen and every word here has been typed one by one without copying in Word 2016 and all compiled in one. Then, following a spell checking process, a good research report has come out.

***Tayfun Adakoğlu***

1. In probability theory, a **Markov model** is a [stochastic model](https://en.wikipedia.org/wiki/Stochastic_model) used to [model](https://en.wikipedia.org/wiki/Mathematical_model) randomly changing systems where it is assumed that future states depend only on the current state and not on the events that occurred before it. [↑](#footnote-ref-1)
2. In statistics, the **mean squared error** (**MSE**) or **mean squared deviation** (**MSD**) of an estimator measures the average of the squares of the errors  that is, the difference between the estimator and what is estimated. [↑](#footnote-ref-2)