11-755— Spring 2021
Large Scale Multimedia Processing

Lecture 2/6

Multimedia processing

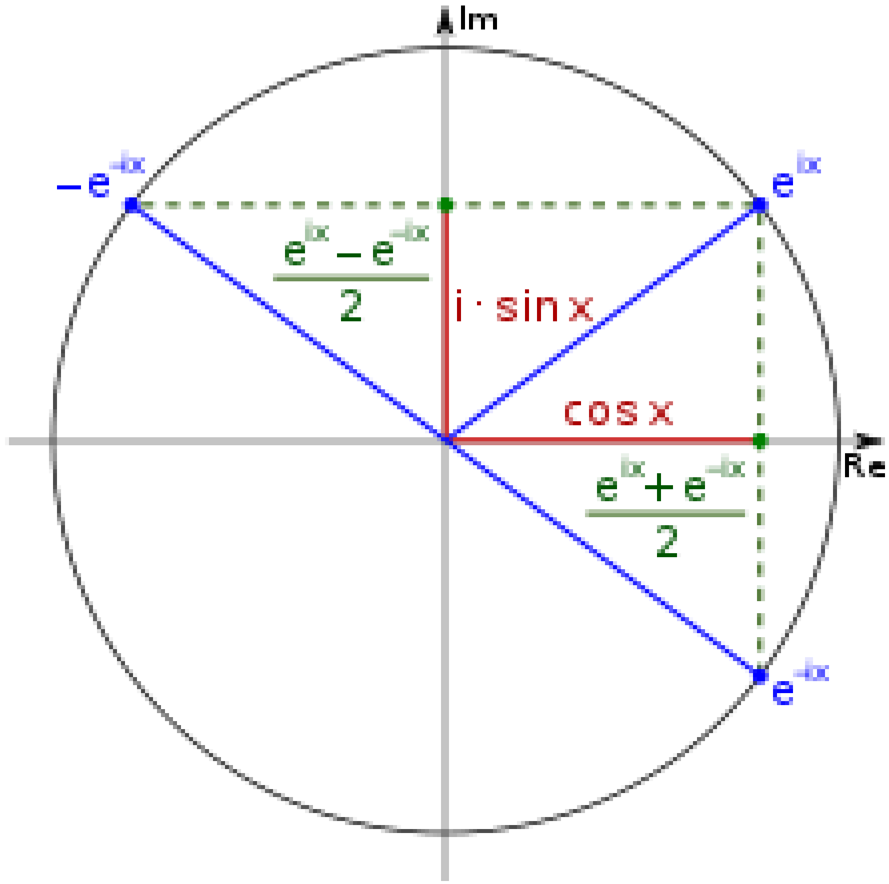
Rita Singh

Carnegie Mellon University

In this lecture

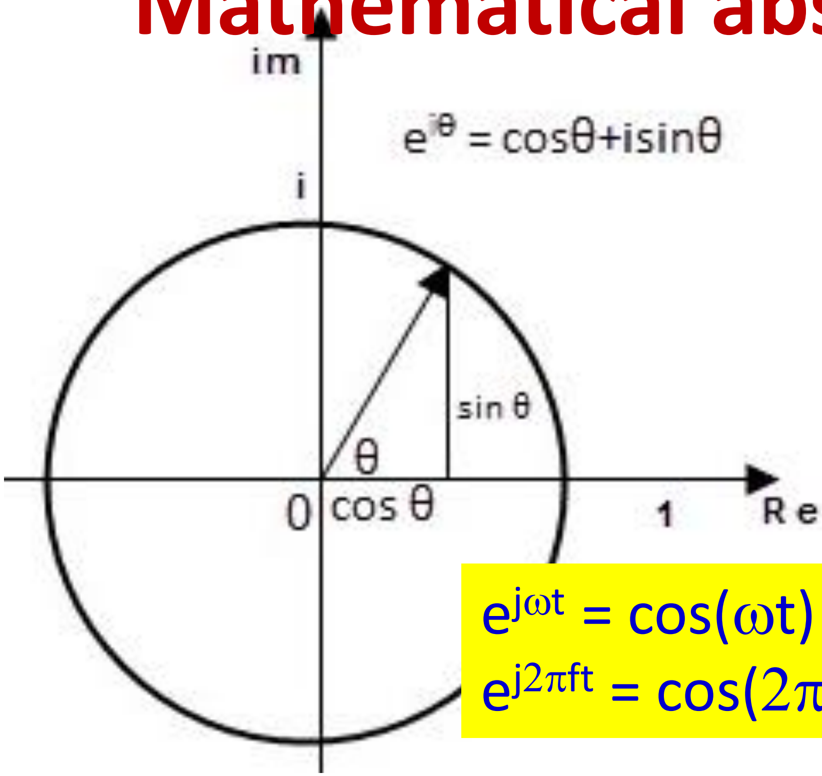
- Digital multimedia: Recording and devices
  – Audio
  – Images
  – Video
  – Text
- Digital multimedia: Processing
  – Audio processing
  – Two generic processing techniques

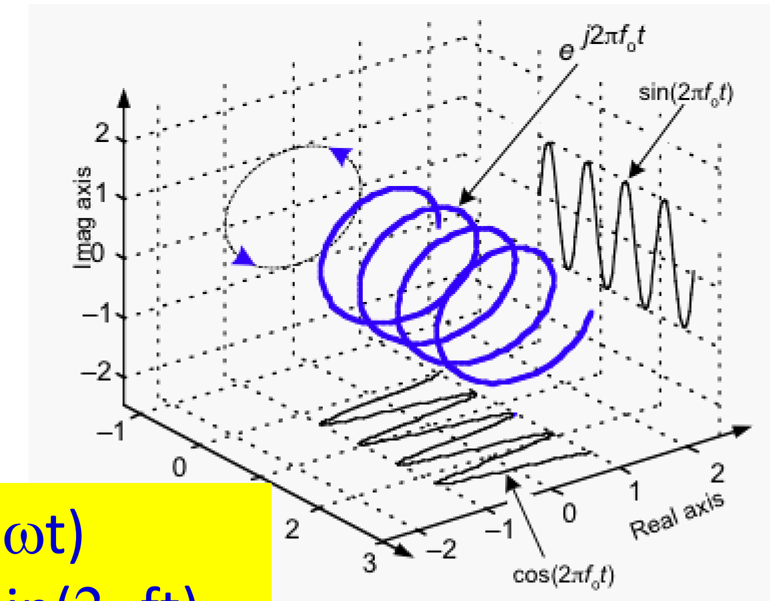
Mathematical abstraction of sinusoids



- A phasor abstraction for sinusoids

Mathematical abstraction of sinusoids



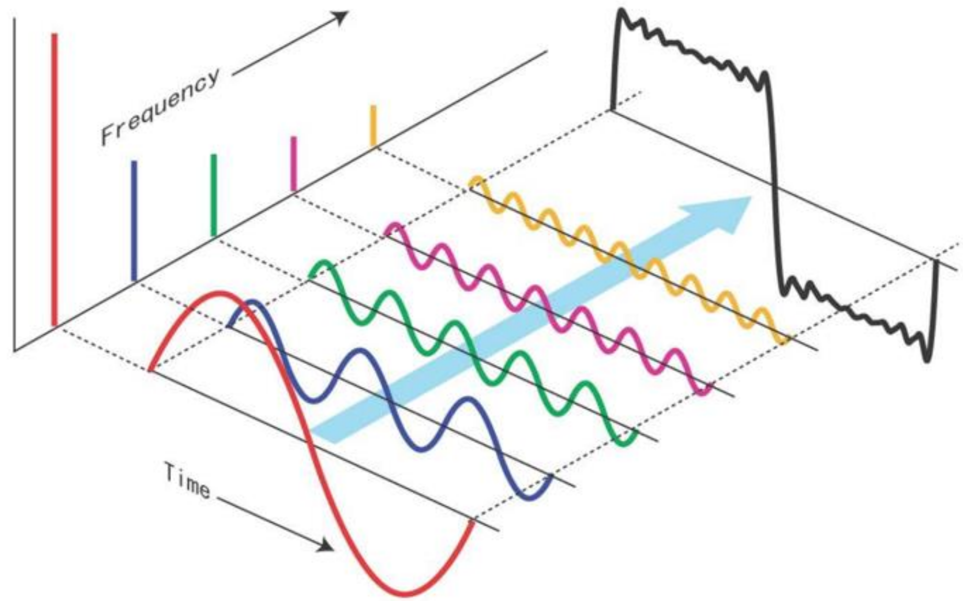


$e^{j\omega t} = \cos(\omega t) + j \sin(\omega t)$

$e^{j2\pi ft} = \cos(2\pi ft) + j \sin(2\pi ft)$

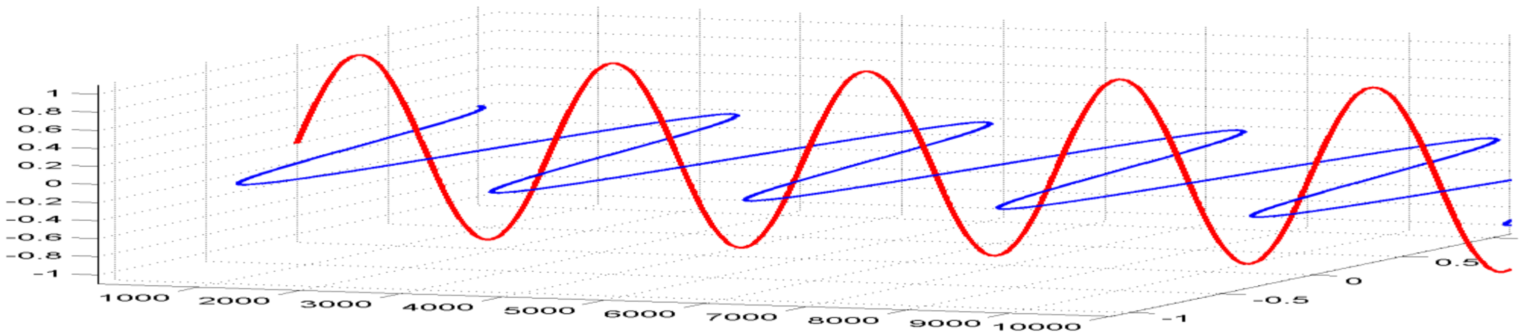
- f: cycles/sec
- Per cycle: 2π radians
- phase = (cycles/sec) x (angles/cycle) x time = f 2π t (cylces per second as unit)
  - Think of it as total angle traversed over time
- Angle/sec is denoted as $\omega = f\,2\pi$ (also called angular frequency)
- Phase - ω t (angles per second as unit)

Signal decomposition



- Any periodic signal can be represented as a sum of sinusoids

The complex exponential



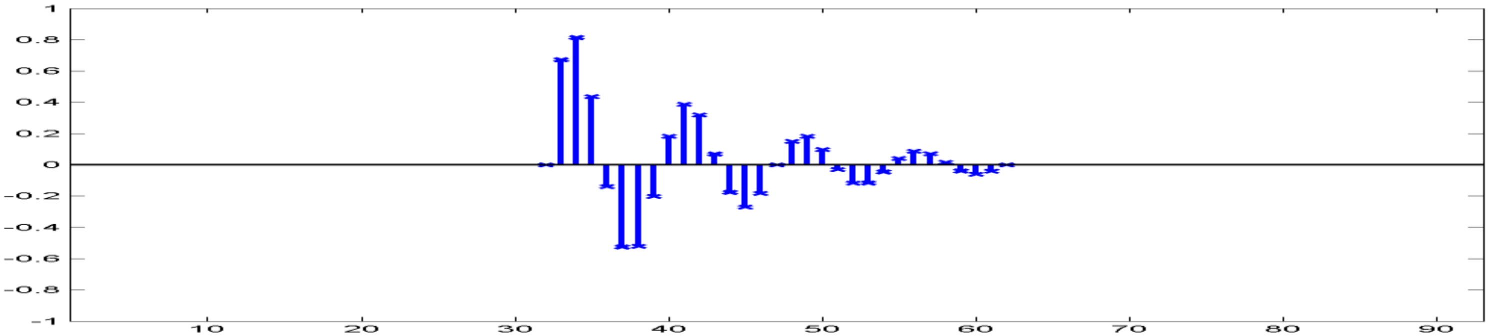
- The complex exponential is a complex sum of two sinusoids
  - $e^{j\theta} = \cos\theta + j \sin\theta$
- The real part is a cosine function
- The imaginary part is a sine function
- A complex exponential time series is a complex sum of two time series
  - $e^{j\omega t} = \cos(\omega t) + j \sin(\omega t)$
- Two complex exponentials of different frequencies are “orthogonal” to each other. i.e.

$$\int_{-\infty}^{\infty} e^{j\alpha t} e^{j\beta t} dt = 0 \qquad \text{if } \alpha \neq \beta$$

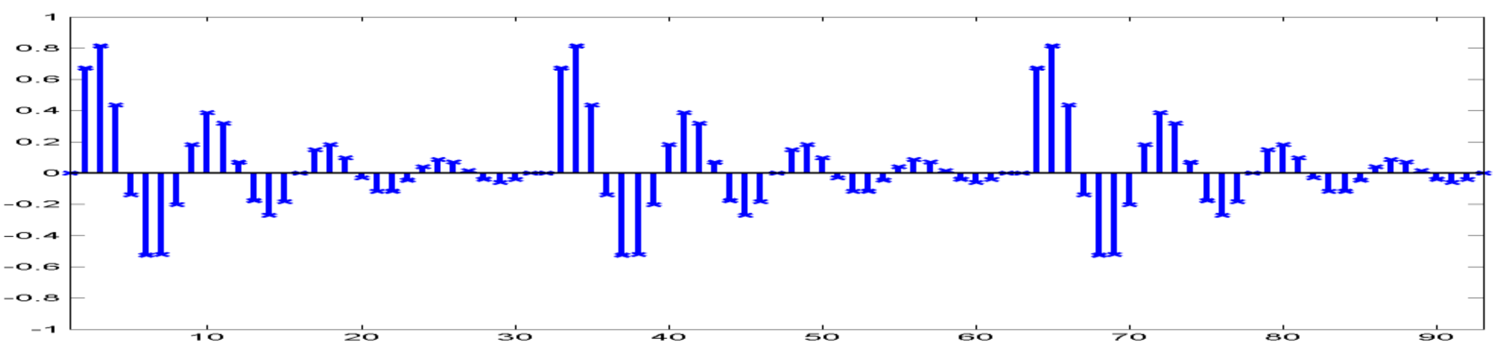
The discrete Fourier transform

- The discrete Fourier transform decomposes the signal into the sum of a finite number of complex exponentials
  - As many exponentials as there are samples in the signal being analyzed
- An aperiodic signal *cannot* be decomposed into a sum of a finite number of complex exponentials
  - Or into a sum of any countable set of periodic signals
- The discrete Fourier transform actually assumes that the signal being analyzed is exactly one period of an infinitely long signal
  - In reality, it computes the Fourier spectrum of the infinitely long periodic signal, of which the analyzed data are one period

The discrete Fourier transform



- The discrete Fourier transform of the above signal actually computes the Fourier spectrum of the periodic signal shown below
  - Which extends from –infinity to +infinity
  - The period of this signal is 31 samples in this example



The discrete Fourier transform

- The k^th point of a Fourier transform is computed as:

$$X[k] = \sum_{n=0}^{M-1} x[n] e^{-\frac{j2\pi kn}{M}}$$

  - $x[n]$ is the n^th point in the analyzed data sequence
  - $X[k]$ is the value of the k^th point in its Fourier spectrum
  - M is the total number of points in the sequence

- Note that the (M+k)^th Fourier coefficient is identical to the k^th Fourier coefficient

$$X[M+k] = \sum_{n=0}^{M-1} x[n] e^{-\frac{j2\pi(M+k)n}{M}} = \sum_{n=0}^{M-1} x[n] e^{-\frac{j2\pi Mn}{M}} e^{-\frac{j2\pi kn}{M}}$$

$$= \sum_{n=0}^{M-1} x[n] e^{-j2\pi n} e^{-\frac{j2\pi kn}{M}} = \sum_{n=0}^{M-1} x[n] e^{-\frac{j2\pi kn}{M}} = X[k]$$

The discrete Fourier transform

- Discrete Fourier transform coefficients are generally complex
  - $e^{j\theta}$ has a real part $\cos\theta$ and an imaginary part $\sin\theta$

$$e^{j\theta} = \cos\theta + j\sin\theta$$

  - As a result, every X[k] has the form

$$X[k] = X_{real}[k] + jX_{imaginary}[k]$$

- A magnitude spectrum represents only the magnitude of the Fourier coefficients

$$X_{magnitude}[k] = \mathrm{sqrt}(X_{real}[k]^2 + X_{imag}[k]^2)$$

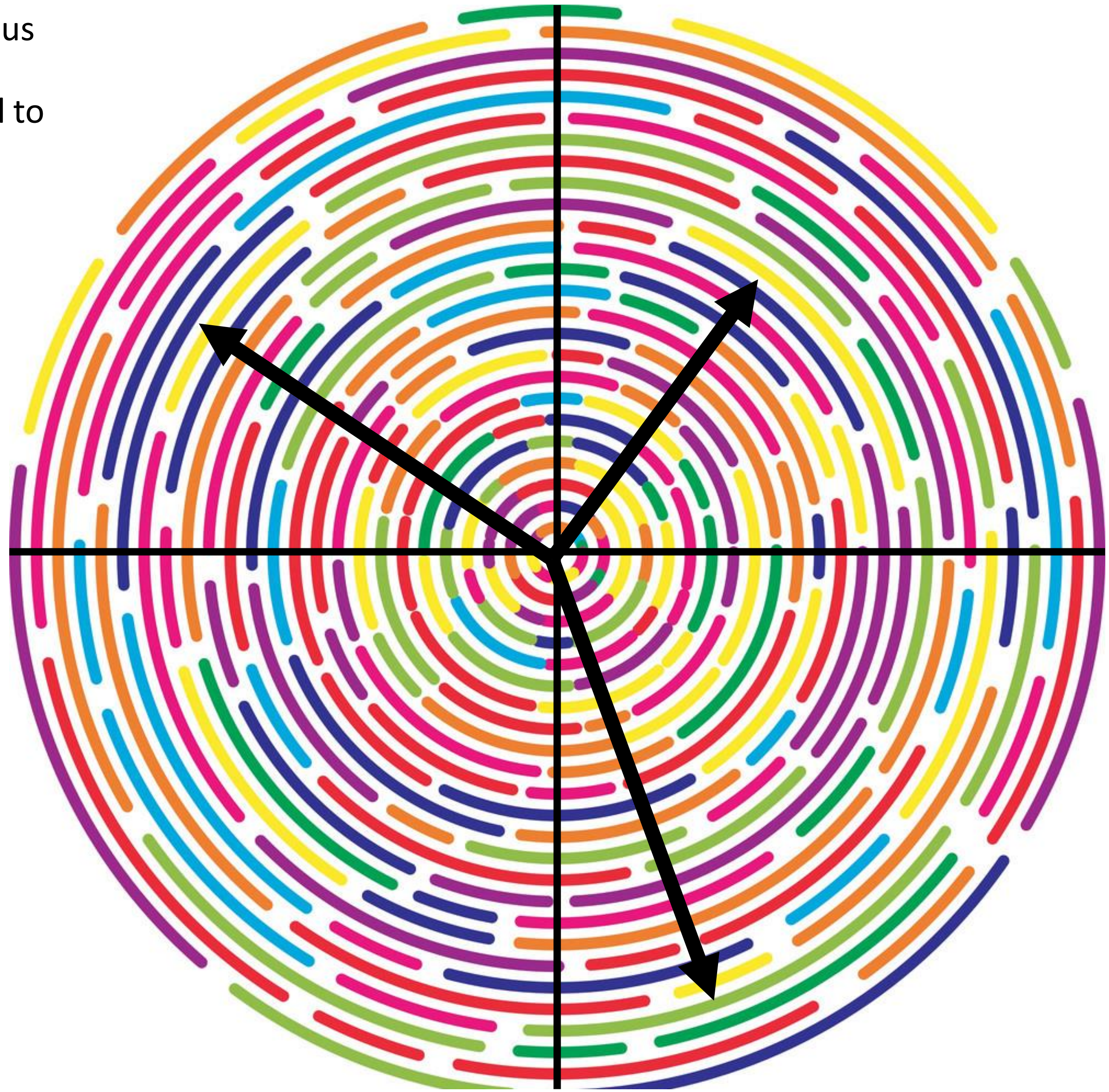
- A power spectrum is the square of the magnitude spectrum

$$X_{power}[k] = X_{real}[k]^2 + X_{imag}[k]^2$$

- For speech recognition and other audio analyses, we usually use the magnitude or power spectra

The phasor, magnitude and power

- Magnitude is the radius
  - always positive
- Power is proportional to energy
  - Squared Fourier coefficient
  - always positive



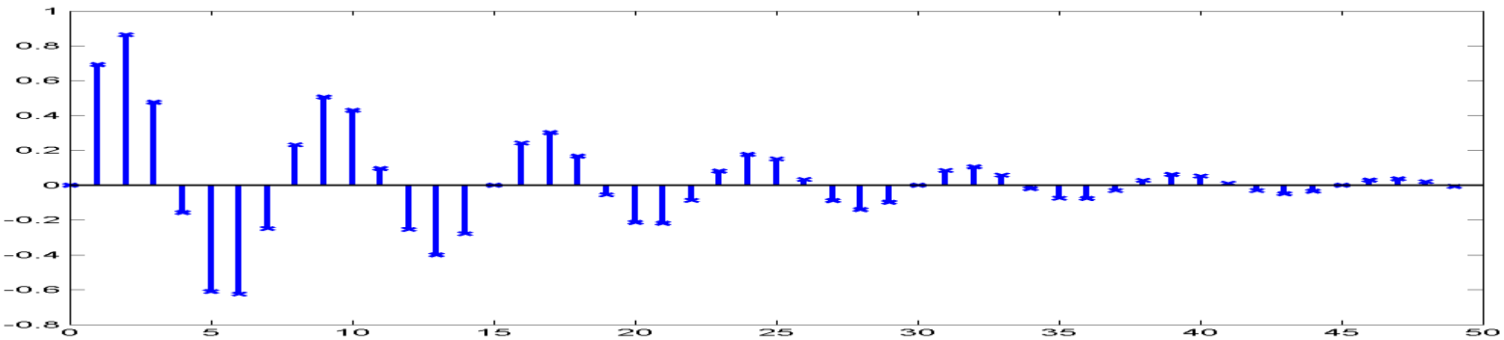
[if curious: https://www.sciencedirect.com/topics/engineering/magnitude-spectrum](https://www.sciencedirect.com/topics/engineering/magnitude-spectrum)

The discrete Fourier transform

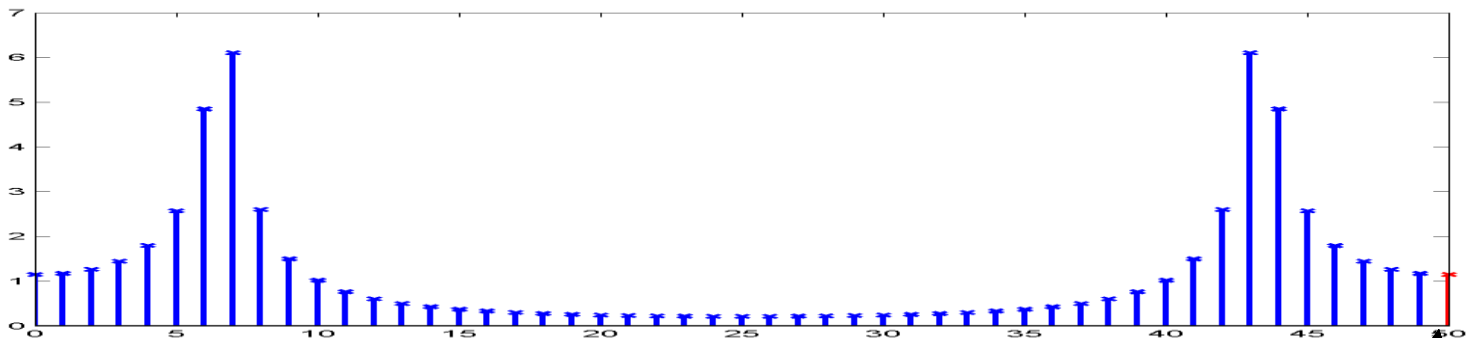
- A discrete Fourier transform of an M-point sequence will only compute M unique frequency components
  - i.e. the DFT of an M point sequence will have M points
  - The M-point DFT represents frequencies in the continuous-time signal that was digitized to obtain the digital signal
- The 0^{th} point in the DFT represents 0Hz, or the DC component of the signal
- The $(M-1)^{th}$ point in the DFT represents (M-1)/M times the sampling frequency
- All DFT points are uniformly spaced on the frequency axis between 0 and the sampling frequency

The discrete Fourier transform

- A 50 point segment of a decaying sine wave sampled at 8000 Hz



The corresponding 50 point magnitude DFT. The 51st point (shown in red) is identical to the 1st point.



Sample 0 = 0 Hz

Sample 50 is the 51st point
It is identical to Sample 0

Sample 50 = 8000Hz

The discrete Fourier transform

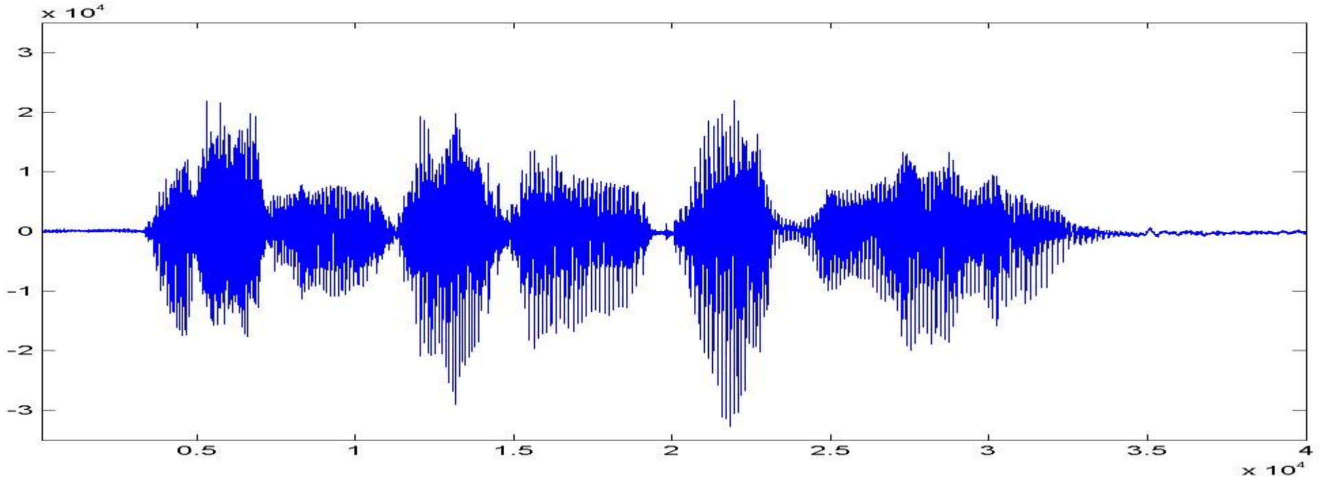
- The *Fast Fourier Transform* (FFT) is simply a fast algorithm to compute the DFT
  - It utilizes symmetry in the DFT computation to reduce the total number of arithmetic operations greatly

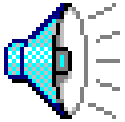
- The time domain signal can be recovered from its DFT as:

$$x[n] = \frac{1}{M} \sum_{k=0}^{M-1} X[k] e^{\frac{j2\pi kn}{M}}$$

Example: An audio signal

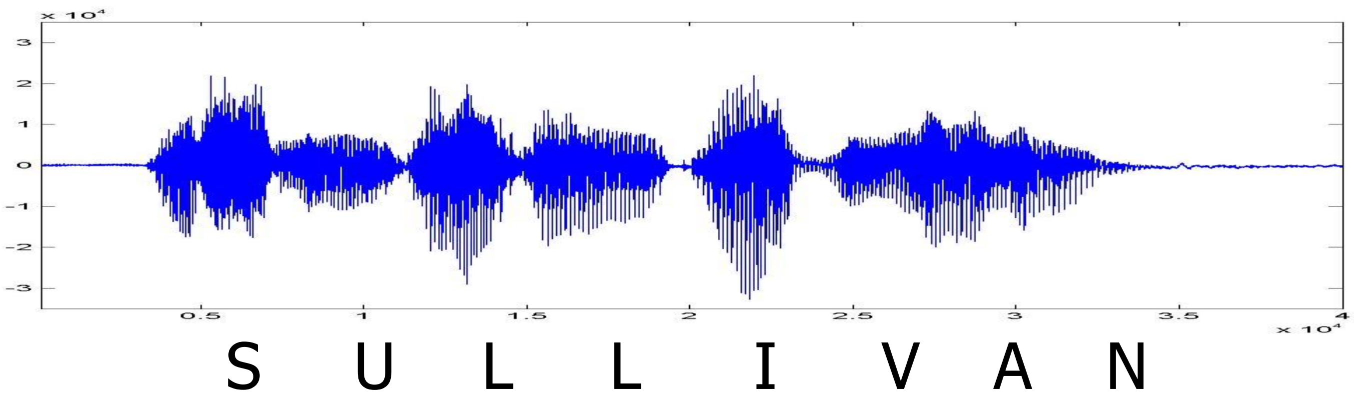
Tom Sullivan spells out his last name: S U L L I V A N





Example: A speech signal

Sound is produced as an analog signal.
It is converted to digital format by analog-to-digital (A/D) converters.



This signal has been digitized at the rate of 16000 numbers per second. ie,
Signal sampling frequency=16 kiloHertz
or
Sampling rate = 16 kiloHertz

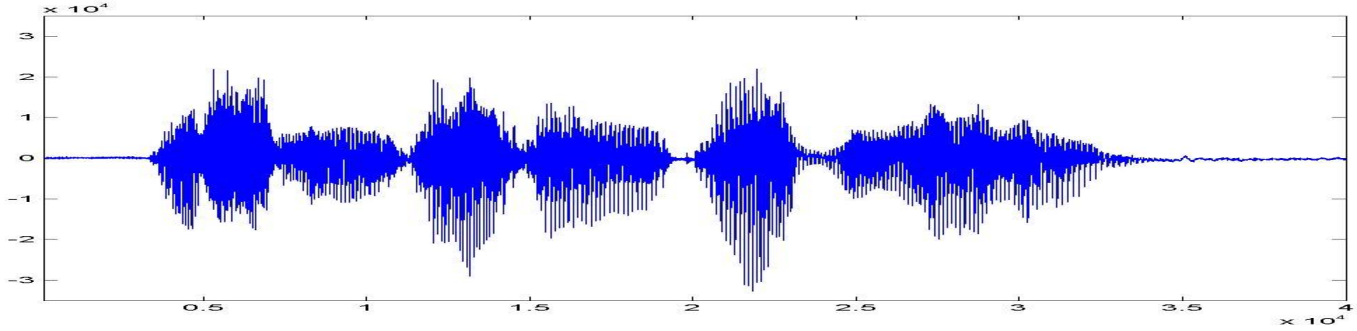
Samples are 16bit integers. Sample No. 13172:13200 (read left-to-right)

-24780	-29075	-26479	-19223	-18953	-13313	-2011	3440	7412	11209
15823	19792	17963	15337	12193	8742	7267	3721	1150	-443
-2989	-2499	-2761	-4020	-2898	-4016	-6988	-9668	-10614	

Capturing the Spectrum: The discrete Fourier transform

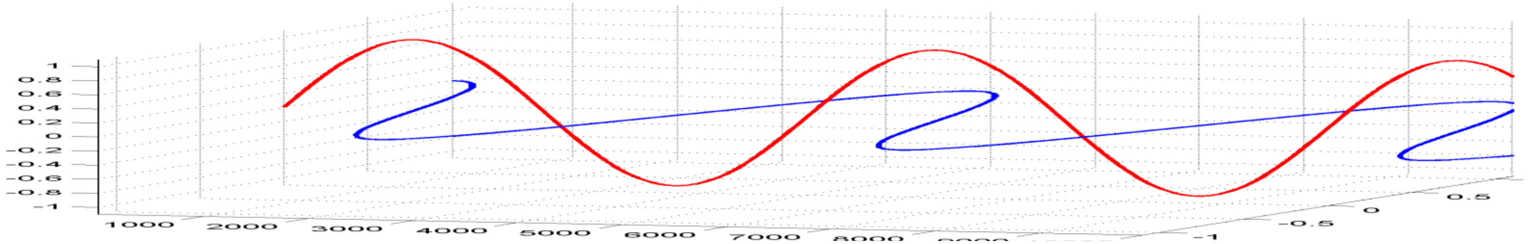
- Transform analysis: Decompose a sequence of numbers into a weighted sum of other time series
  - The component time series must be defined
  - For the Fourier Transform, these are complex exponentials

- The transform analysis determines the weights of the component time series



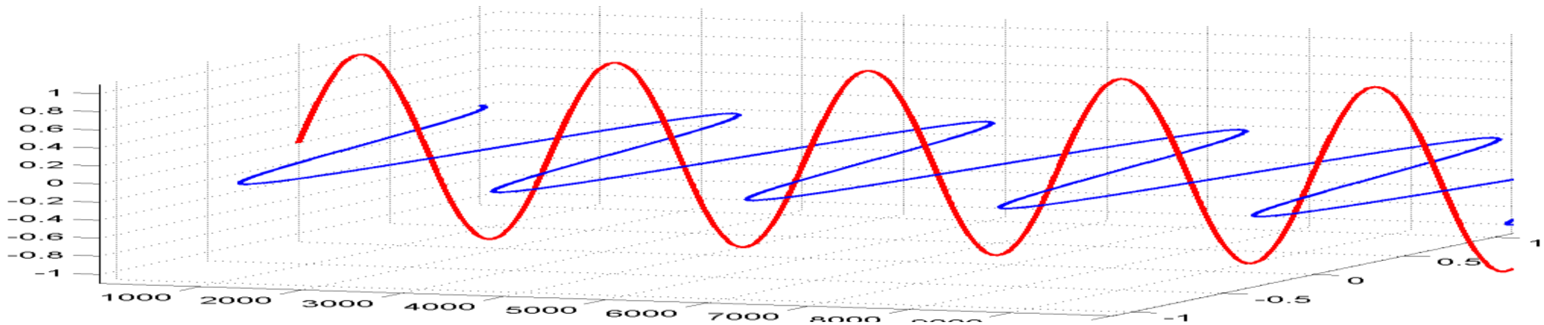
The discrete Fourier transform

A x



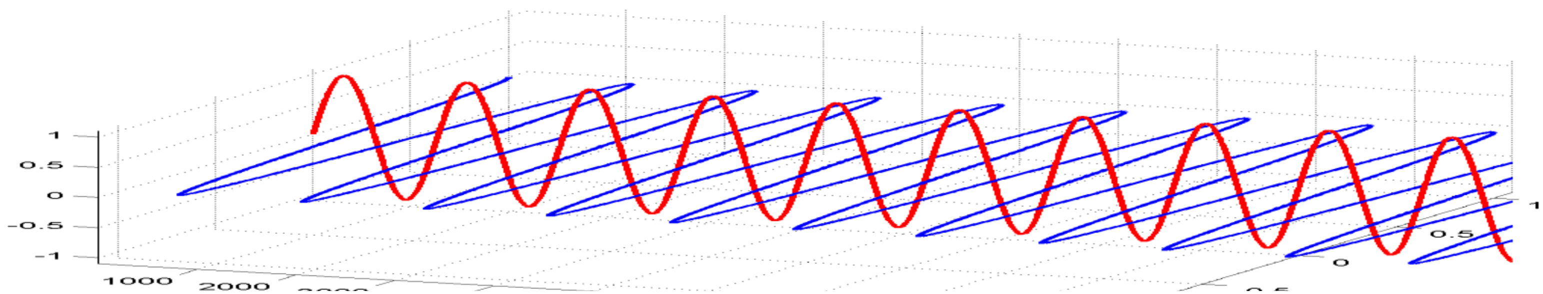
+

B x

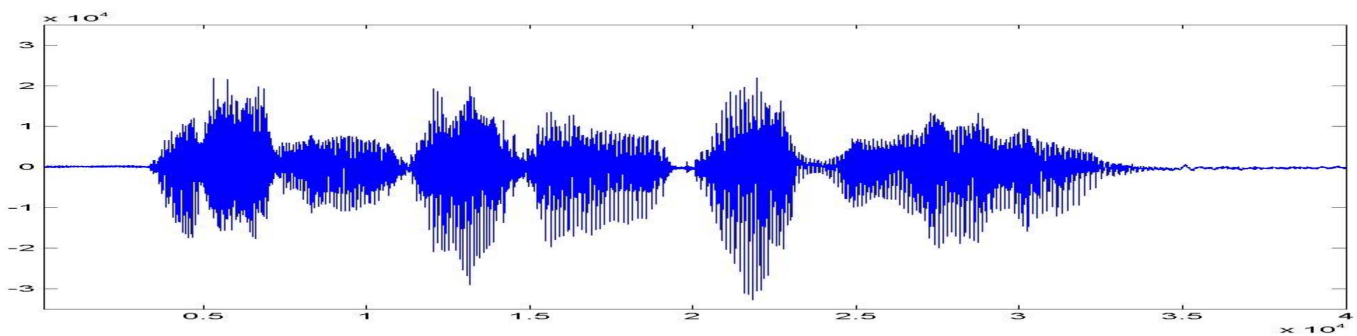


+

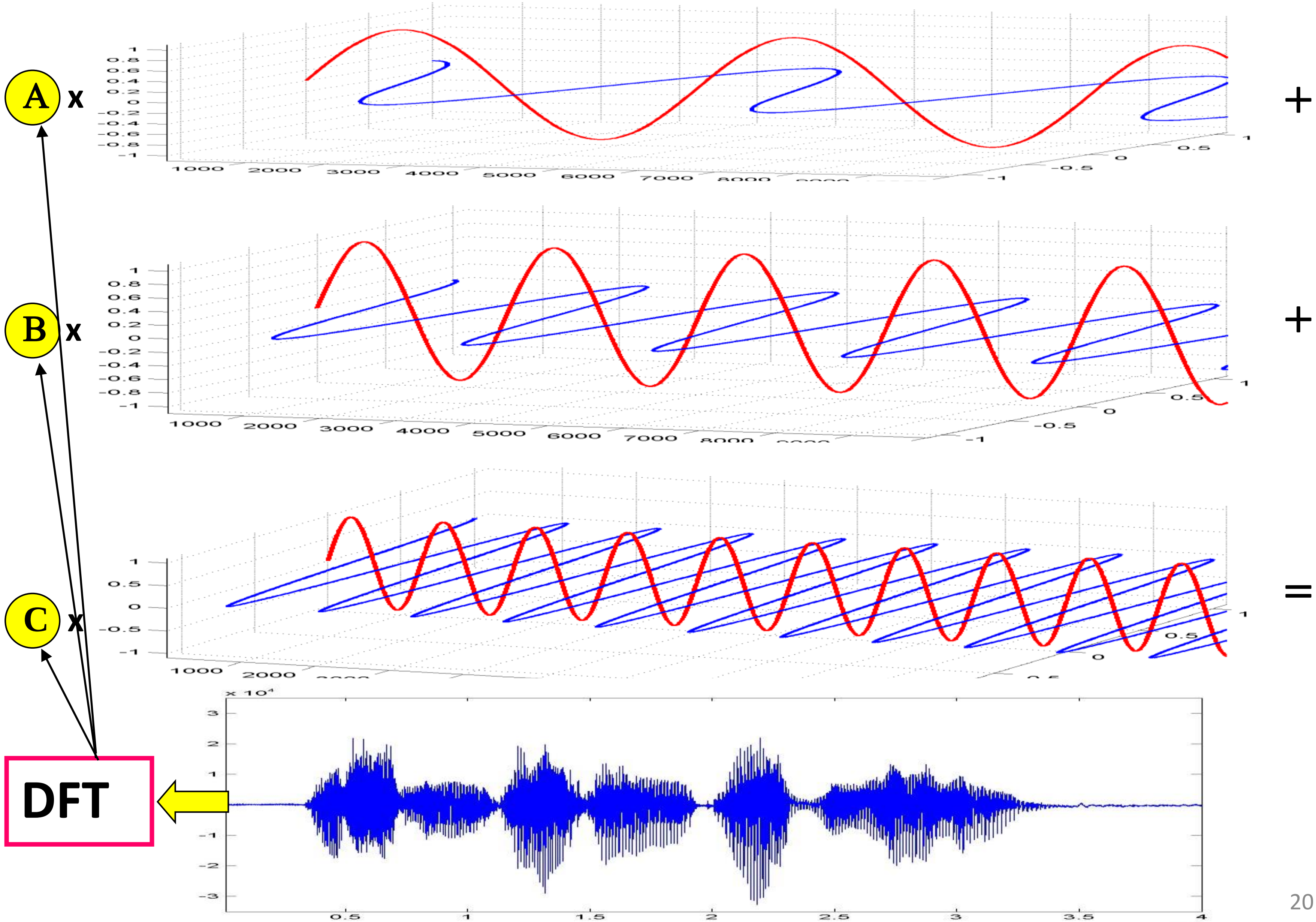
C x



=

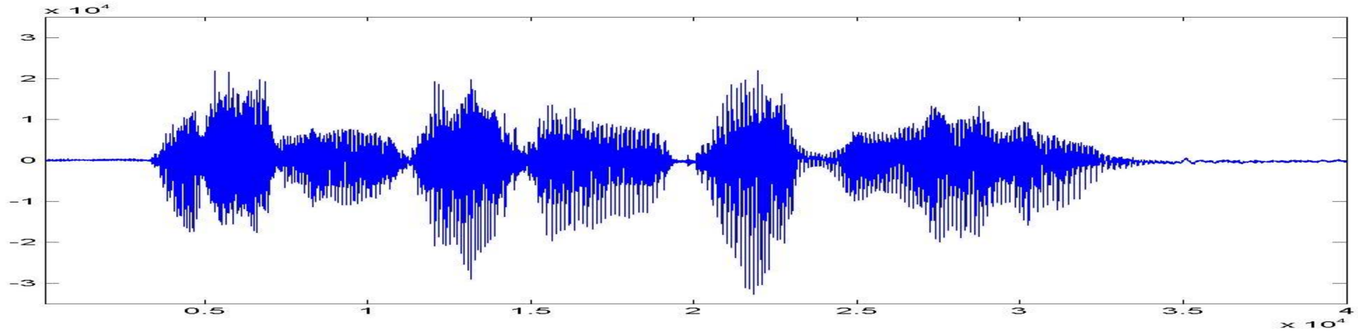


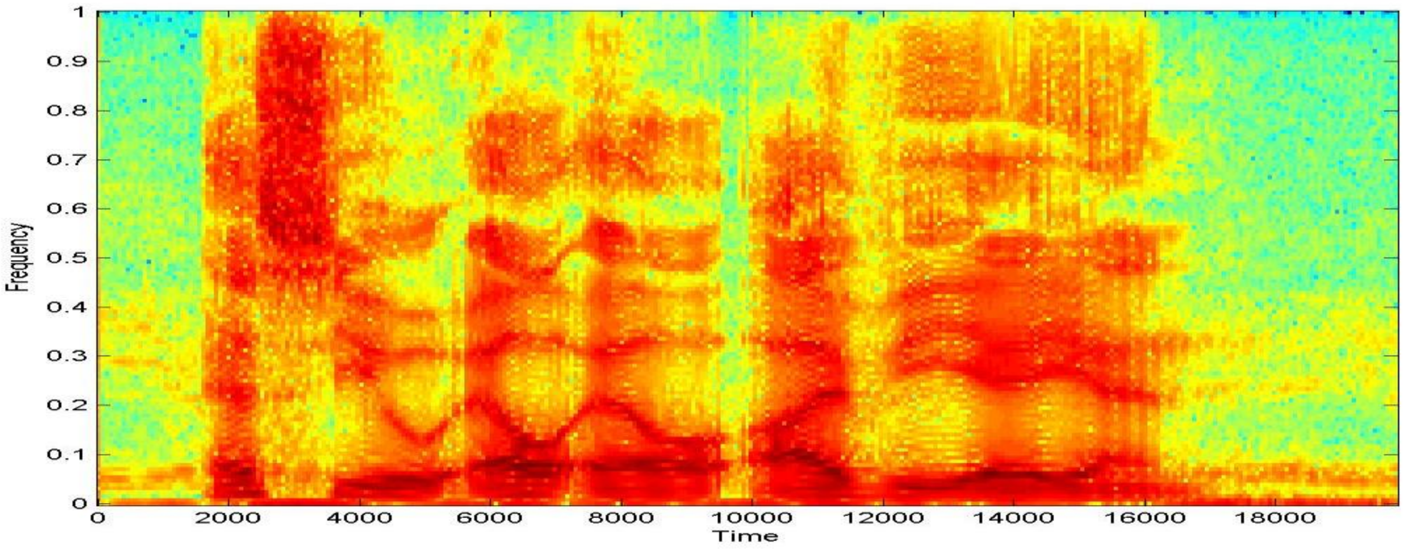
The discrete Fourier transform



A clean speech signal and it's spectrogram

Tom Sullivan spells out his last name: S U L L I V A N

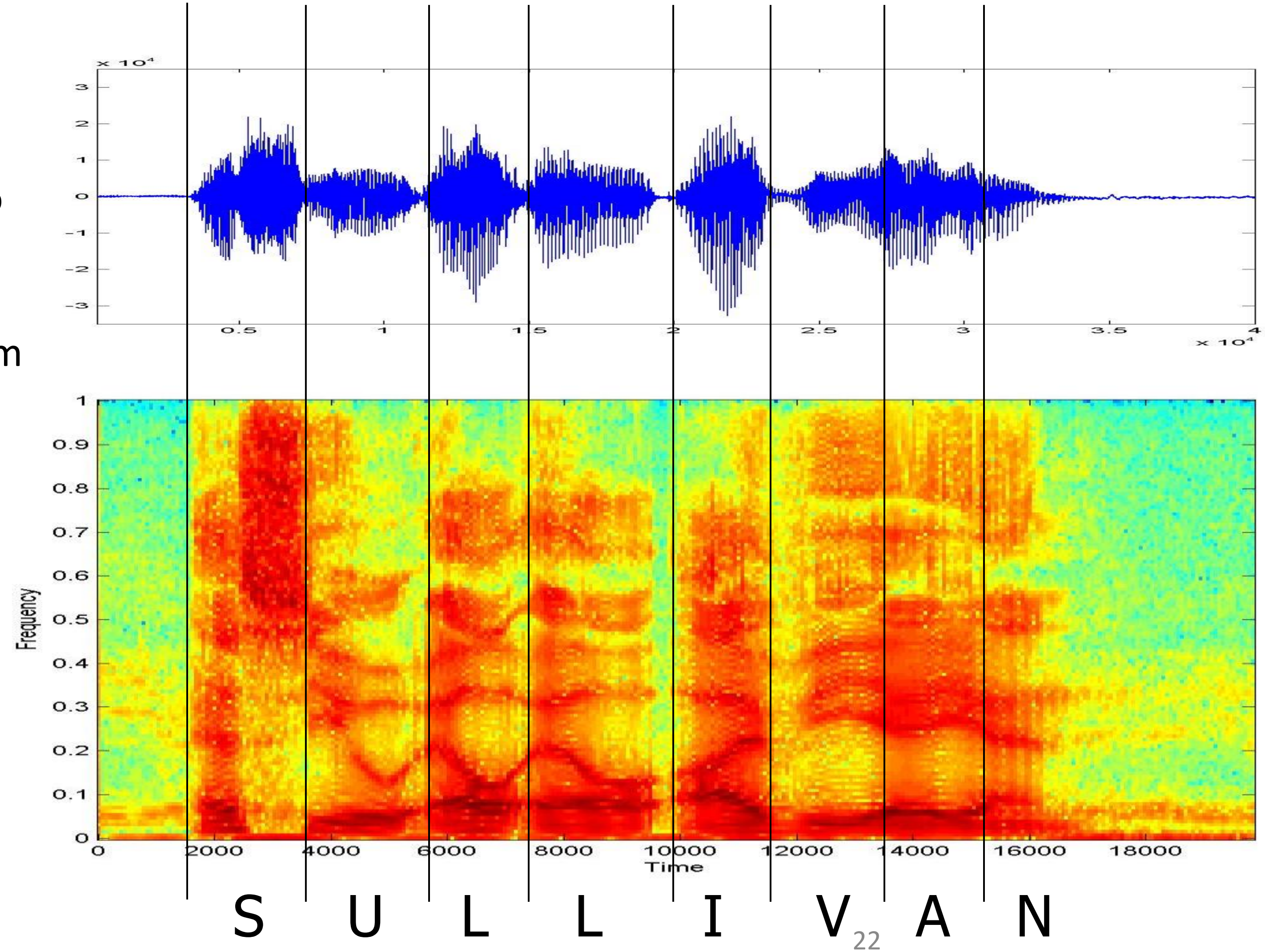


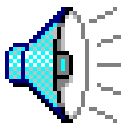


A clean speech signal and it’s spectrogram

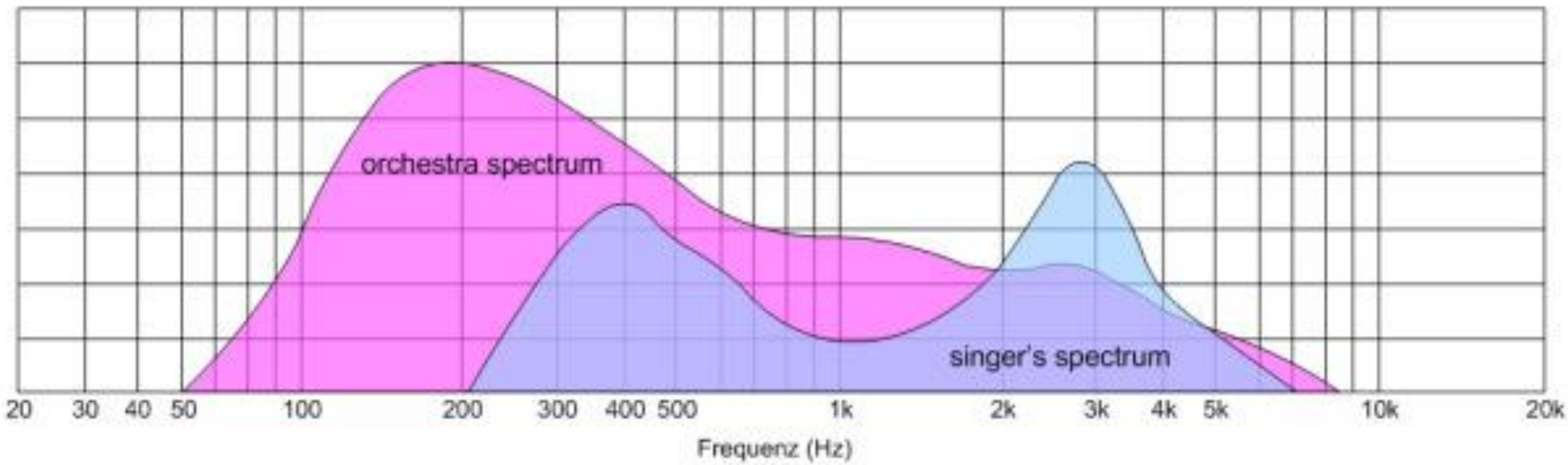
Tom Sullivan spells out his last name: S U L L I V A N

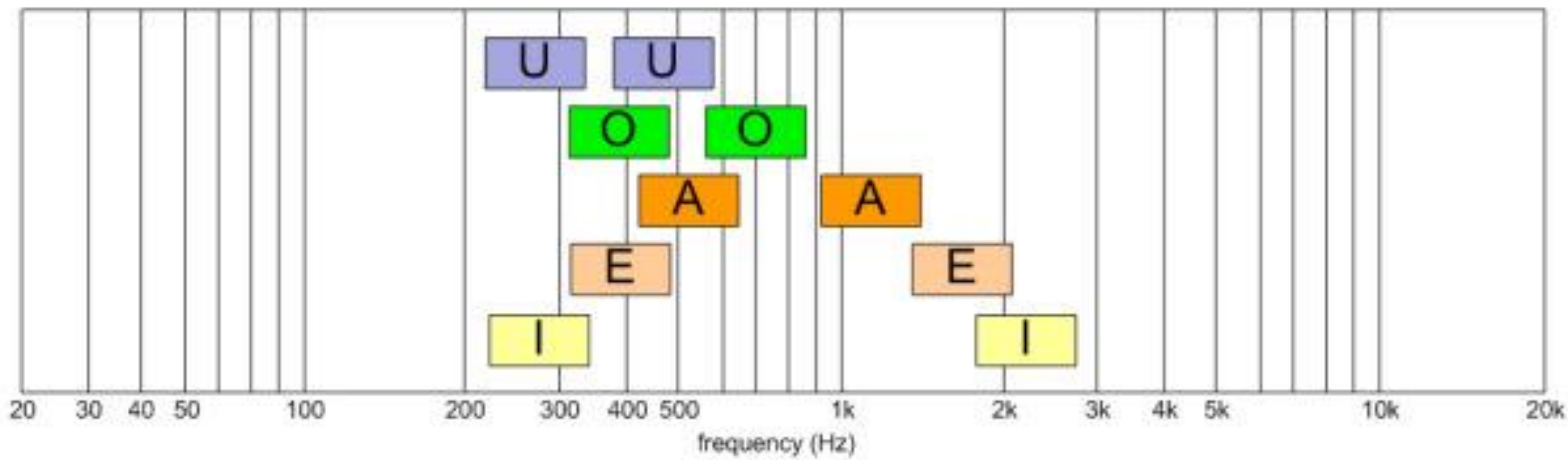
Tool: Many free tools available to compute and study spectrograms, e.g. Audacity Praat





Spectrum

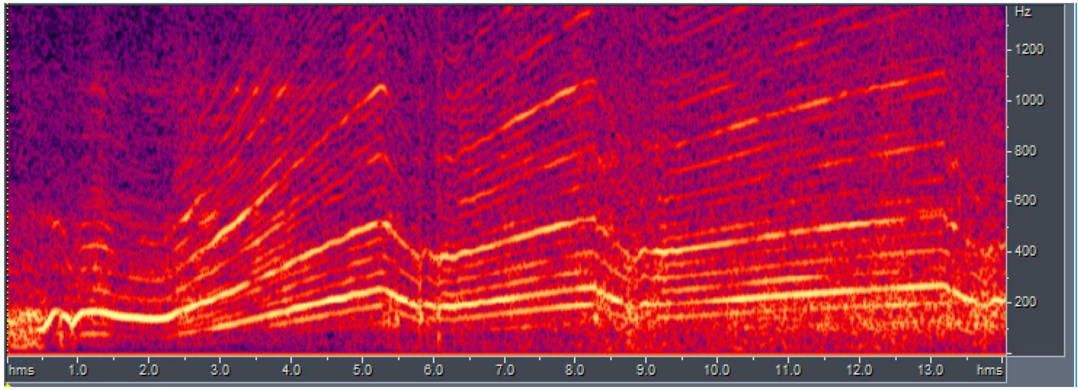


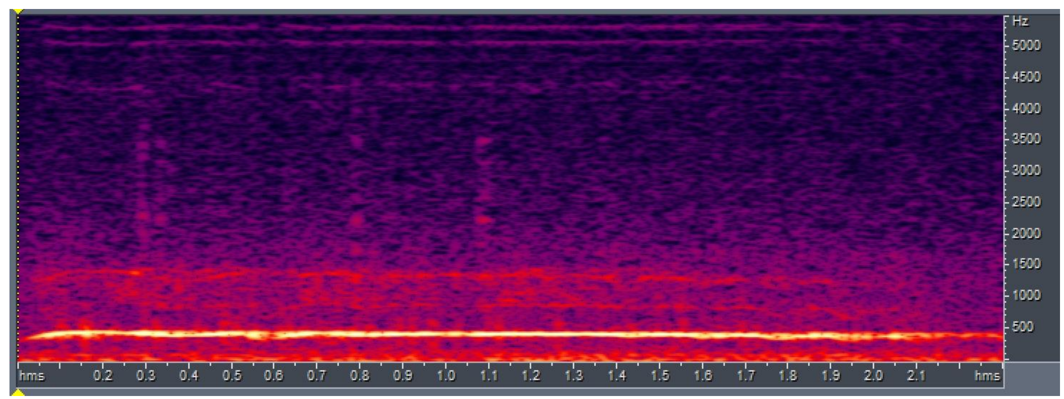


- Audio/speech spectrum

http://www.bnoack.com/index.html?http&&&
www.bnoack.com/audio/speech-level.html

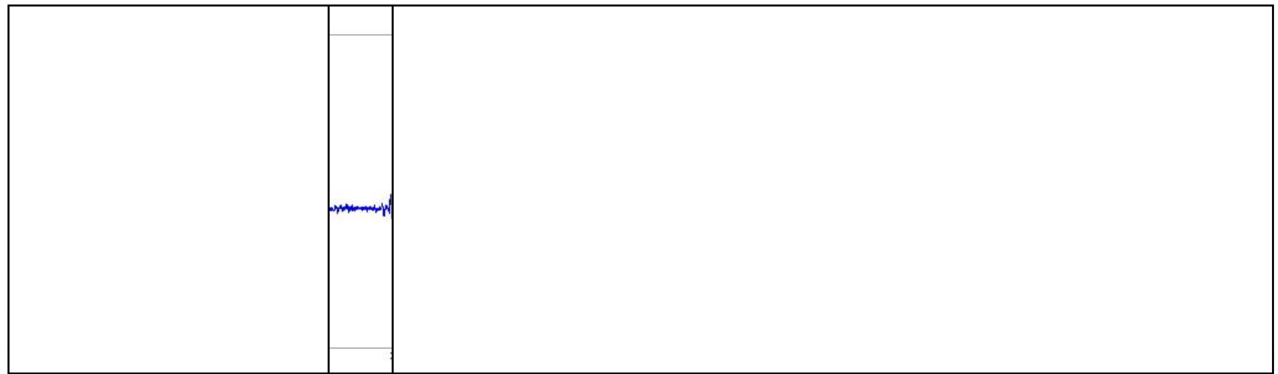
Audio (sound) signatures in a spectrogram

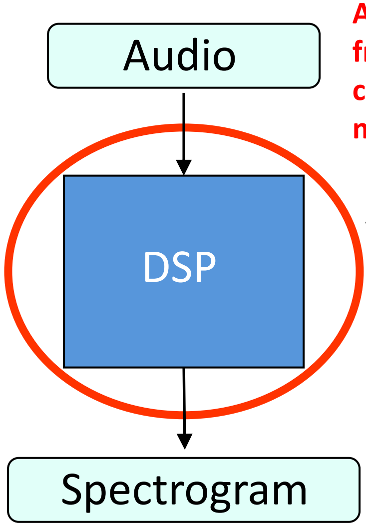




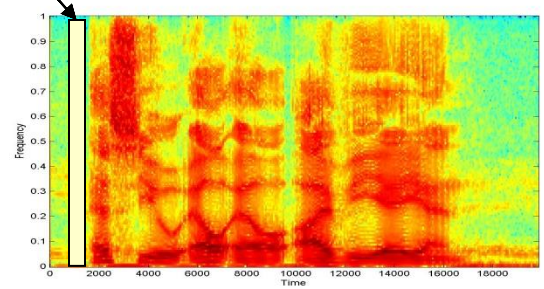
- A Ferrari going at 290kmph
- A barn owl

How is the spectrogram computed?

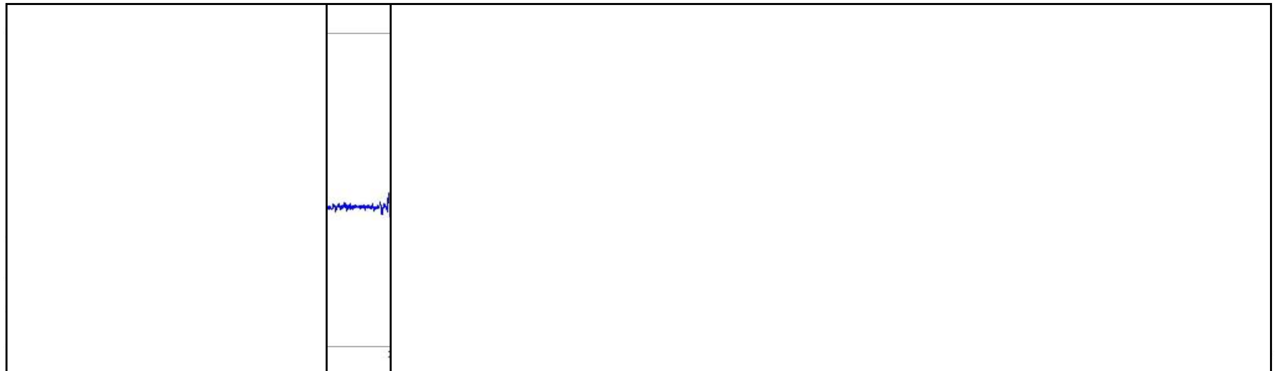


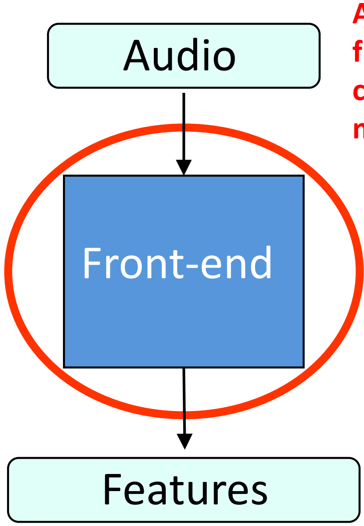


Audio is windowed into frames. Each frame is converted into a multidimensional vector

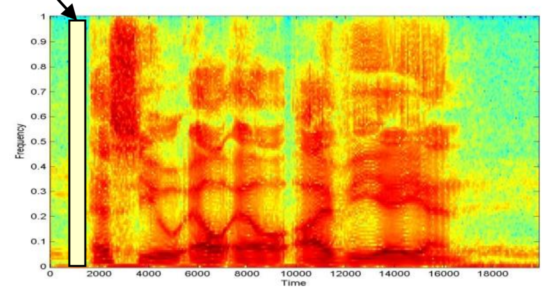


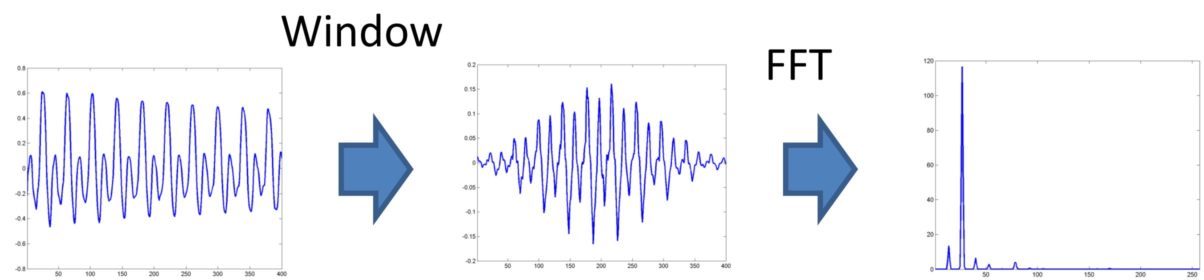
How is this *vector* computed?



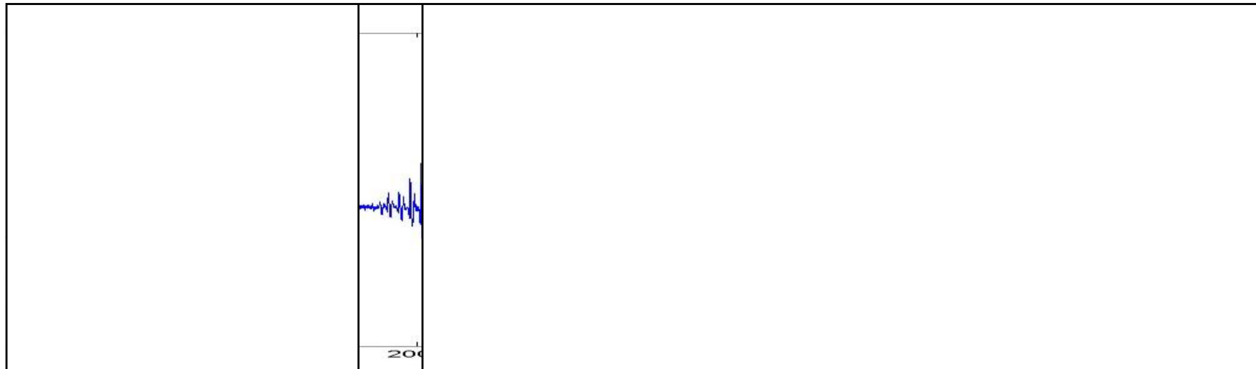


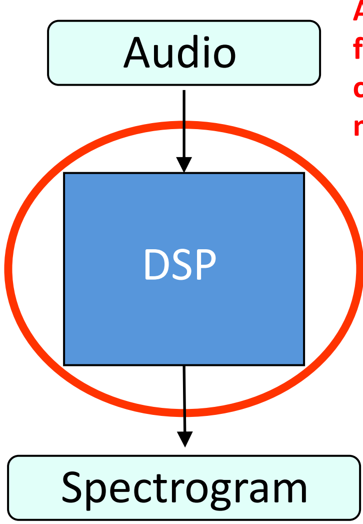
Audio is windowed into frames. Each frame is converted into a multidimensional vector



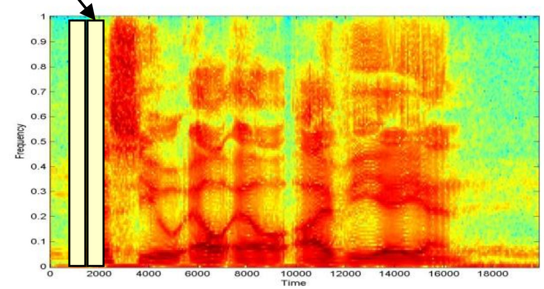


How is the spectrogram computed?

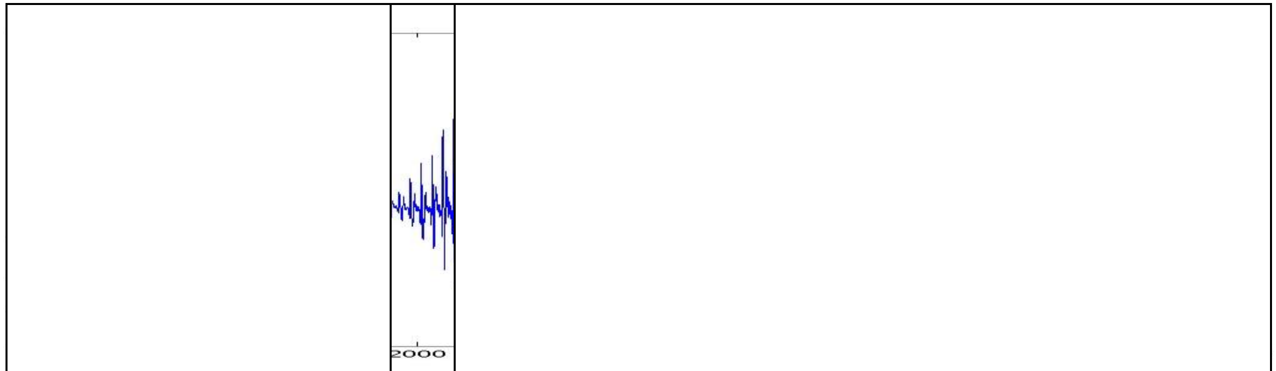


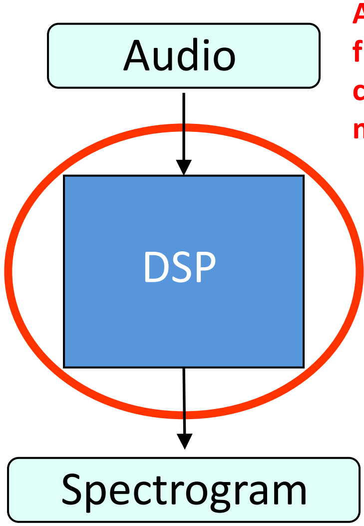


Audio is windowed into frames. Each frame is converted into a multidimensional vector

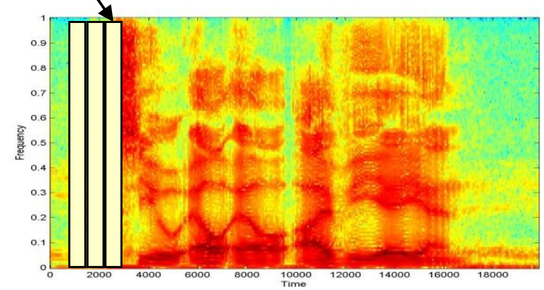


How is the spectrogram computed?

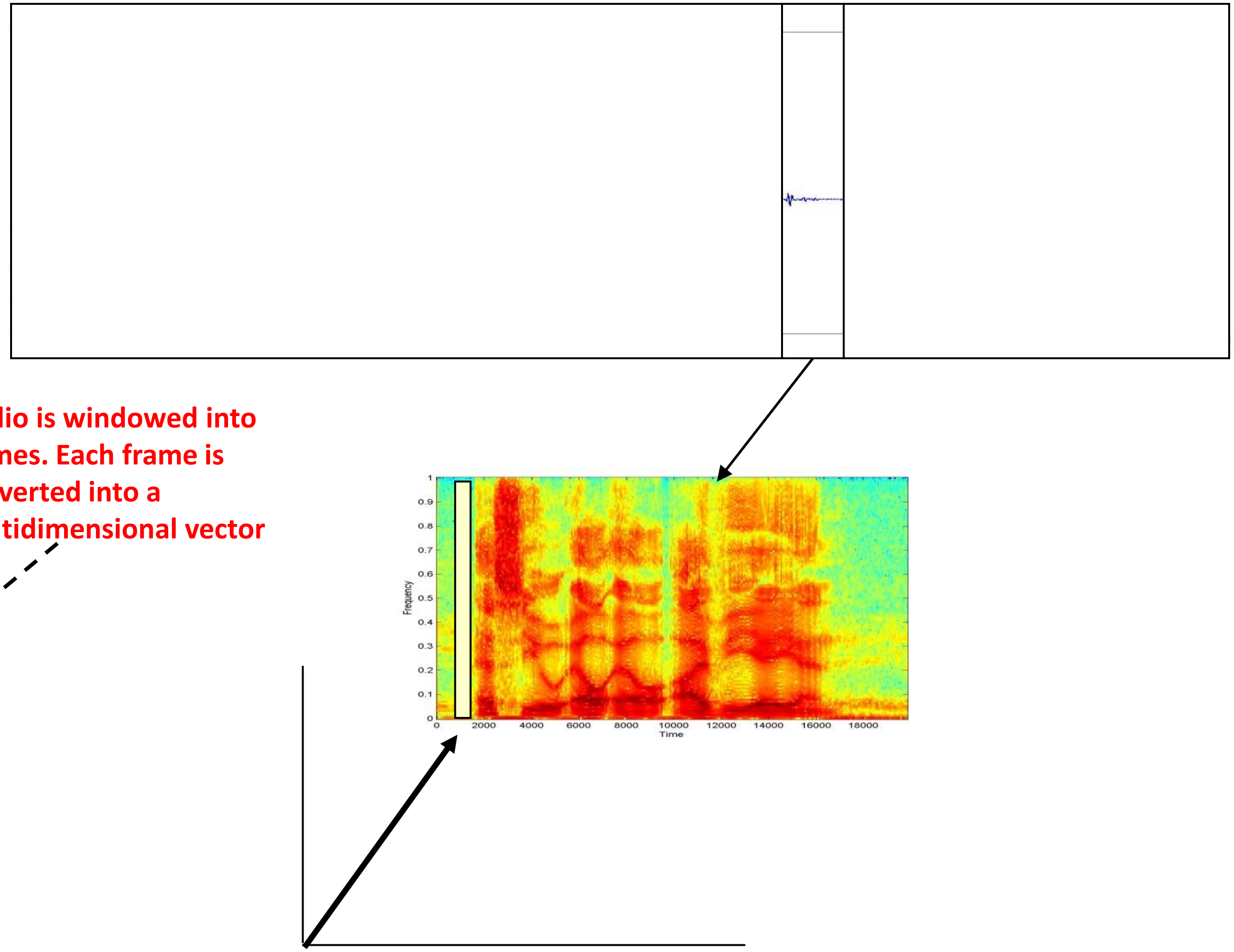


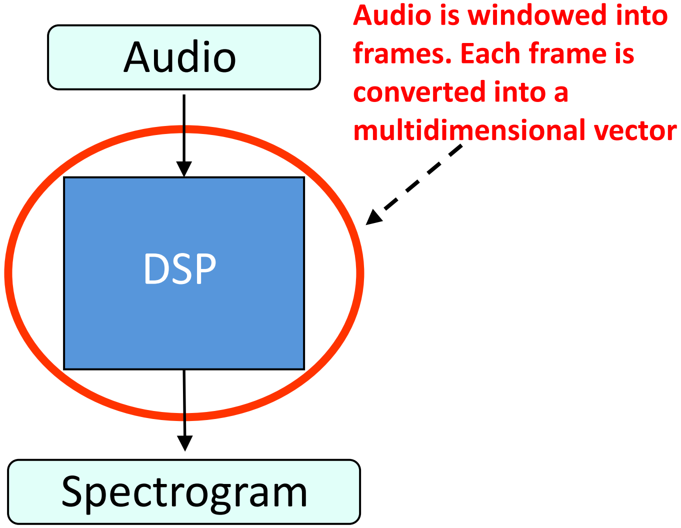


Audio is windowed into frames. Each frame is converted into a multidimensional vector



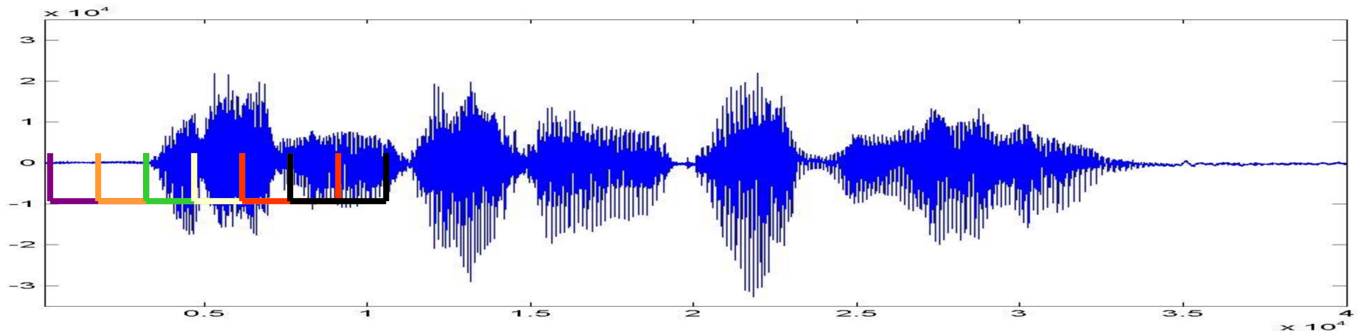
A spectogram is composed of vectors





- The "picture" is simply a visualization of the sequence of vectors
- Each high-dimensional vector represents a slice of time
- Each component of the vector represents a particular frequency at that time

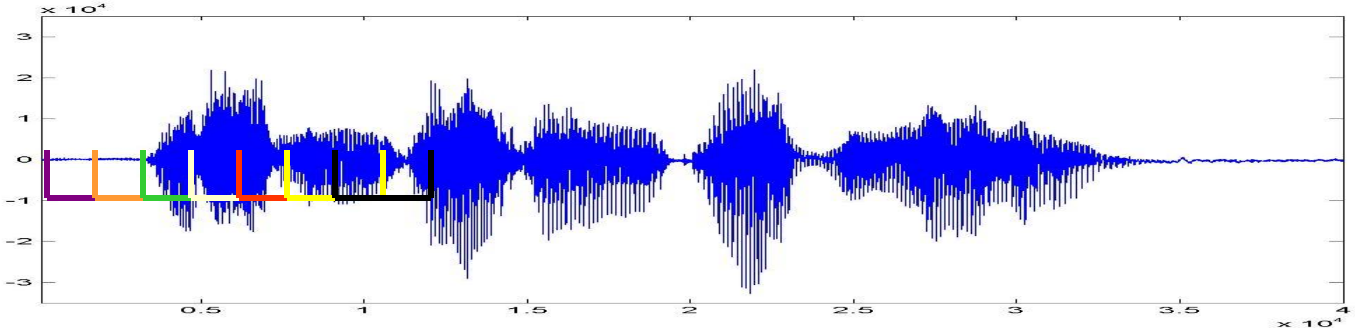
Computing Mel Cepstra



The signal is processed in segments. Segments are typically 25 ms wide.

Adjacent segments typically overlap by 15 ms.

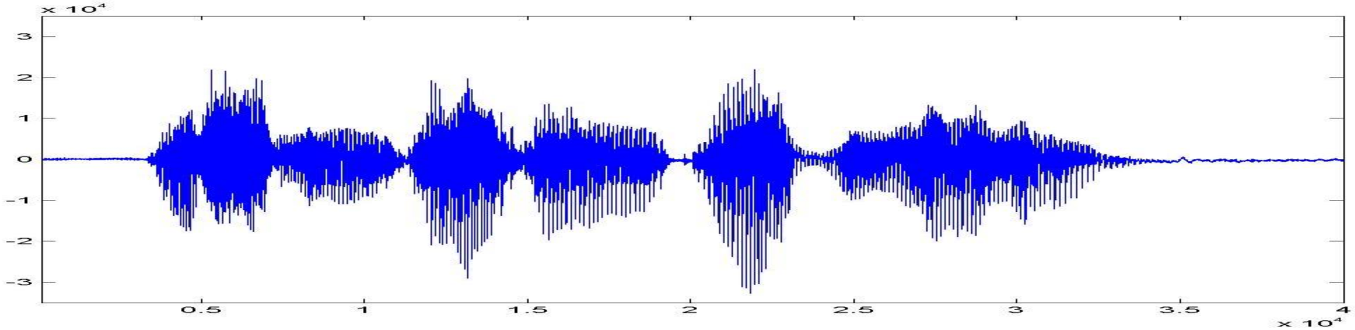
Computing Mel Cepstra

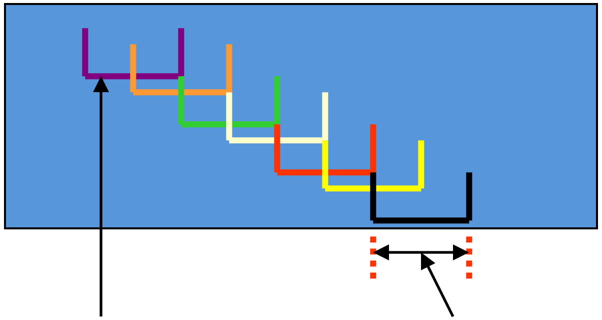


The signal is processed in segments. Segments are typically 25 ms wide.

Adjacent segments typically overlap by 15 ms.

Computing Mel Cepstra



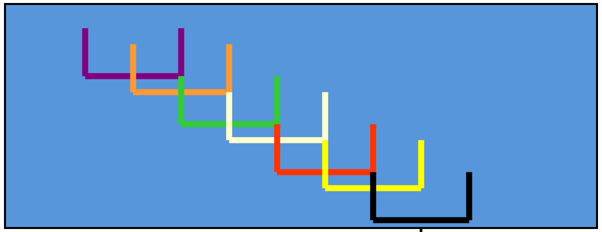


Segments shift every 10 milliseconds

Each segment is typically 20 or 25 milliseconds wide

Speech signals do not change significantly within this short time interval

Computing Mel Cepstra



Each segment is preemphasized

Preemphasized segment

The preemphasized segment is windowed

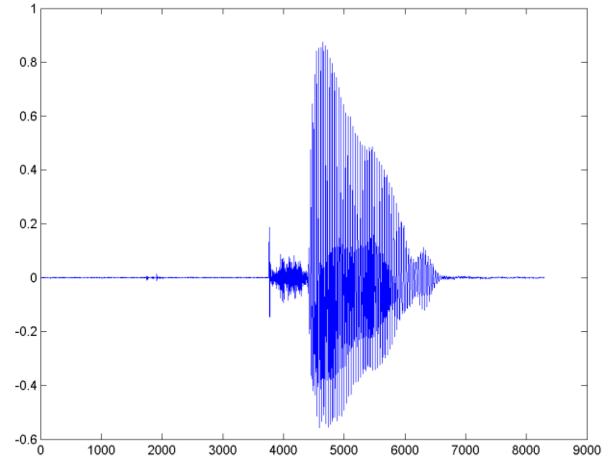
Preemphasized and windowed segment

Pre-emphasizing a speech signal

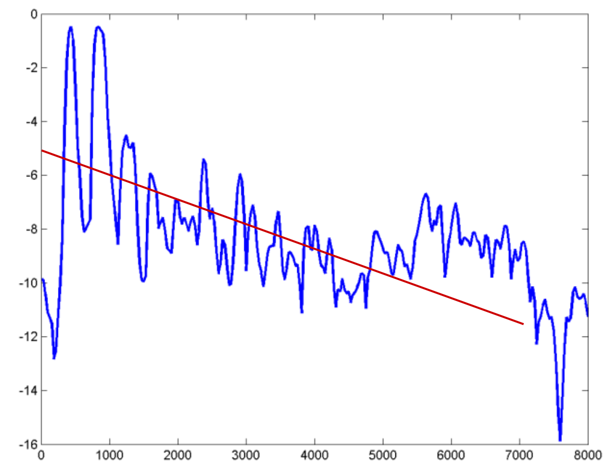
- The spectrum of the speech signal naturally has lower energy at higher frequencies

- This can be observed as a downward trend on a plot of the logarithm of the magnitude spectrum of the signal

- For many applications this can be undesirable
  - E.g. Linear predictive modeling of the spectrum



Log(average(magnitude spectrum))

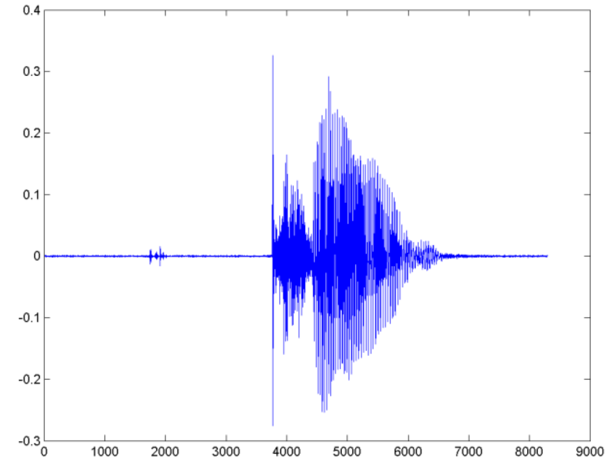


Pre-emphasizing a speech signal

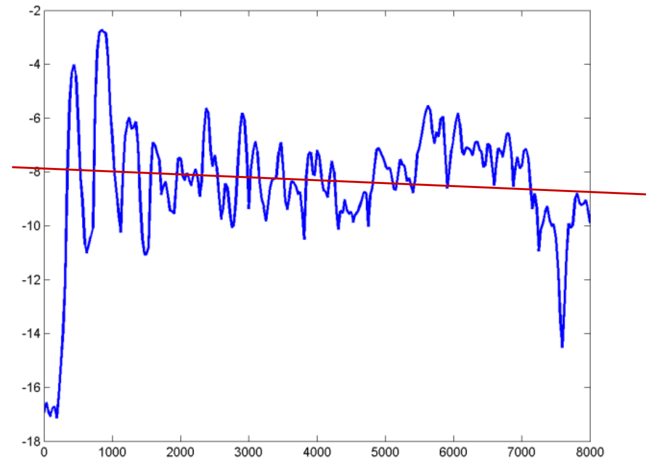
- This spectral tilt can be corrected by preemphasizing the signal
  - $s_{preemp}[n] = s[n] - \alpha * s[n-1]$
  - Typical value of α = 0.95

- This is a form of differentiation that boosts high frequencies

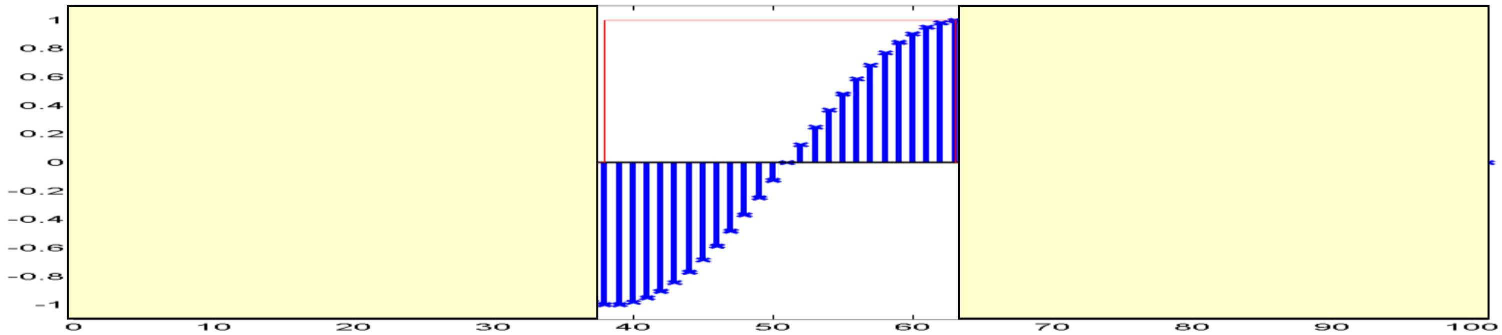
- This spectrum of the preemphasized signal has more horizontal trend
  - Good for linear prediction and other similar methods



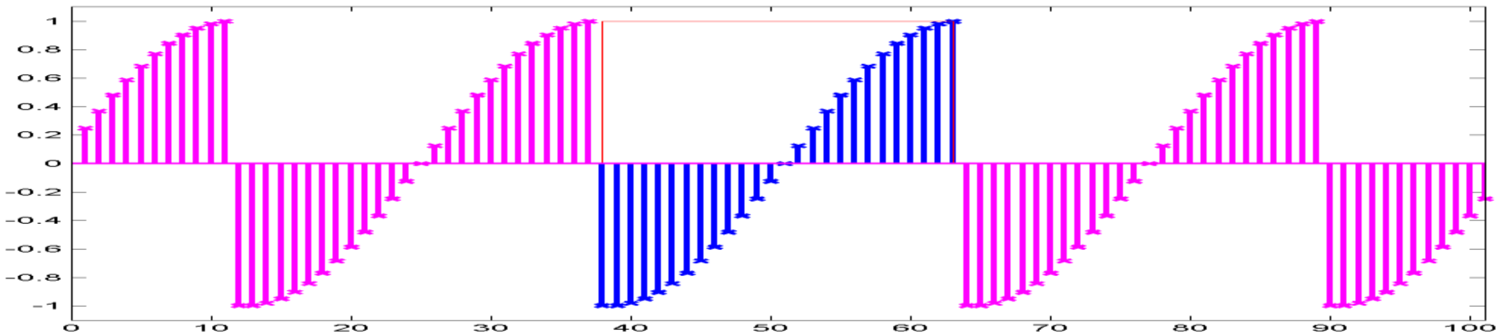
Log(average(magnitude spectrum))



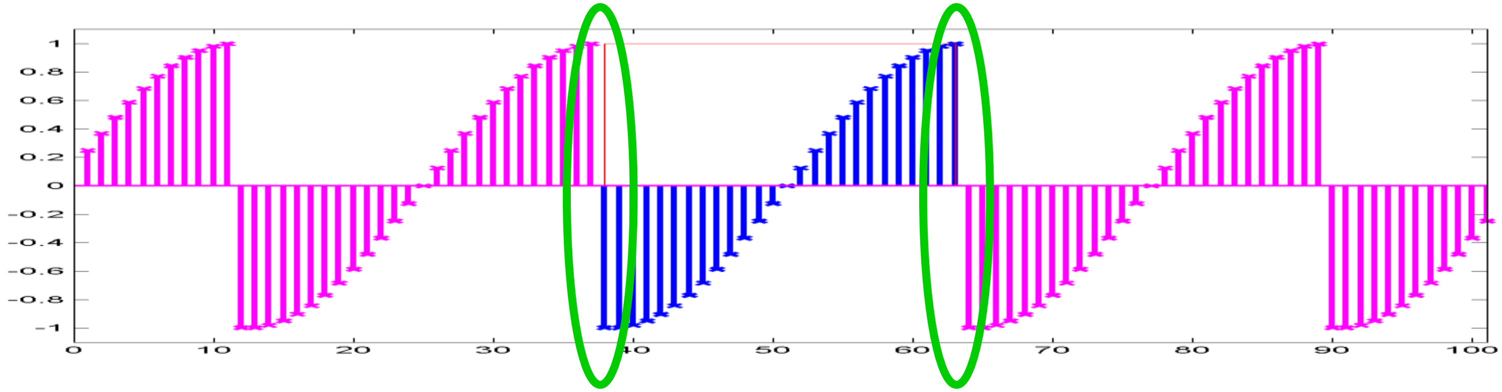
Windowing



- The DFT of *any* sequence computes the Fourier series for an infinite repetition of that sequence
- The DFT of a partial segment of a sinusoid computes the Fourier series of an infinite repetition of that segment, and not of the entire sinusoid
- This will not give us the DFT of the sinusoid itself!

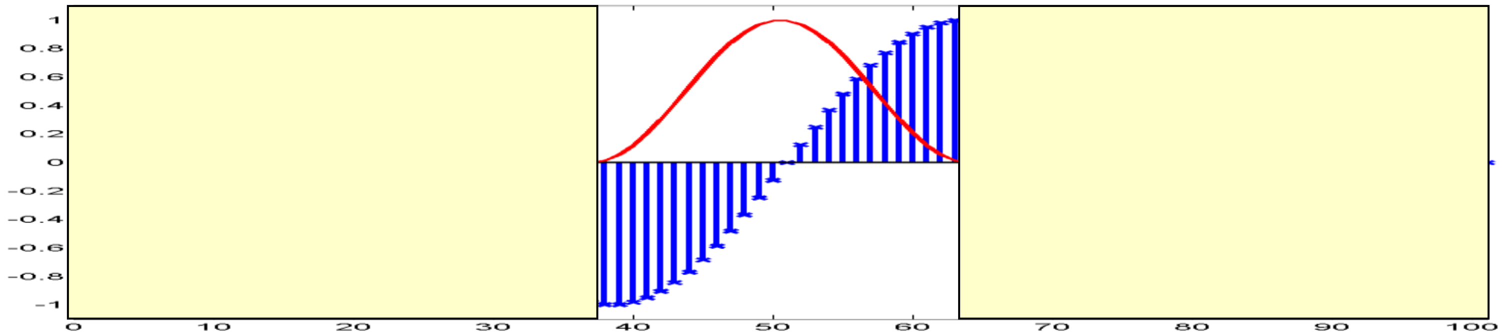


Windowing



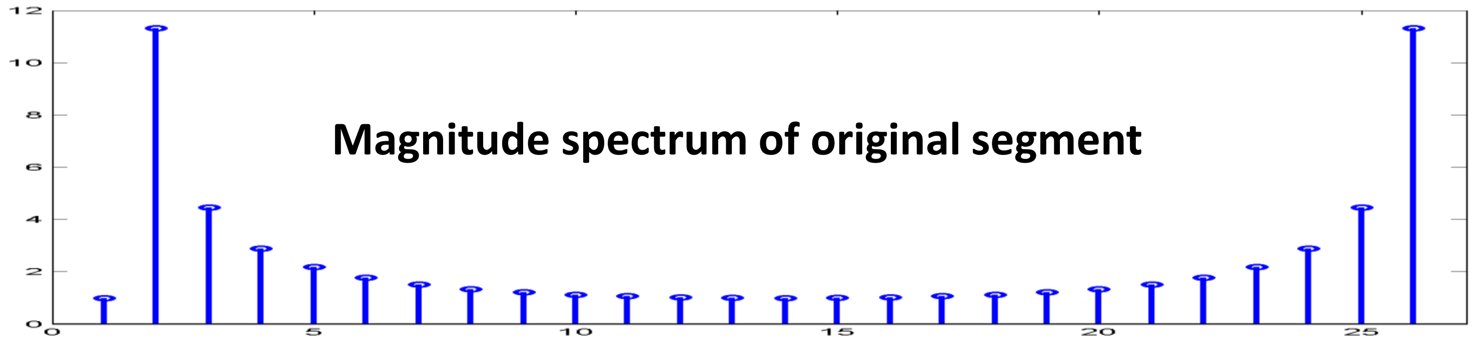
- In addition, the implicit repetition of the observed signal introduces large discontinuities at the points of repetition
  - This distorts even our measurement of what happens at the boundaries of what has been reliably observed
  - The actual signal (whatever it is) is unlikely to have such discontinuities

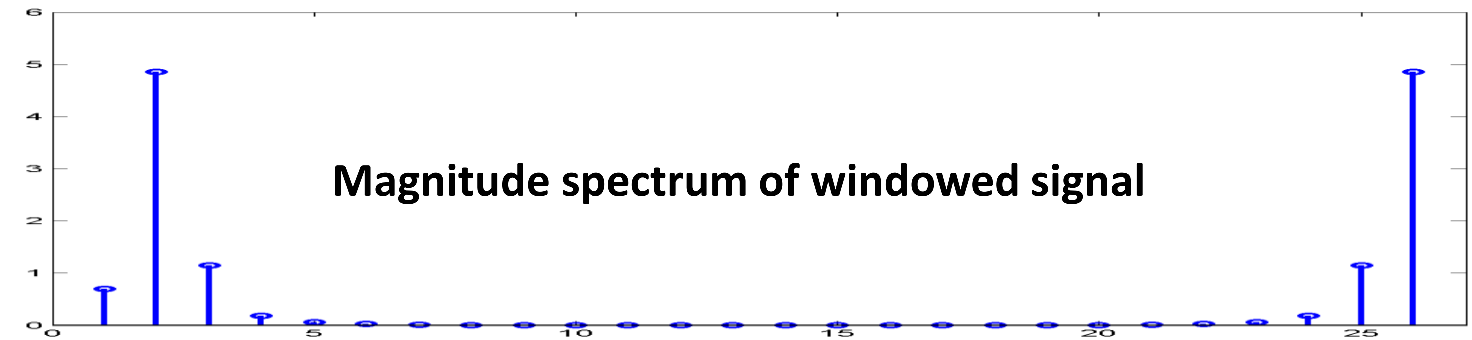
Windowing

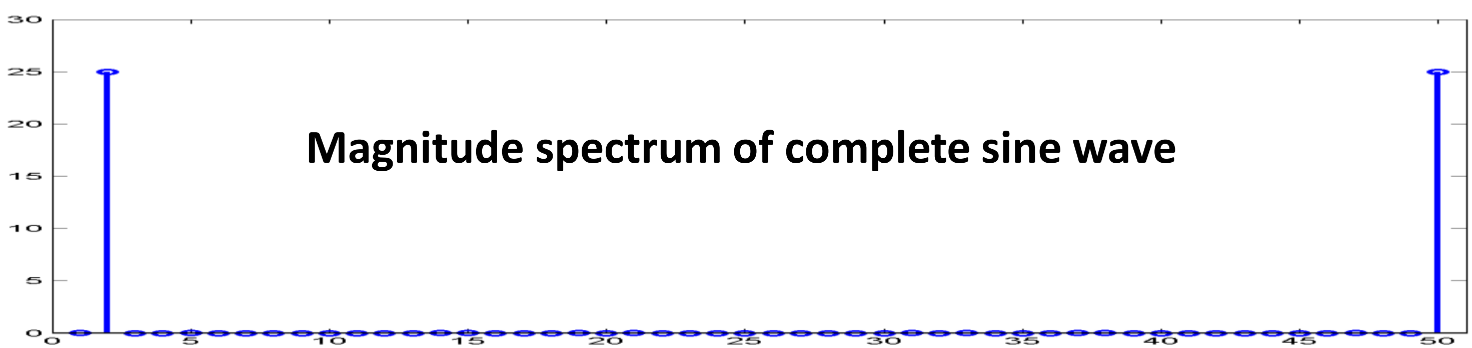


- While we can never know what the signal looks like outside the window, we can try to minimize the discontinuities at the boundaries
- We do this by multiplying the signal with a *window* function
  - We call this procedure windowing
  - We refer to the resulting signal as a "windowed" signal

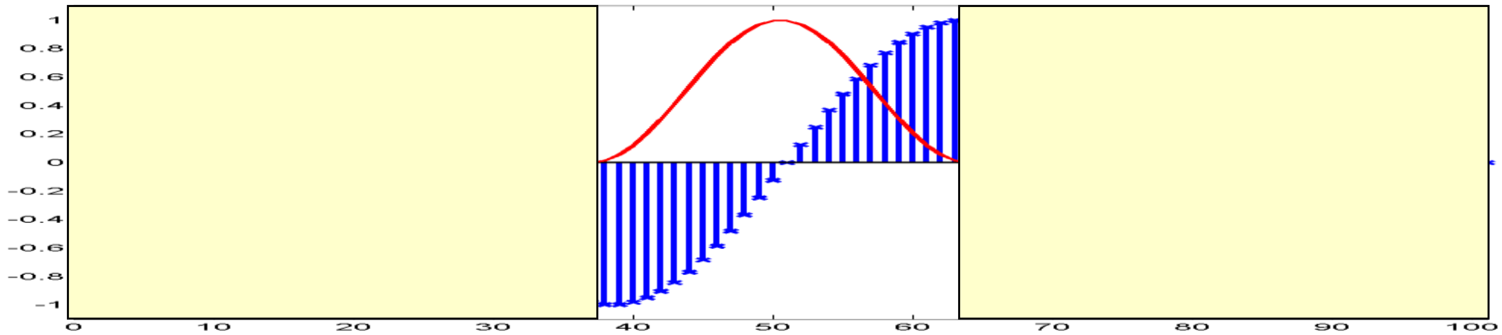
Windowing


Magnitude spectrum of original segment
12
10
8
6
4
2
0
0
5
10
15
20
25


Magnitude spectrum of windowed signal
6
5
4
3
2
1
0
0
5
10
15
20
25

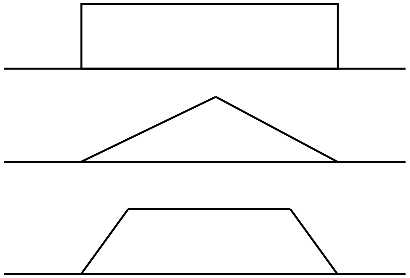

Magnitude spectrum of complete sine wave
30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
40
45
50

Windowing



Geometric windows:

- Rectangular (boxcar)
- Triangular (Bartlett)
- Trapezoid



Cosine windows:

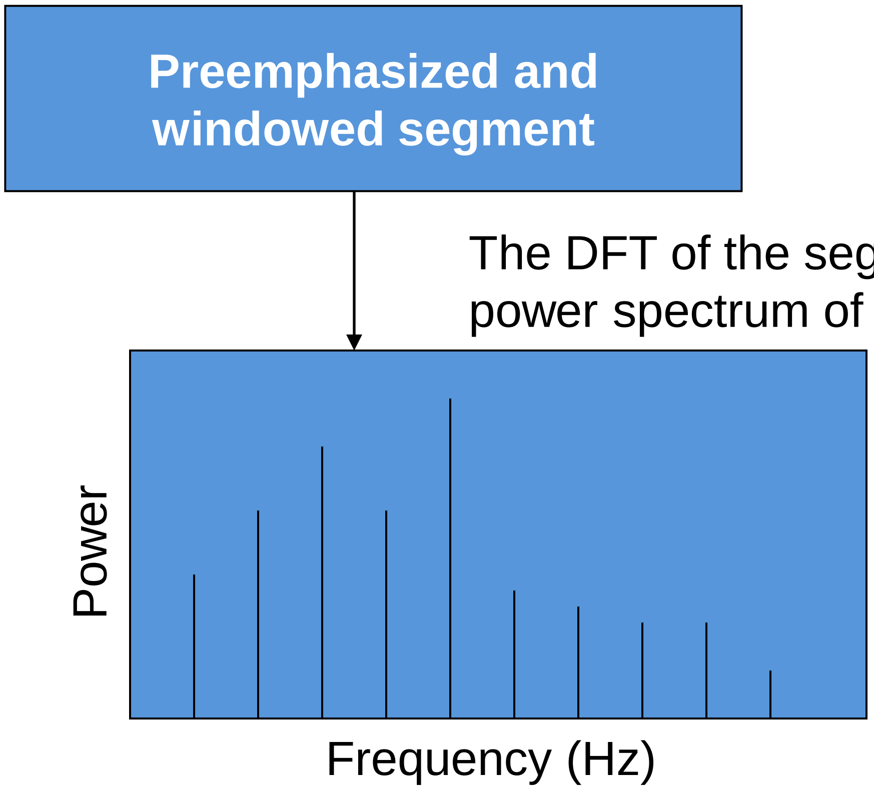
- Window length is M
- Index begins at 0

Hamming: $w[n] = 0.54 - 0.46\cos(2\pi n/M)$

Hanning: $w[n] = 0.5 - 0.5\cos(2\pi n/M)$

Blackman: $0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M)$

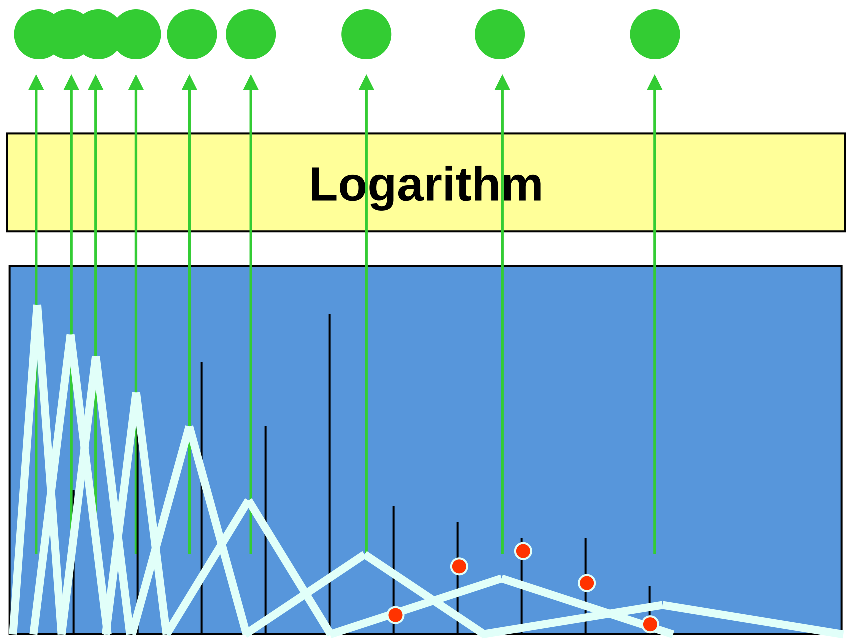
Computing Mel Cepstra (MFCC)



The DFT of the segment, and from it the power spectrum of the segment is computed

= power spectrum

The process of parametrization

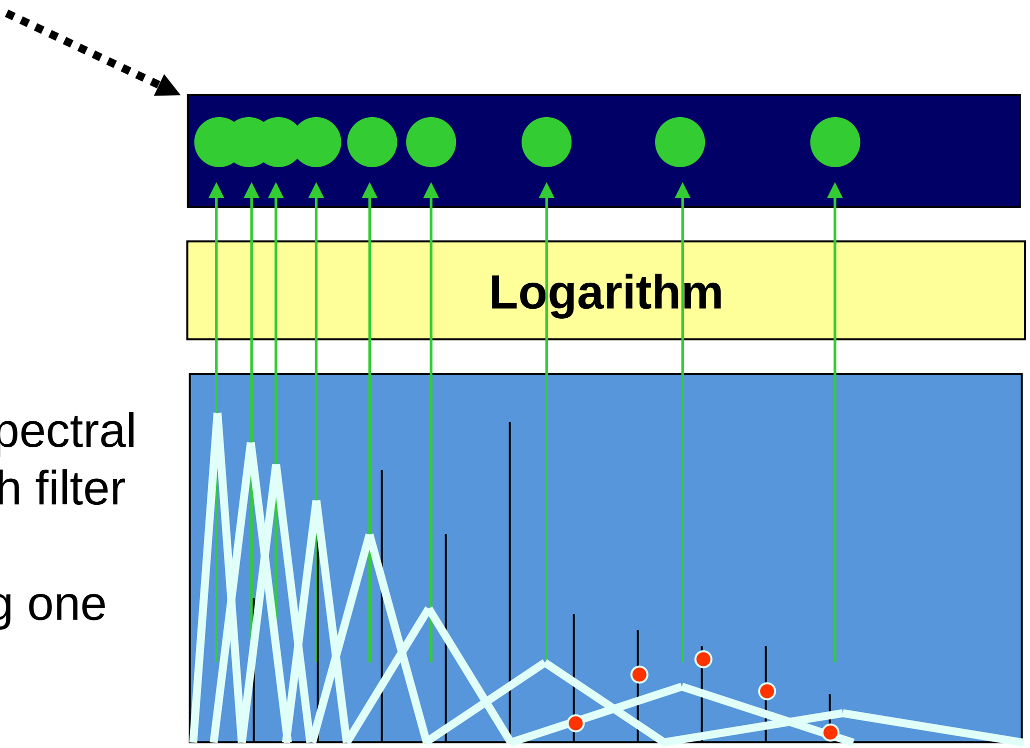


Compress Values

All weighted spectral values for each filter are integrated (added), giving one value per filter

The process of parametrization

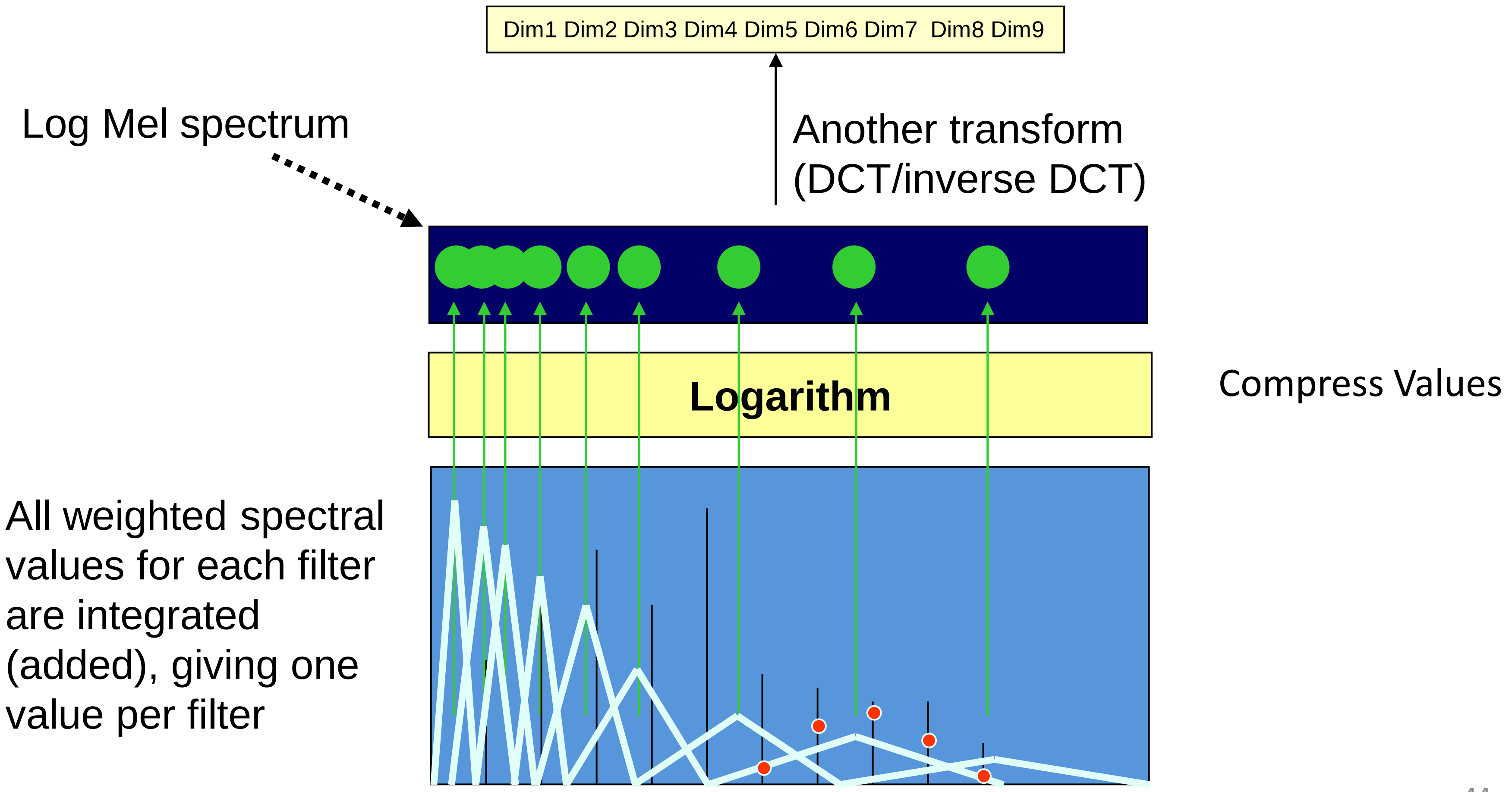
Log Mel spectrum



Compress Values

All weighted spectral values for each filter are integrated (added), giving one value per filter

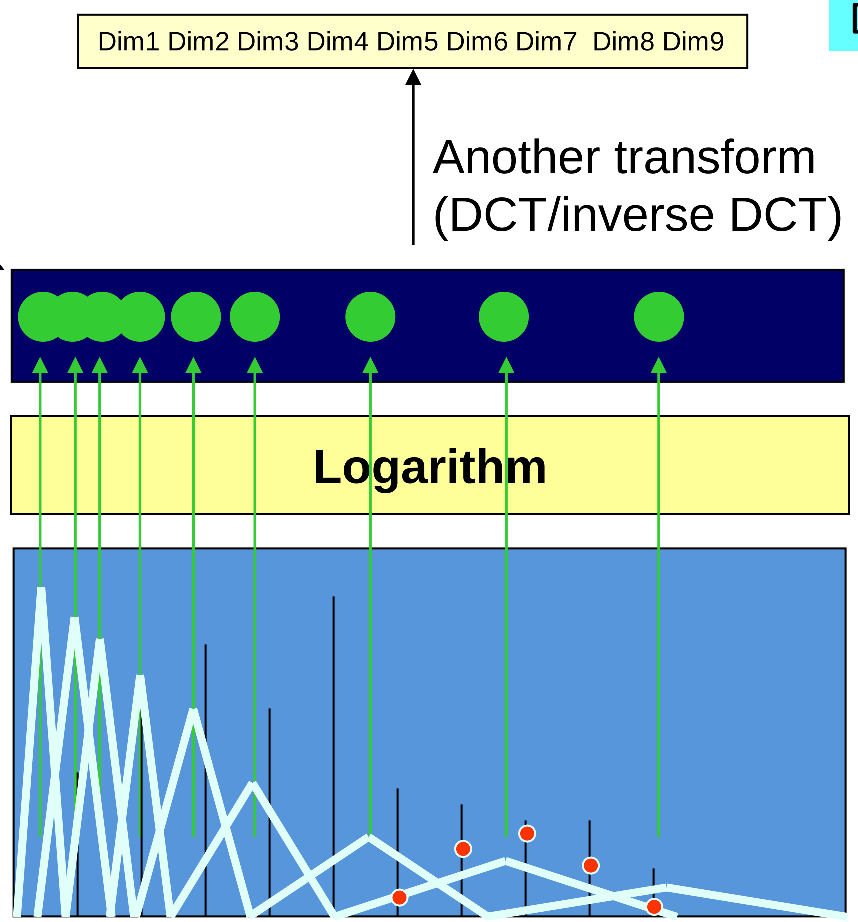
The process of parametrization



The process of parametrization

The sequence is truncated
(typically after 13 values)

Dimensionality reduction

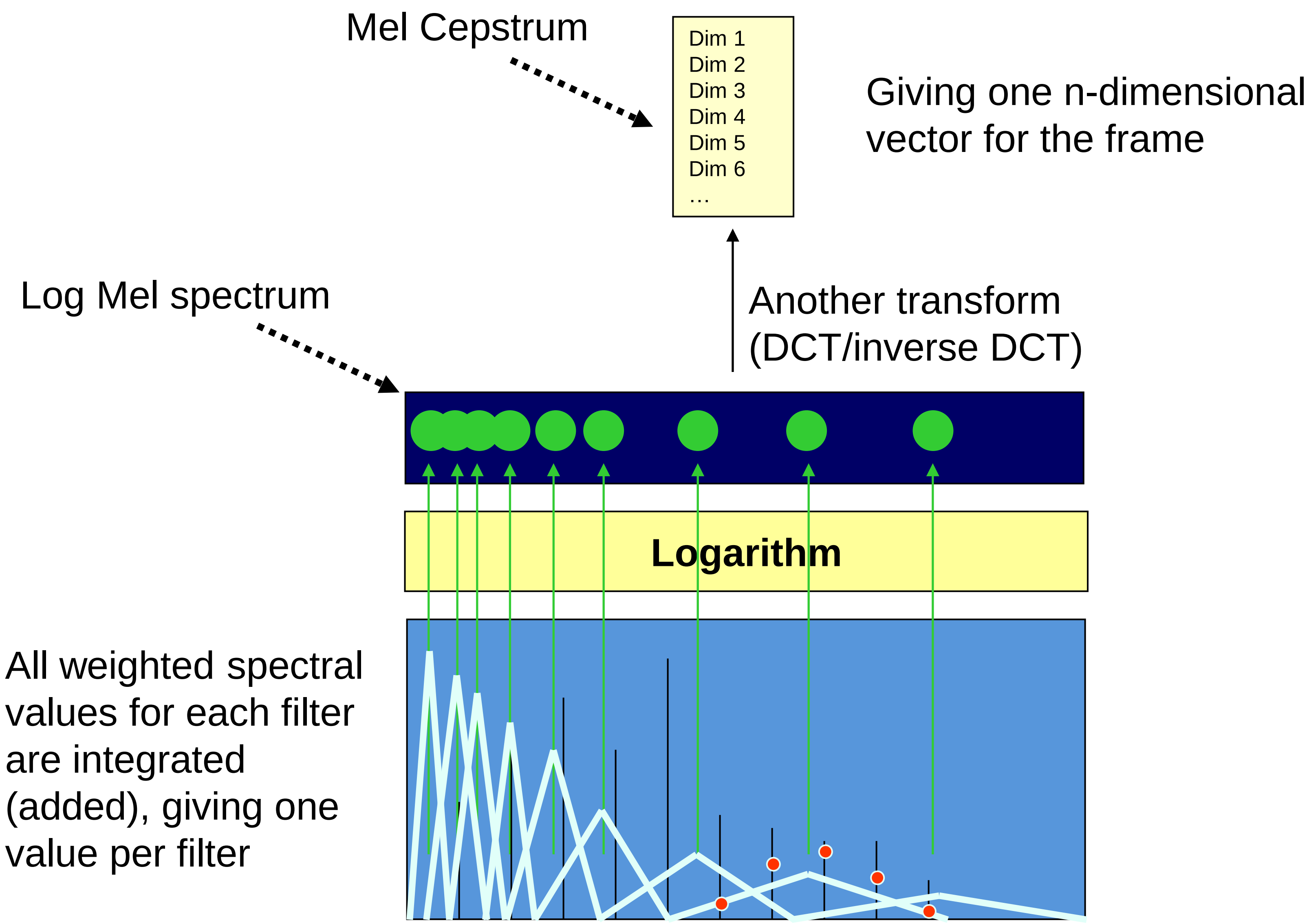


Log Mel spectrum

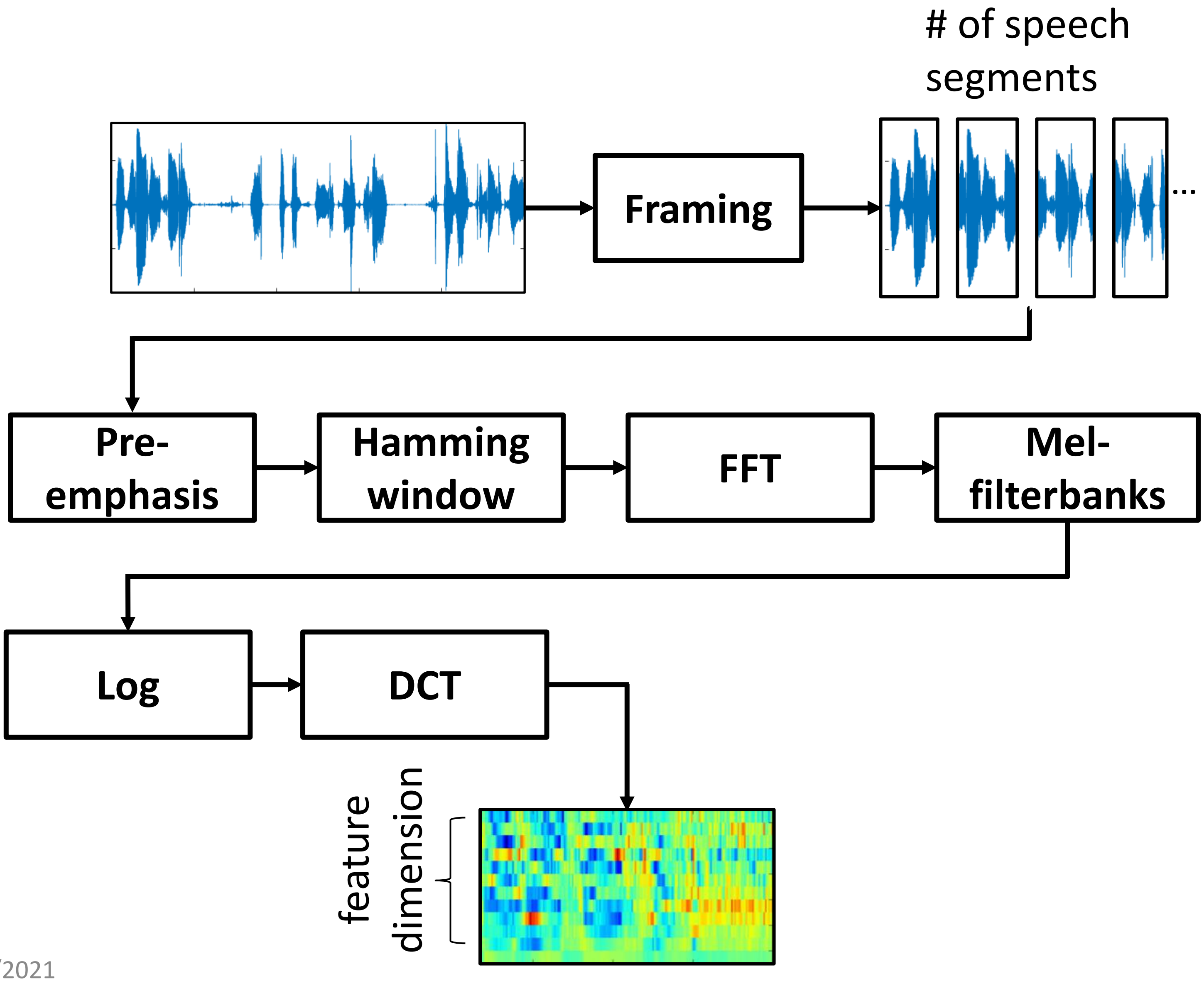
Another transform
(DCT/inverse DCT)

All weighted spectral values for each filter are integrated (added), giving one value per filter

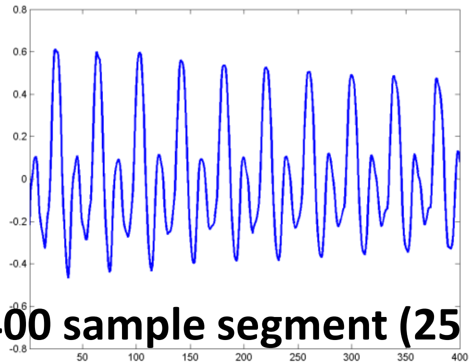
The process of parametrization



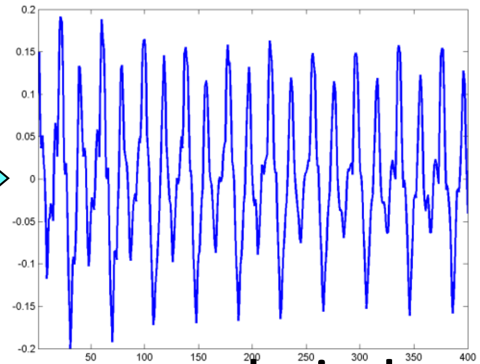
Extracting cepstra



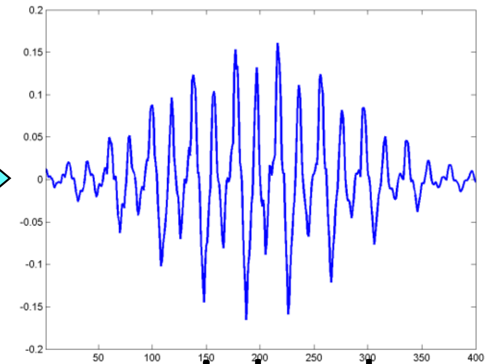
An example segment



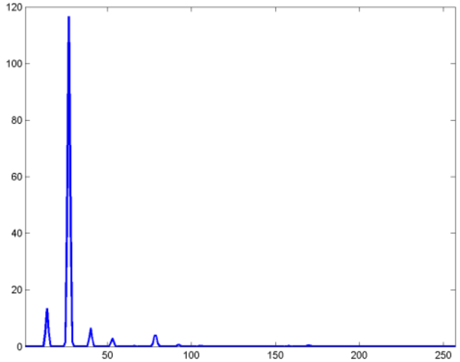
400 sample segment (25 ms) from 16khz signal



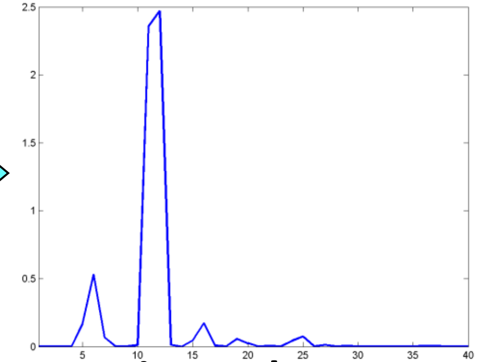
preemphasized



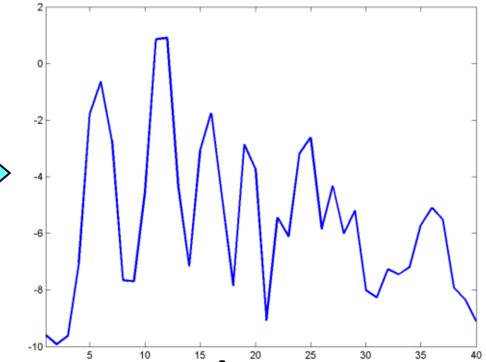
windowed



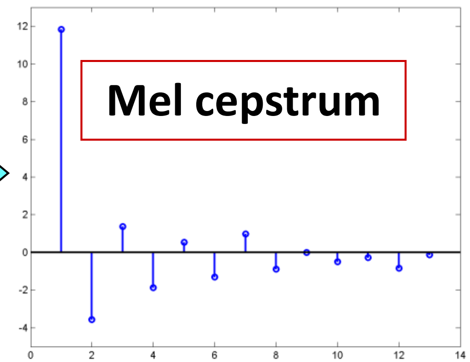
Power spectrum



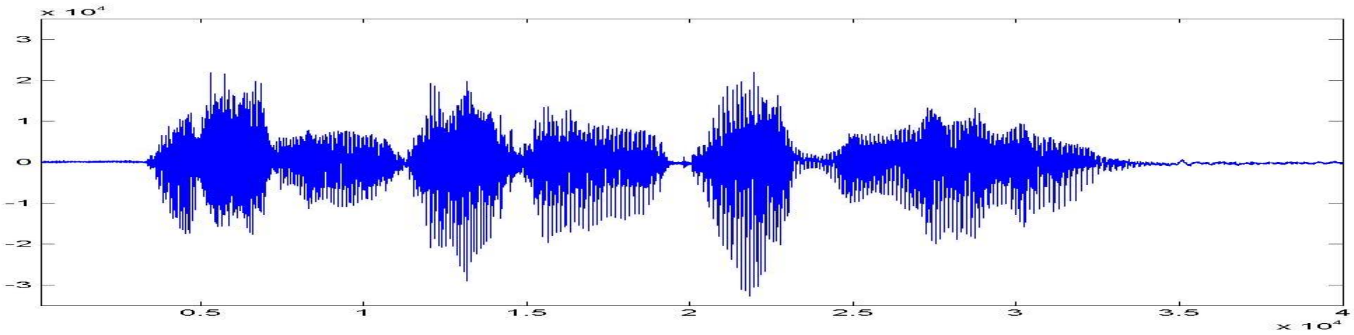
40 point Mel spectrum

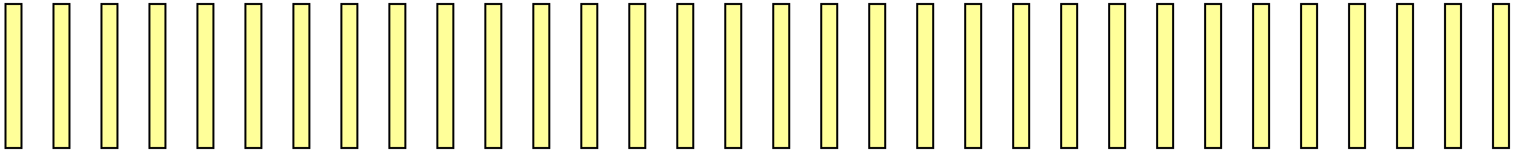


Log Mel spectrum



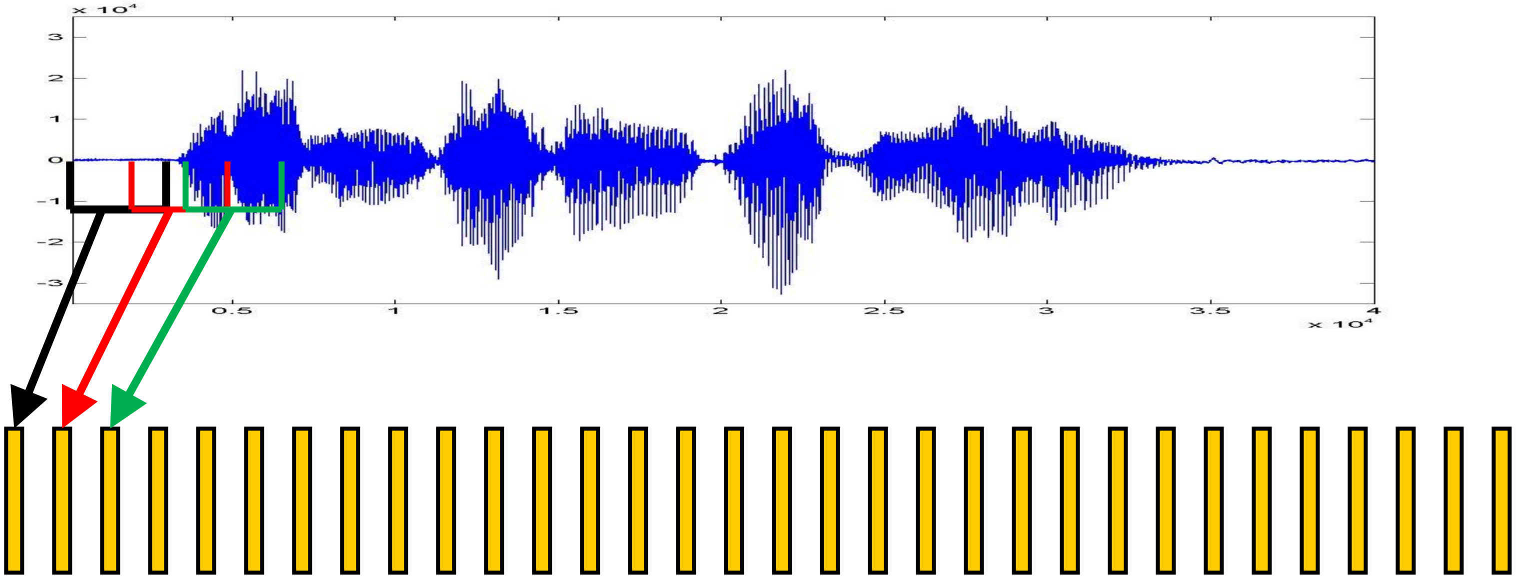
The process of parametrization





The entire speech signal is thus converted into a sequence of vectors. These are Mel cepstral vectors (also known as MFCC, or Mel Frequency Cepstral Ceofficients).

In general, audio is represented as features



Step 1

- The time domain audio signal is transformed into a sequence of different measurements
  - These are usually vectors, and are called feature-vectors
  - The process is called **parametrization**

Step 2

- Either the features are used directly or
- Secondary features are extracted from them
  - Secondary features are then compared

Variations to the basic theme

- Perceptual Linear Prediction (PLP) features:
  - ERB filters instead of MEL filters
  - Cube-root compression instead of Log
  - Linear-prediction spectrum instead of Fourier Spectrum
- Auditory features
  - Detailed and painful models of various components of the human ear

Audio analysis overview

- Feature computation (selected)
  - Spectra
  - Spectrograms
  - Mel-cepstra
  - i-vectors
  - Supervectors
  - Bag-of-words
  - NN-based features
  - Visual features
  - Videographic features

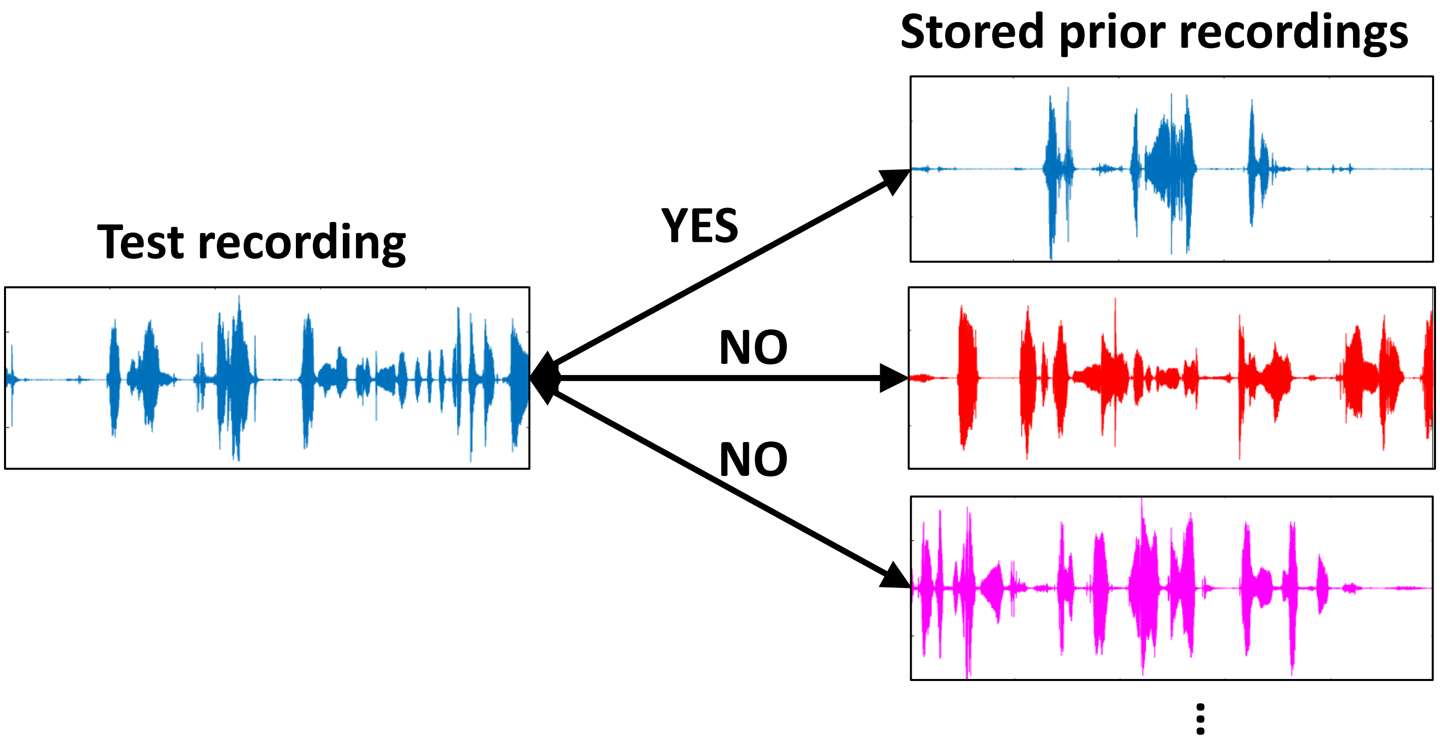
- Specific applications of audio processing (selected)
  - Audio authentication
  - Audio enhancement
  - Audio fingerprinting
  - Audio localization
  - Audio object detection
  - Audio retrieval
  - Audio summarization
  - Environmental profiling
  - Geolocation
  - Source Identification
  - Source separation
  - Speaker identification
  - Speaker profiling
  - Speaker verification
  - Speech recognition
  - Speech separation

- Key analysis techniques (selected)

Macro and micro features

- **macro-features**
  - derived from large windows, or the entire signal
  - Derived as secondary features by aggregating features derived from short-duration windows
    - typically 25 ms wide
    - Adjacent segments typically overlap by 15 ms
- **micro-features**
  - derived from very small (duration < 25ms) segments of the signal

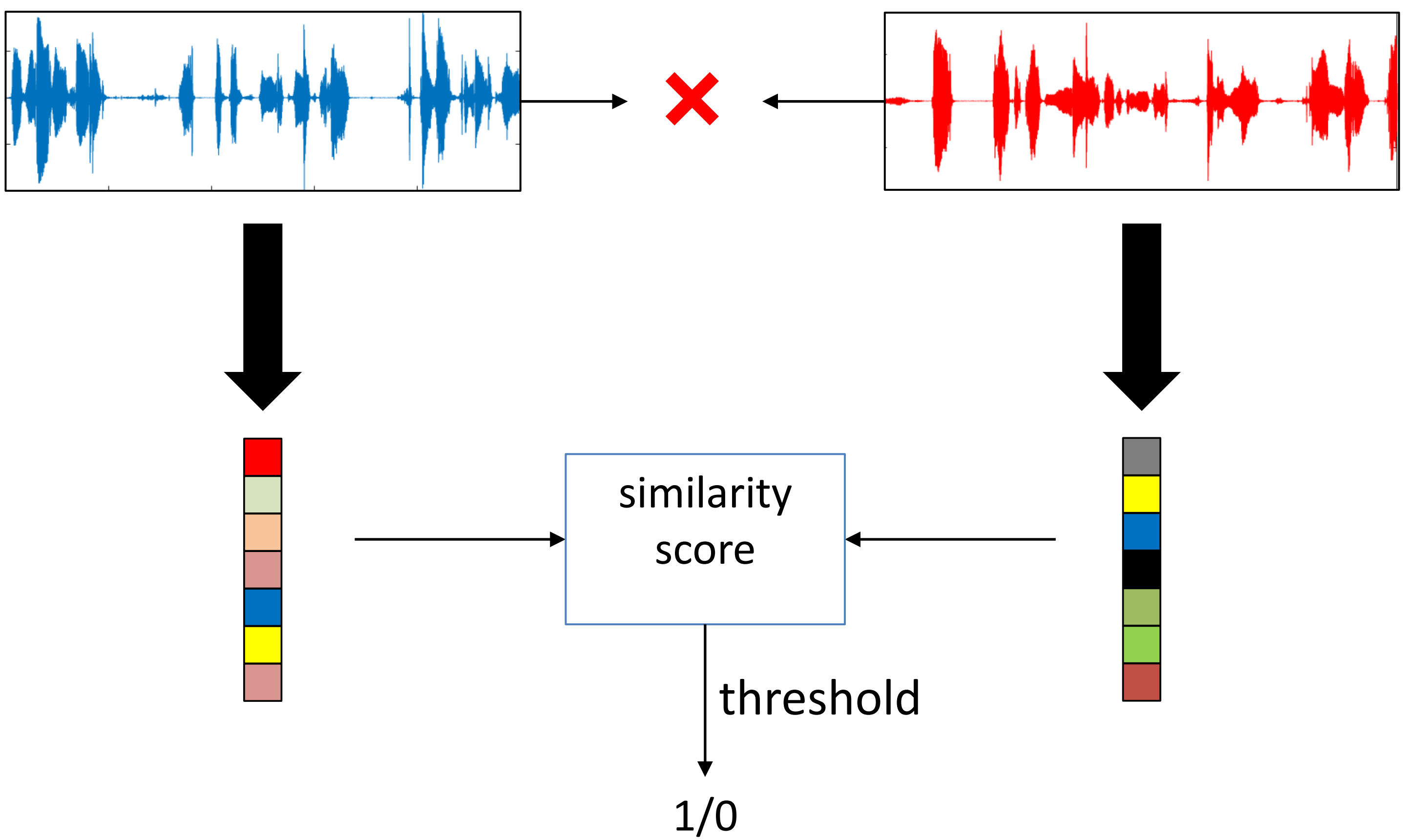
Audio/Speaker Matching



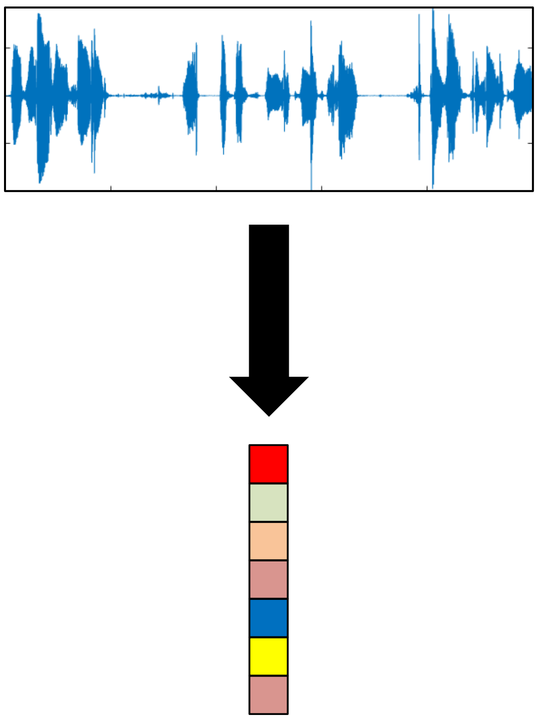
- Determine if the speaker in a “test” recording is the same as that in a previously obtained recording

Approach

- Extract a “representation vector” from the recordings
- Identify matches by comparing the representation vectors
- Key challenge: Learning the right representations for the recordings



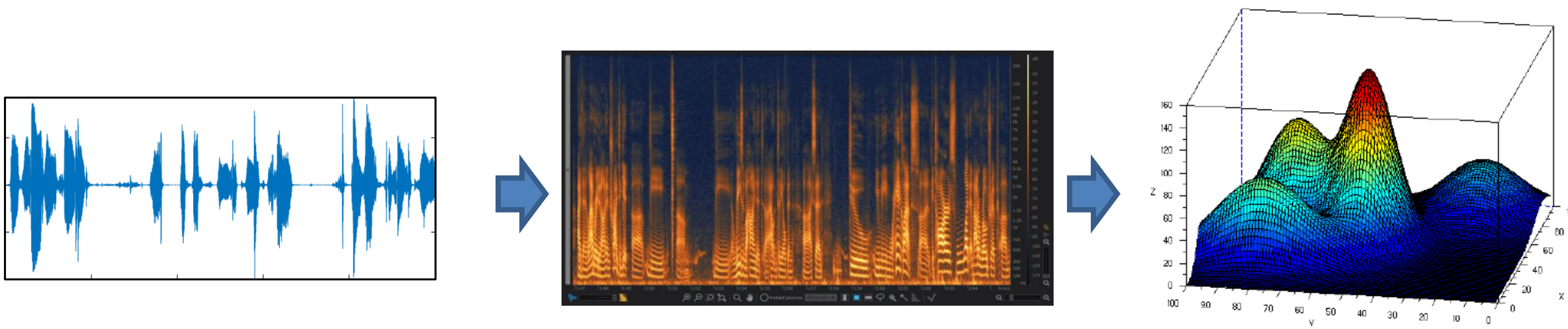
Extracting a representation vector from a recording



- Must focus on *overall* characteristics of the recording
  - Rather than instantaneous patterns, which may never be repeated

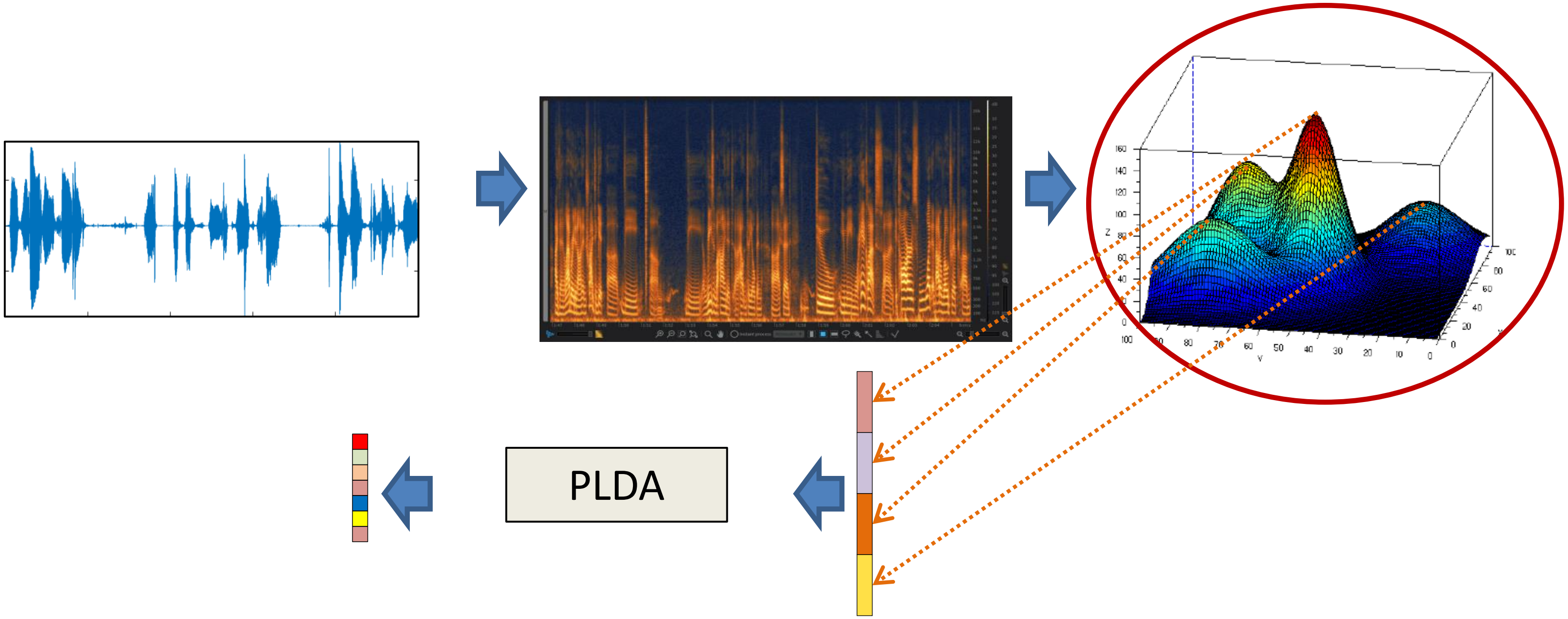
- Must reduce *variable-sized* recordings to a fixed-size vector
  - Required for comparisons

Features for matching



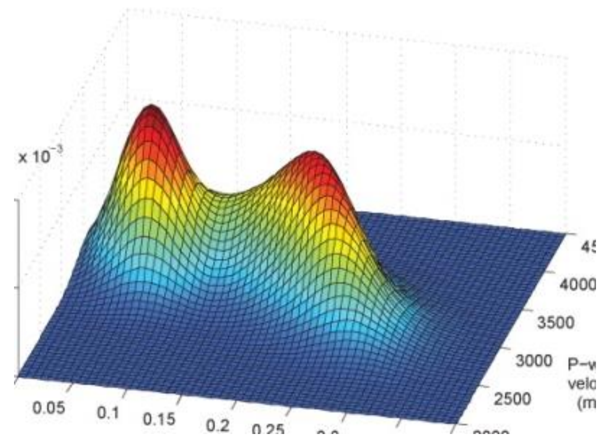
- Convert the audio recording to a sequence of *spectral* vectors, that together comprise a "spectrogram"
  - May applications use *cepstral* vectors instead, which are derived from the spectral vectors
- Estimate the *distribution* of the vectors in the recording
  - Hypothesis: The ID of the speaker is captured by this distribution
    - Modes capture both linguistic and acoustic tendencies

Features for matching



- Learn a Gaussian mixture density from the collection of *cepstral* vectors in the recording
  - Adapt a "universal background model" to do so
- Concatenate the parameters of the GMM into a single vector called a "**Super-Vector**"
- Reduce the dimensionality of the super-vector through discriminative factor analysis
  - "**Probabilistic Latent Discriminant Analysis**" (PLDA)

Supervectors and I-vectors

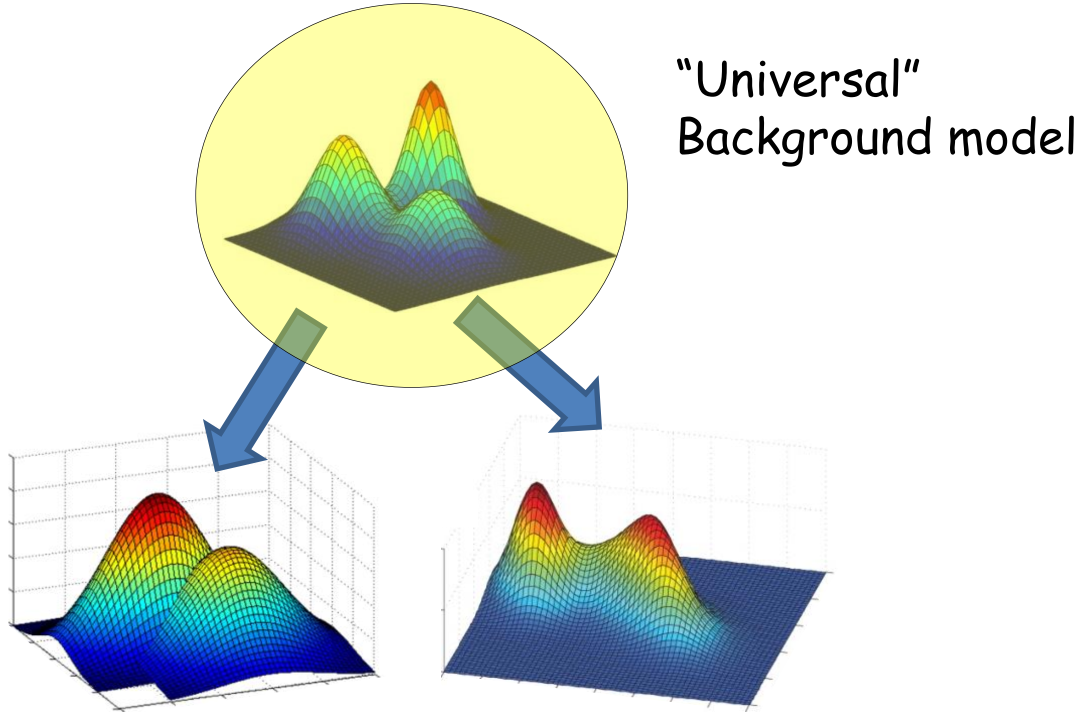


- In practice, there is generally insufficient data in the recordings to compute a distribution
  - So we adapt a "Universal" model to the individual recordings
- The adapted distribution captures the overall statistical characteristics of the recording
- We can represent this using a *single* vector, obtained by simply concatenating the means of the individual Gaussians in the mixture

$$S = [\mu_1 \quad \mu_2 \quad \dots \quad \mu_L]$$

  - μ_i is the mean of the ith Gaussian
  - L is the total number of Gaussians
- This vector is called a ***Supervector*** representing the distribution of spectral vectors derived from the recording

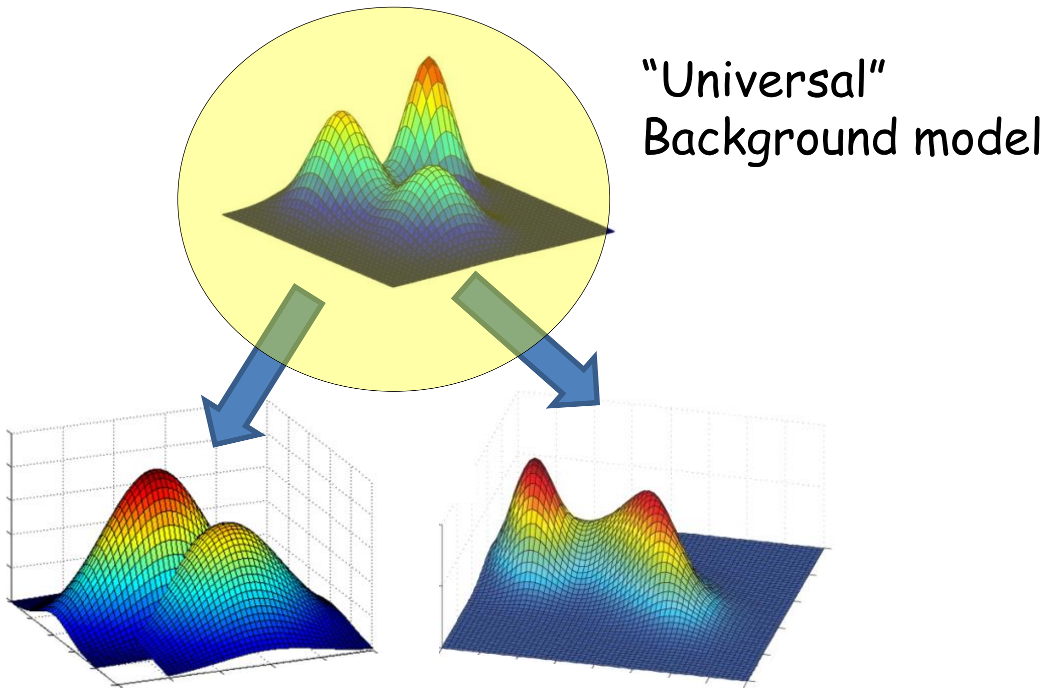
Statistical Distribution Matching



Problems with this approach

- Needs enough data to estimate distribution
  - Even with all the dimensionality reduction, needs several minutes in each recording for reliable results
- Average's information across time: Ignores temporal structure.
  - More generally, ignores structural/phonetic nature of speech,
- No reliable mechanism to *incrementally* include evidence from additional data
- Not robust to changes due to age, disease etc.

Statistical Distribution Matching

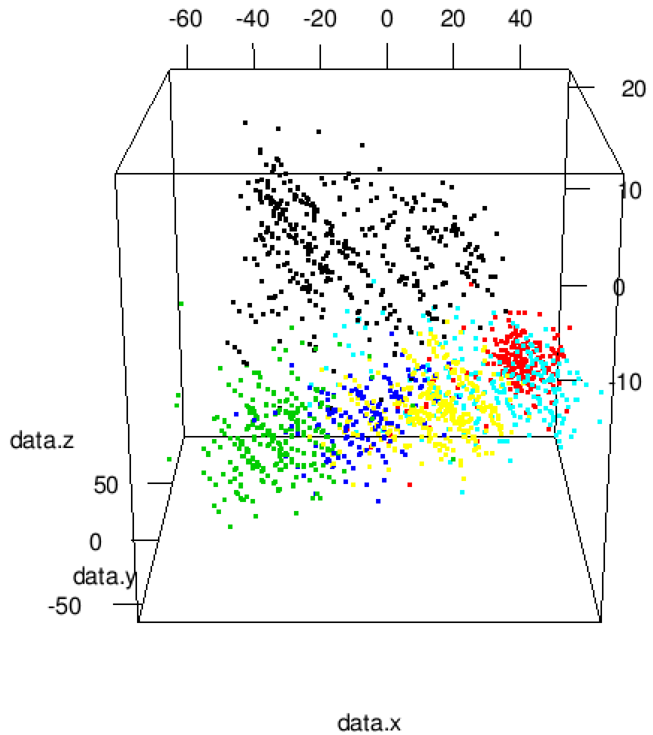


- To further reduce data requirement, various dimensionality-reducing techniques like **factor analysis** and **Linear Discriminant Analysis (LDA) (for classification)** are employed
- With jargon such as **I-vectors**, "**Total variability space**", **PLDA**..

I-vectors: concept

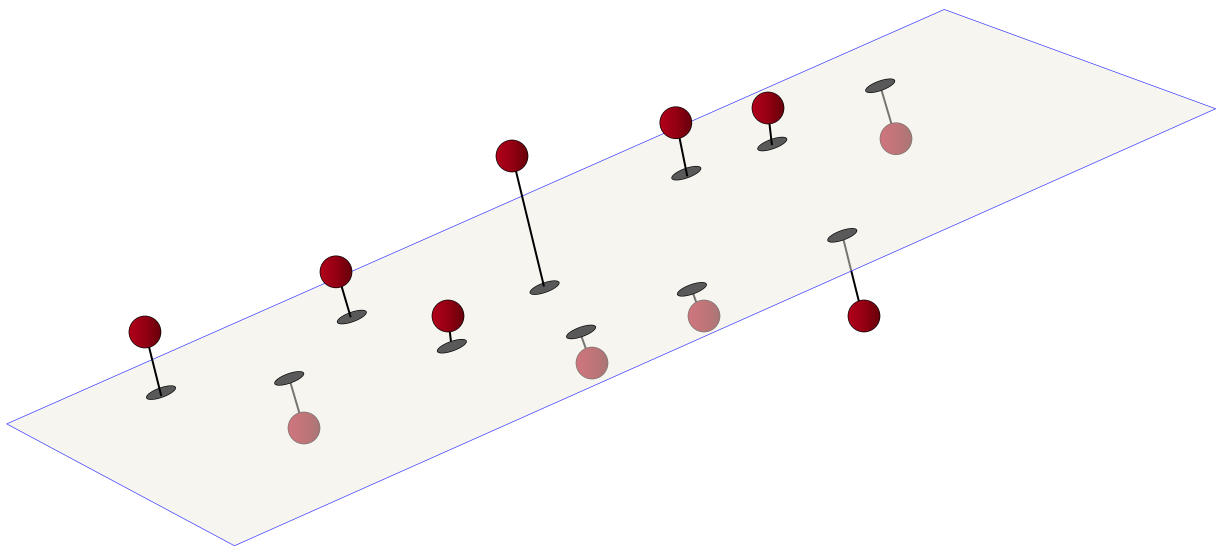
- “I”-vectors are derived from super-vectors through “factor analysis”
- Factor Analysis is in turn an extension of “Principal Component Analysis”..

Principal Component Analysis



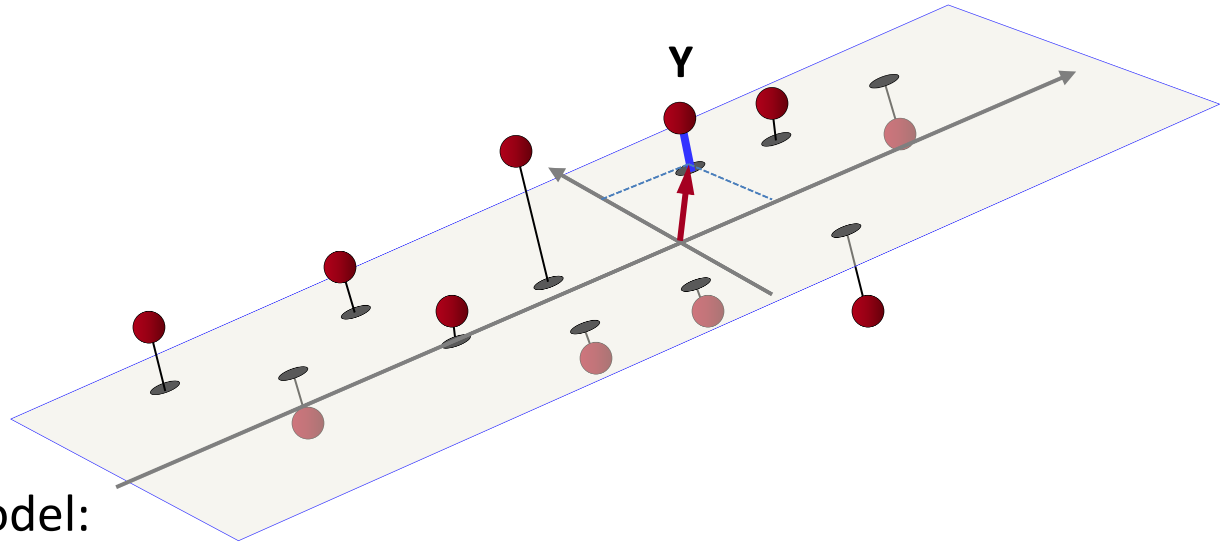
- In high dimensions, most data lie on or close to lower dimensional "linear manifold" -- a hyperplane
- Principal Component Analysis and attempts to find this hyper plane and "place" the data on it

Principal Component Analysis



- PCA finds a plane such that the total squared distance from the points to their "shadow" on the plane is minimum
  - The "shadow" is the "projection" of the data on the plane
- The "shadow" is assumed to be the true data, the off-plane component is the noise

Principal Component Analysis

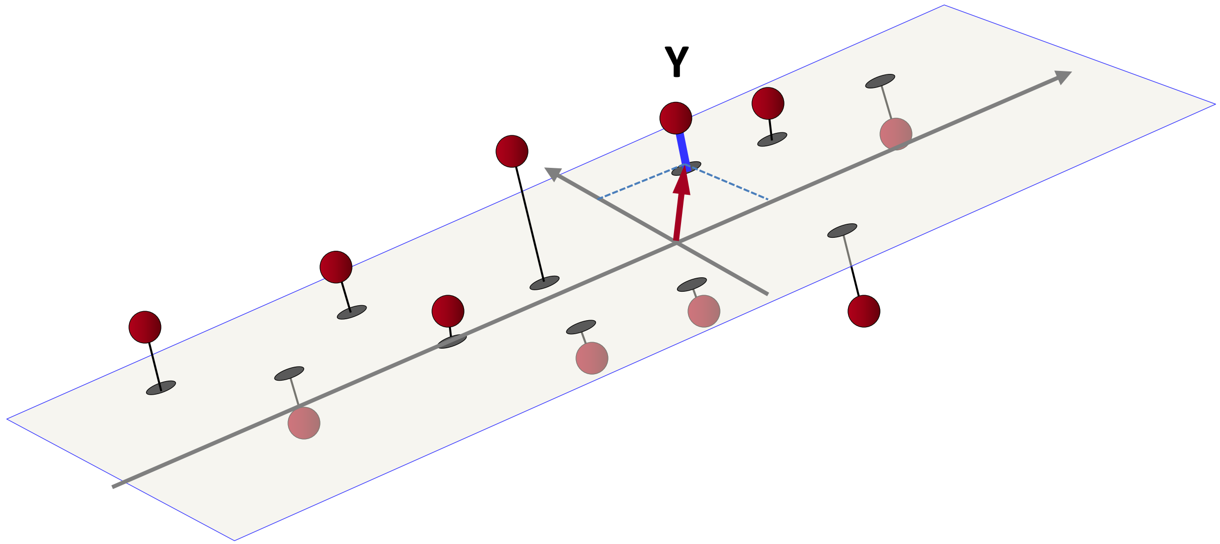


- PCA model:
  - The actual data are formed on the plane (red arrow on plane)
  - The noise is *orthogonal* to the plane

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{X} + \boldsymbol{N}$$

  - $\boldsymbol{Y}$ is final vector (3D vector in our example, indicated by red ball)
  - $\boldsymbol{B}$ is a matrix with the unit vectors for the "bases" (long grey arrows, coordinate axes of plane,
    - 3x2 matrix in our example: 2 vectors in 3 dimensions
  - $\boldsymbol{X}$ represents coordinates of the shadow point (2D vector in our example)
  - $\boldsymbol{N}$ is the noise (blue line)

Principal Component Analysis

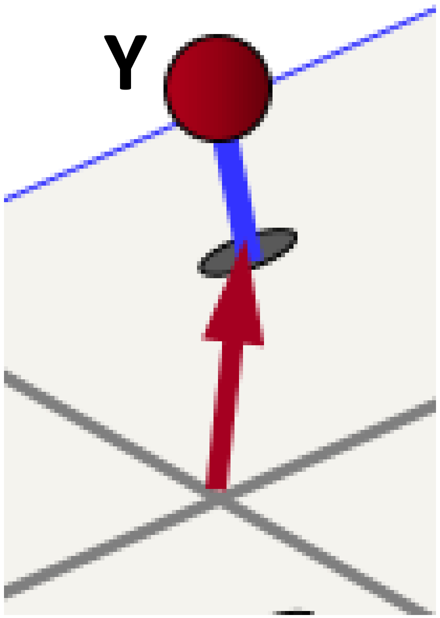


- **PCA statistical generative model**:

$$\boldsymbol{Y} = \boldsymbol{BX} + \boldsymbol{N}$$

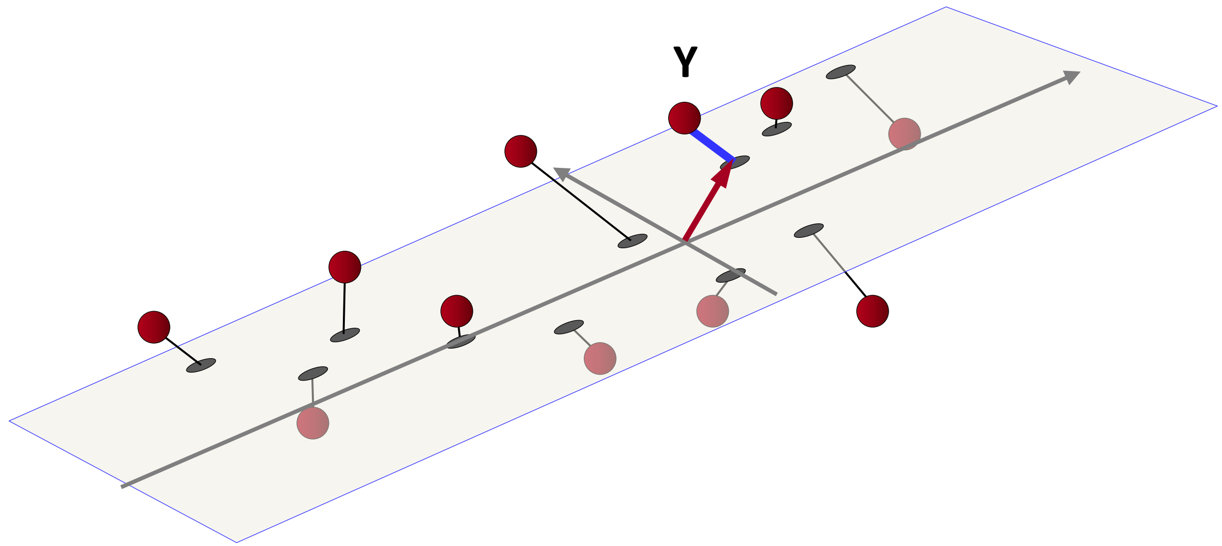
  - $\boldsymbol{X}$ is drawn from a 0-mean Gaussian distribution with a diagonal covariance matrix
    - Co-ordinates are uncorrelated
  - $\boldsymbol{N}$ is drawn from a 0-mean Gaussian with a ***low-rank*** covariance
    - Rank is 1 in our example
    - More generally, for D-dimensional data, explained through K-dim PCA, rank of noise covariance is D-K
- ***PCA challenge:*** *Given a collection of $\boldsymbol{Y}$ vectors find bases $\boldsymbol{B}$ of the data plane and the coordinates $\boldsymbol{X}$ on the plane for each vector $\boldsymbol{Y}$*
  - **Find the grey lines, and the location of the shadow for each point**
- Obtained through a maximum-likelihood estimator, which is your familiar PCA

Inadequacy of PCA



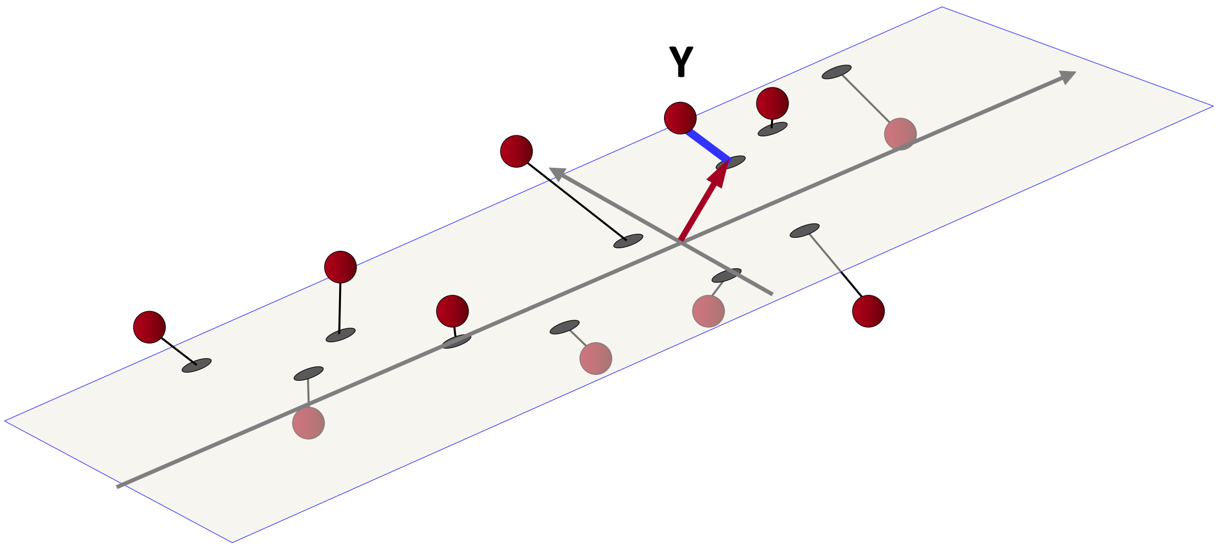
- Assumes noise is always *orthogonal* to the data
- Not a reasonable assumption in most cases
  - e.g. in speech, noise may also sound a bit like speech
  - i.e. noise is not perpendicular to the plane, although it does not *lie* on the plane

Factor Analysis



- Factor analysis:
  - The noise is not required to be perpendicular to the plane
    - May be at any angle to it
- Factor analysis model for the data:
  - Data are formed on the plane
  - A random noise is added to the data

Factor Analysis

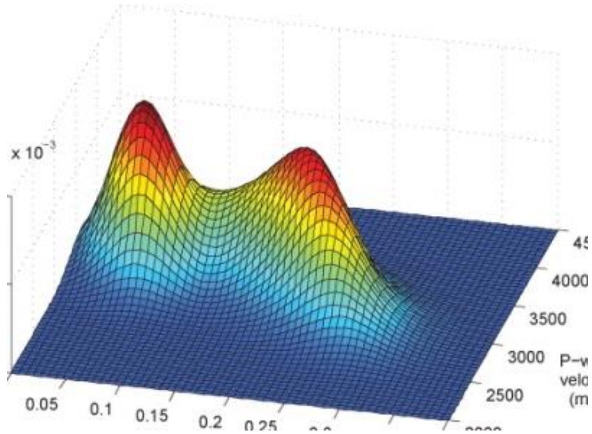


- Factor Analysis statistical generative model:

$$\boldsymbol{Y} = \boldsymbol{BX} + \boldsymbol{N}$$

  - $\boldsymbol{X}$ is drawn from a 0-mean Gaussian distribution with a diagonal covariance matrix
  - $\boldsymbol{N}$ is drawn from a 0-mean Gaussian with a ***full-rank*** covariance
    - Rank is 3 in our example
    - More generally, for D-dimensional data, explained through K-dim FA, rank of noise covariance is D
- ***FA challenge:*** *Given a collection of $\boldsymbol{Y}$ vectors find bases $\boldsymbol{B}$ and the coordinates $\boldsymbol{X}$ on the plane for each vector $\boldsymbol{Y}$*
- The coordinates $\boldsymbol{X}$ for any $\boldsymbol{Y}$ are called the *factors* of $\boldsymbol{Y}$
- Must be estimated through an iterative *expectation maximization algorithm*

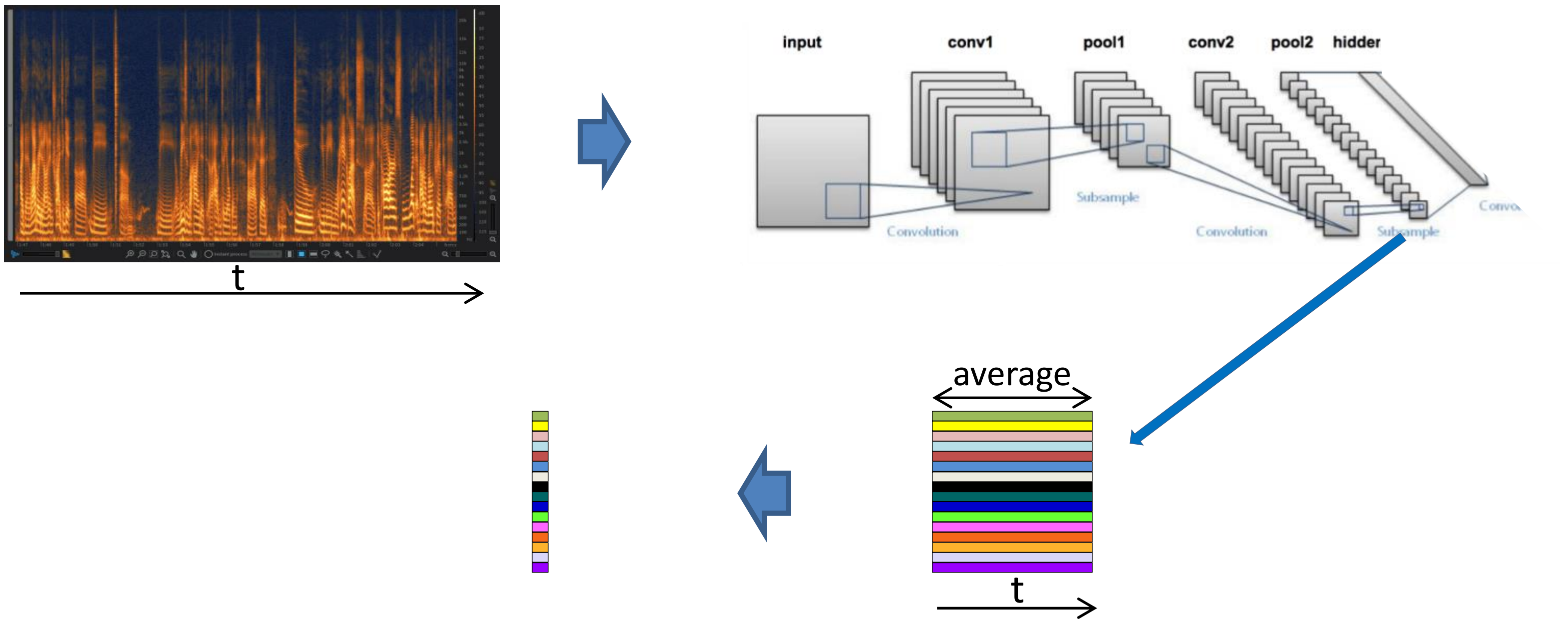
I-vectors applied to speaker identification



$$S = [\mu_1 \quad \mu_2 \quad \cdots \quad \mu_L]$$

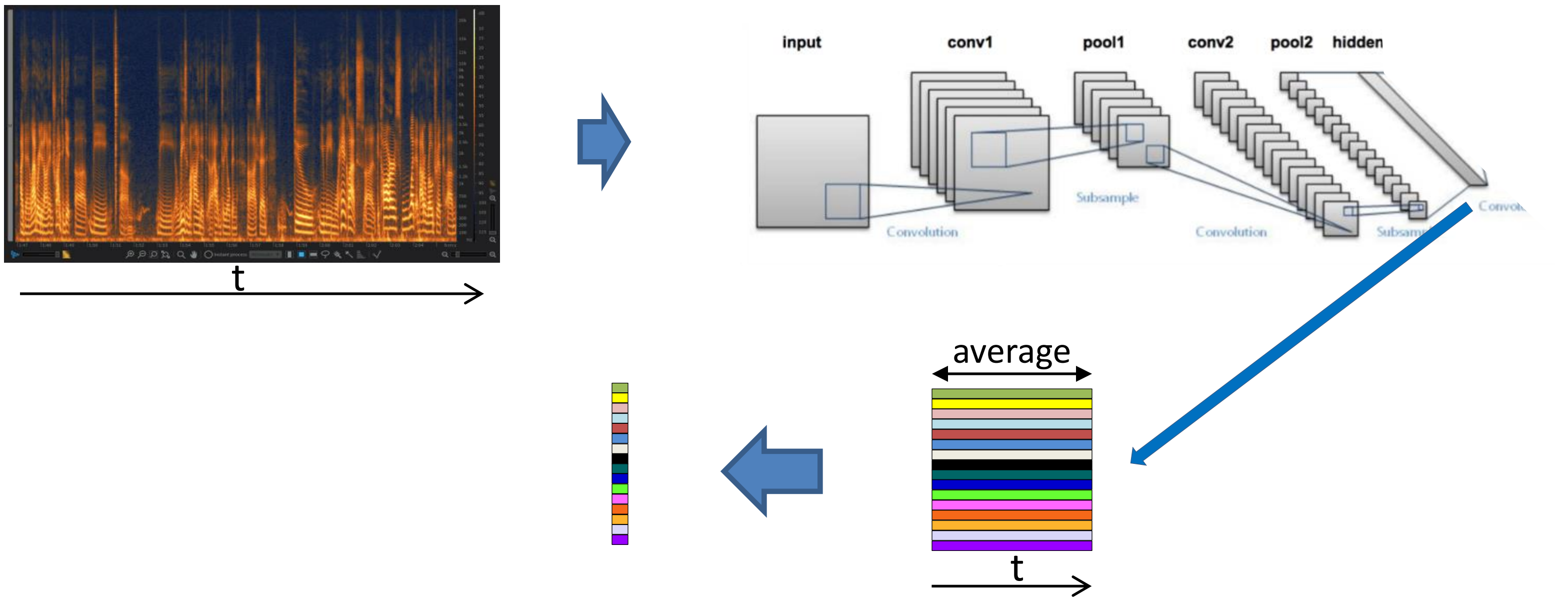
- Each recording from every speaker gives us a single super-vector
- Factor analysis is applied to the collection of super-vectors
- The ***factor*** vector corresponding to each super-vector is called the "I"-vector for that recording
- I-vectors of two recordings can be directly matched to determine if they are from the same speaker or not
  - Matching is done using machine learning algorithms

CNNs for feature extraction



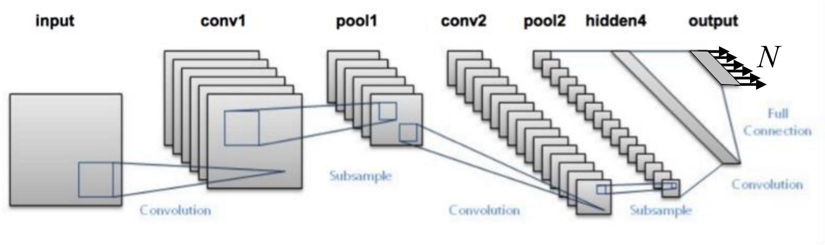
- Treat the spectrogram as an image
- Run a "convolutional neural network" over the image
  - Will give you several scaled-down versions of the image, called "maps" as output
- Average the final maps across time to derive a fixed-size feature

Neural Networks/CNNs for feature extraction

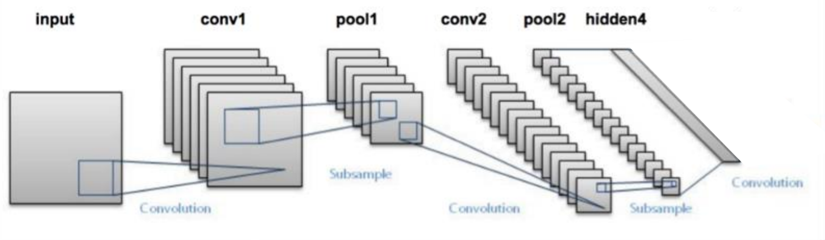


- "Filters" capture local time-frequency patterns
  - Good! Capture structural characteristics
  - Bad: Only retains average occurrence frequency of patterns
- No mechanism for incremental inclusion of new information

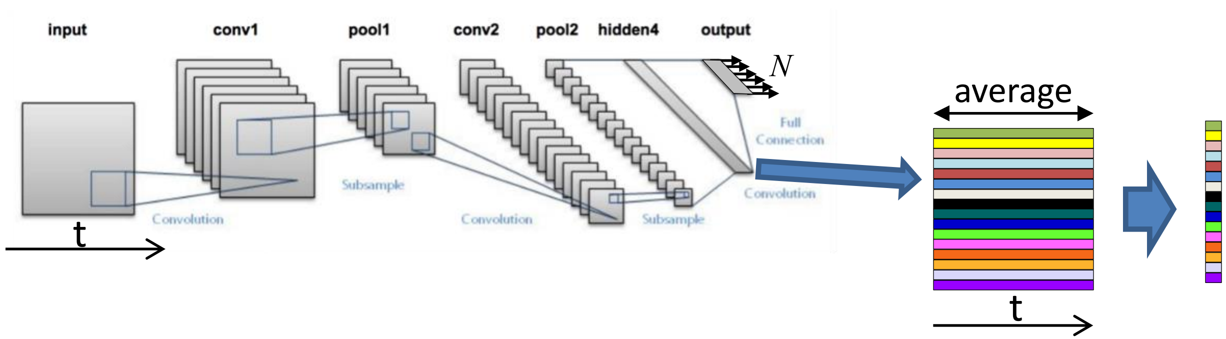
Problem from a mathematical perspective: how the CNN is trained



- The CNN is originally trained to classify between thousands of “training” speakers
  - Network has N outputs, where N = no. of training speakers
  - For any recording, only the output corresponding to that speaker must “light up”
- The final classification layer is then removed
  - The remaining network computes features that make it easy to classify between speakers

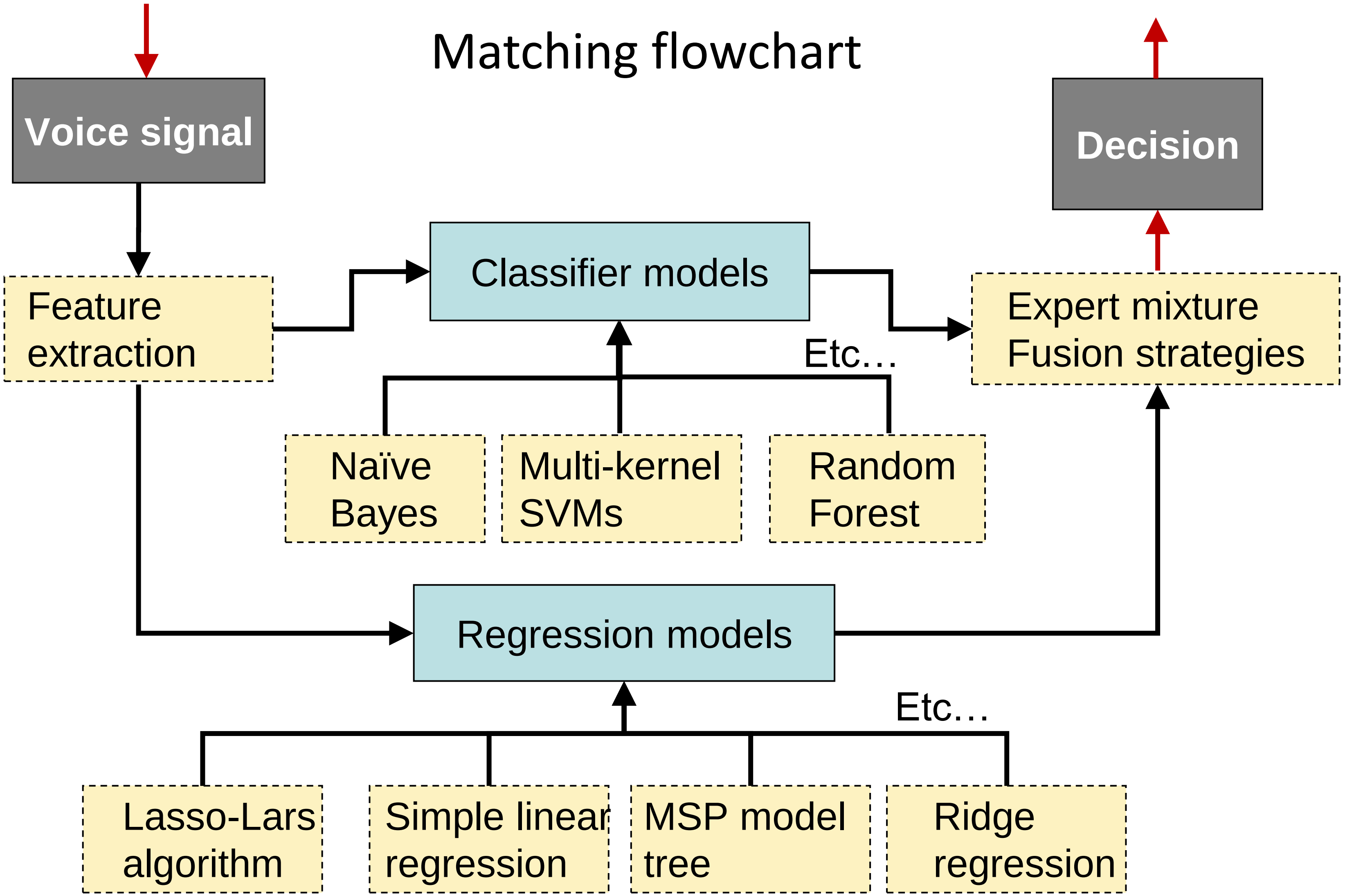


Problem from a mathematical perspective

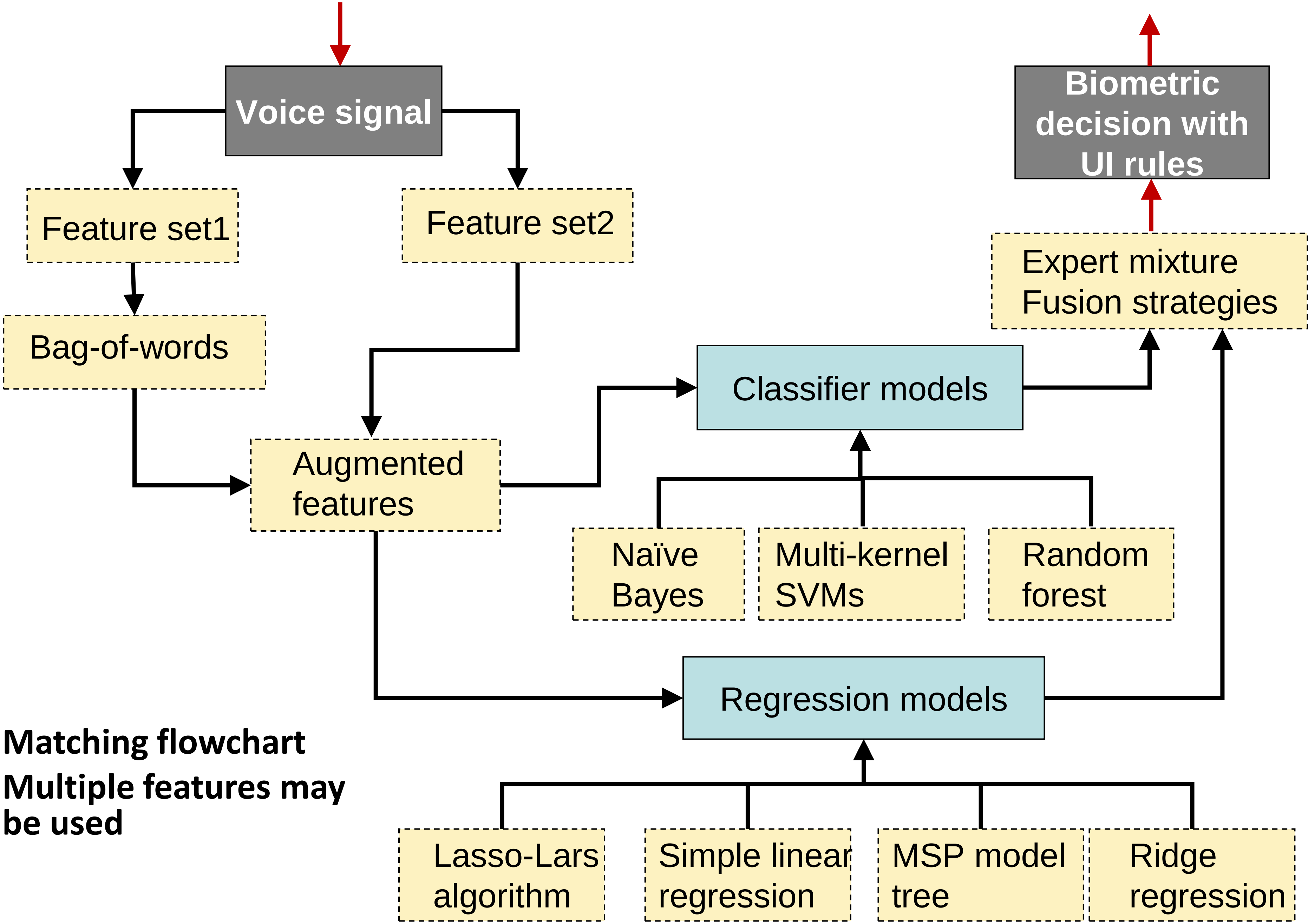


- A neural network computes class (speaker ID) log posterior: $\log P(Y|x)$
  - Or alternately, $P(Y|x)$
- Given a collection of speech segments $x_1, x_2, \dots, x_T$ the sum output computes $\sum_t \log P(Y|x_t)$
  - Or alternately, $\prod_t P(Y|x_t)$
- What is *really* required
  - $P(Y|x_1, \dots, x_T)$
- But $\prod_t P(Y|x_t) \neq P(Y|x_1, \dots, x_T)$
- The "average" operation is inappropriate to combine evidence from inputs

From features to decisions



From features to decisions



Matching flowchart

Multiple features may be used

So far…

- Feature computation (selected)
  - Spectra
  - Spectrograms
  - Mel-cepstra
  - i-vectors
  - Supervectors
  - Bag-of-words
  - NN-based features
  - Visual features
  - Videographic features

- Specific applications of audio processing (selected)
  - Audio authentication
  - Audio enhancement
  - Audio fingerprinting
  - Audio localization
  - Audio object detection
  - Audio retrieval
  - Audio summarization
  - Environmental profiling
  - Geolocation
  - Source Identification
  - Source separation
  - Speaker identification
  - Speaker profiling
  - Speaker verification
  - Speech recognition
  - Speech separation

- Key analysis techniques (selected)

Speech recognition/translation/generation

- Universally use **deep learning architectures** (Take Introduction to Deep Learning 11-785 to understand how these work)

Latest techniques:

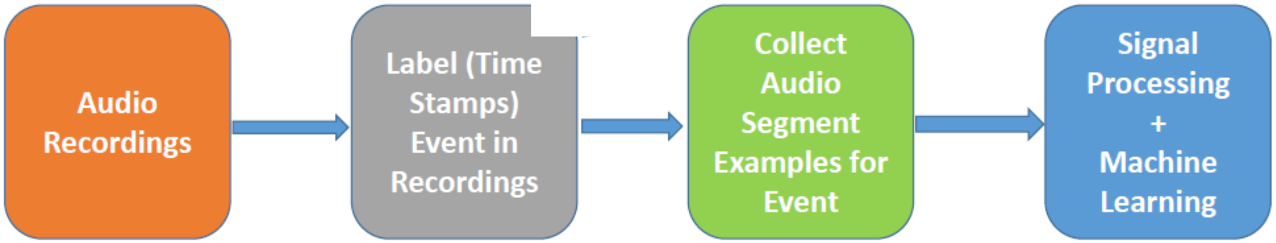
- waveform in
  - For ASR: Text out
    - Read https://arxiv.org/pdf/1609.06773.pdf
    - Combines both CTC based model and attention based model while training, with improved performance
  - For translation: Translated word sequence out
- Text in
  - For generation (synthesis): waveform out

- Best practice: use available APIs from industry for these tasks, e.g
  - Google Cloud Speech API
  - IBM Watson Speech to Text
  - IBM Watson Text to Speech
  - Microsoft Azure Bing Speech API
  - Amazon Transcribe
  - Amazon Polly

Source identification

- Examples of sounds are stored in databases
  - Matching techniques are used to compare signatures
    - Machine learning techniques are used

Audio event detection



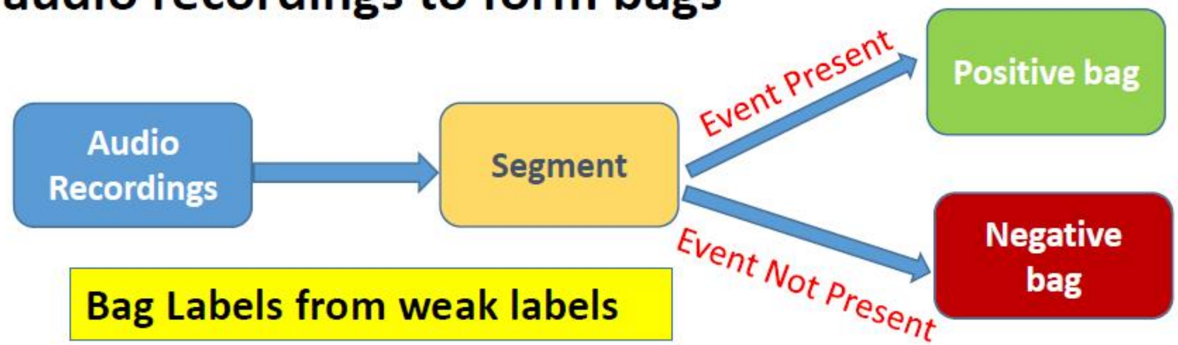
General Framework for Audio Event Detection

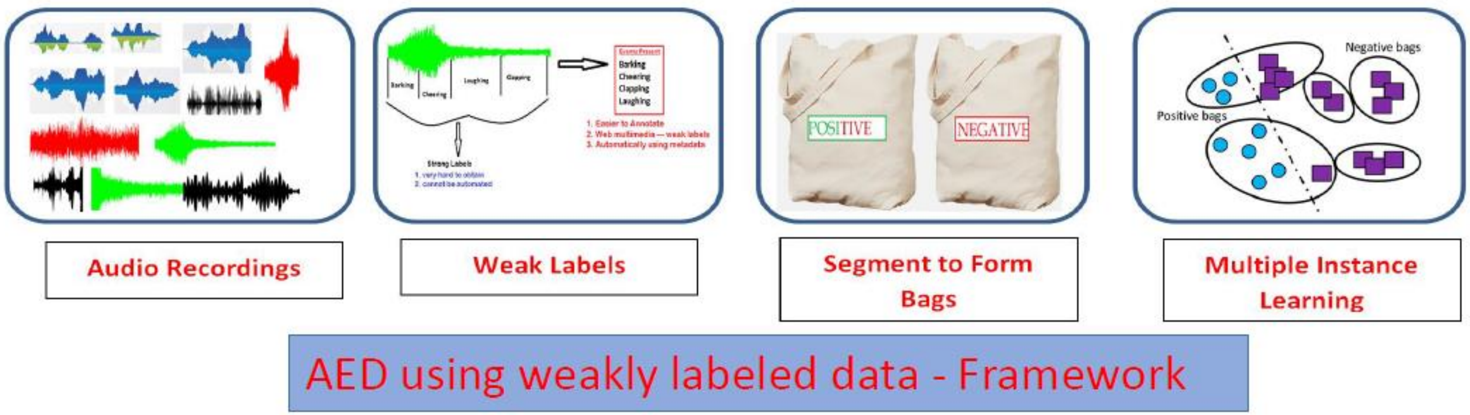
Labeling data with time stamps – Biggest Problem

- AED using strong labels

Audio event detection

- **Segment audio recordings to form bags**



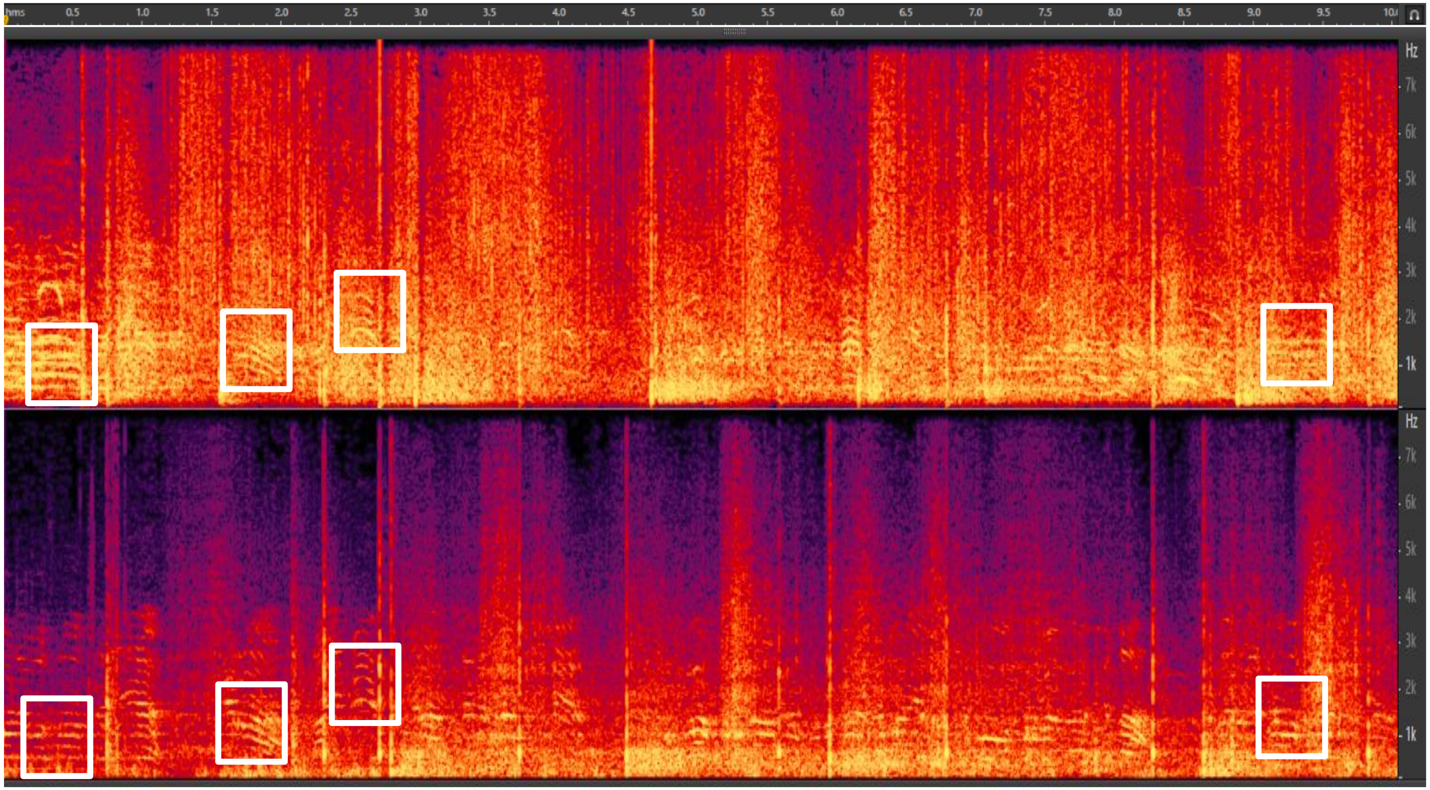


- AED using weak labels: Weakly supervised learning
- Machine learning algorithm generally used: **multiple instance learning (MIL)**

Audio fingerprinting

- "Fingerprints" (patterns) of audio are detected and used to align evidence from multiple audio recordings
  - These may be associated with video recordings

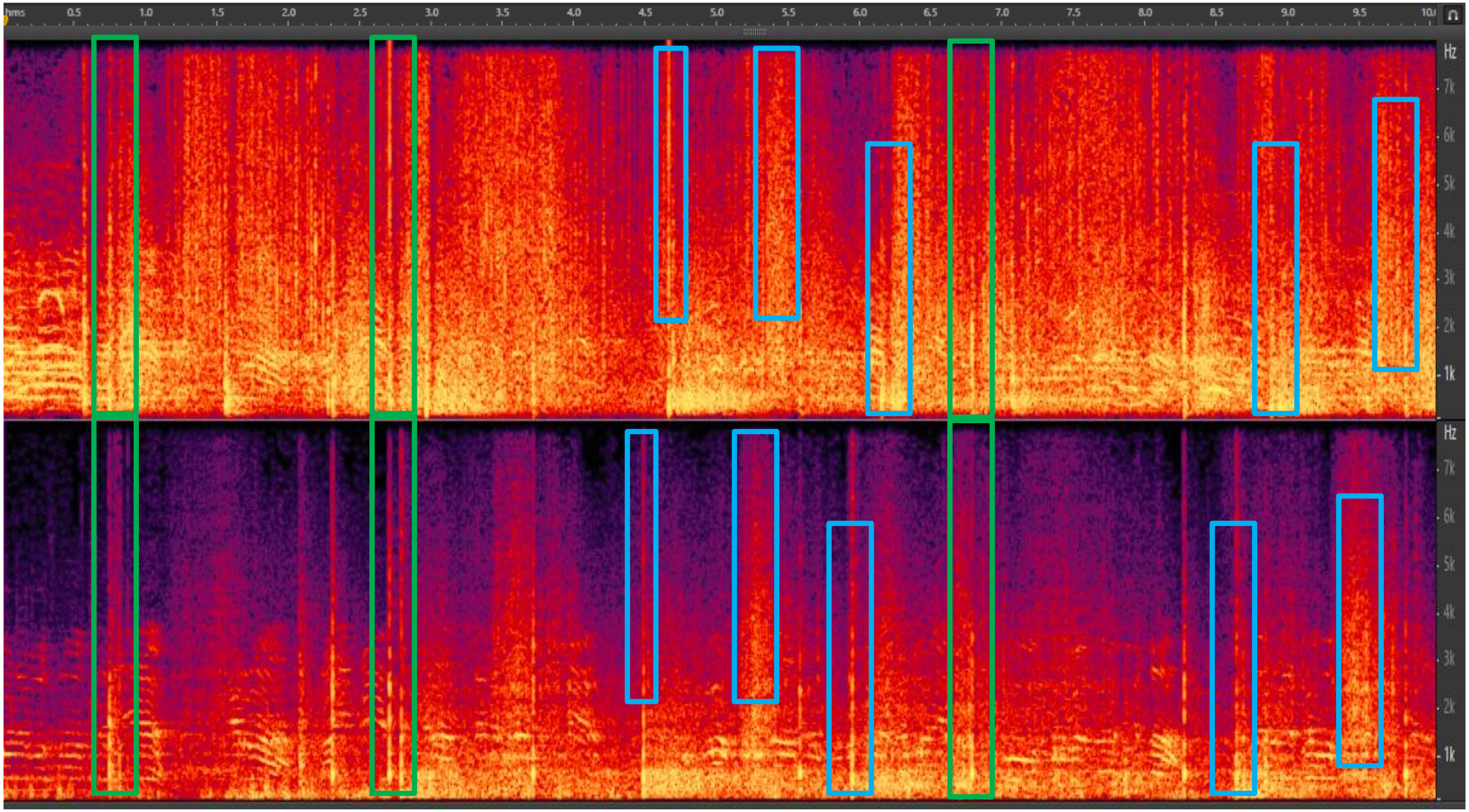
Audio fingerprinting



- Spectral peaks are aligned

Slide source: Alexander Hauptmann
Center for Human Rights Science, CMU

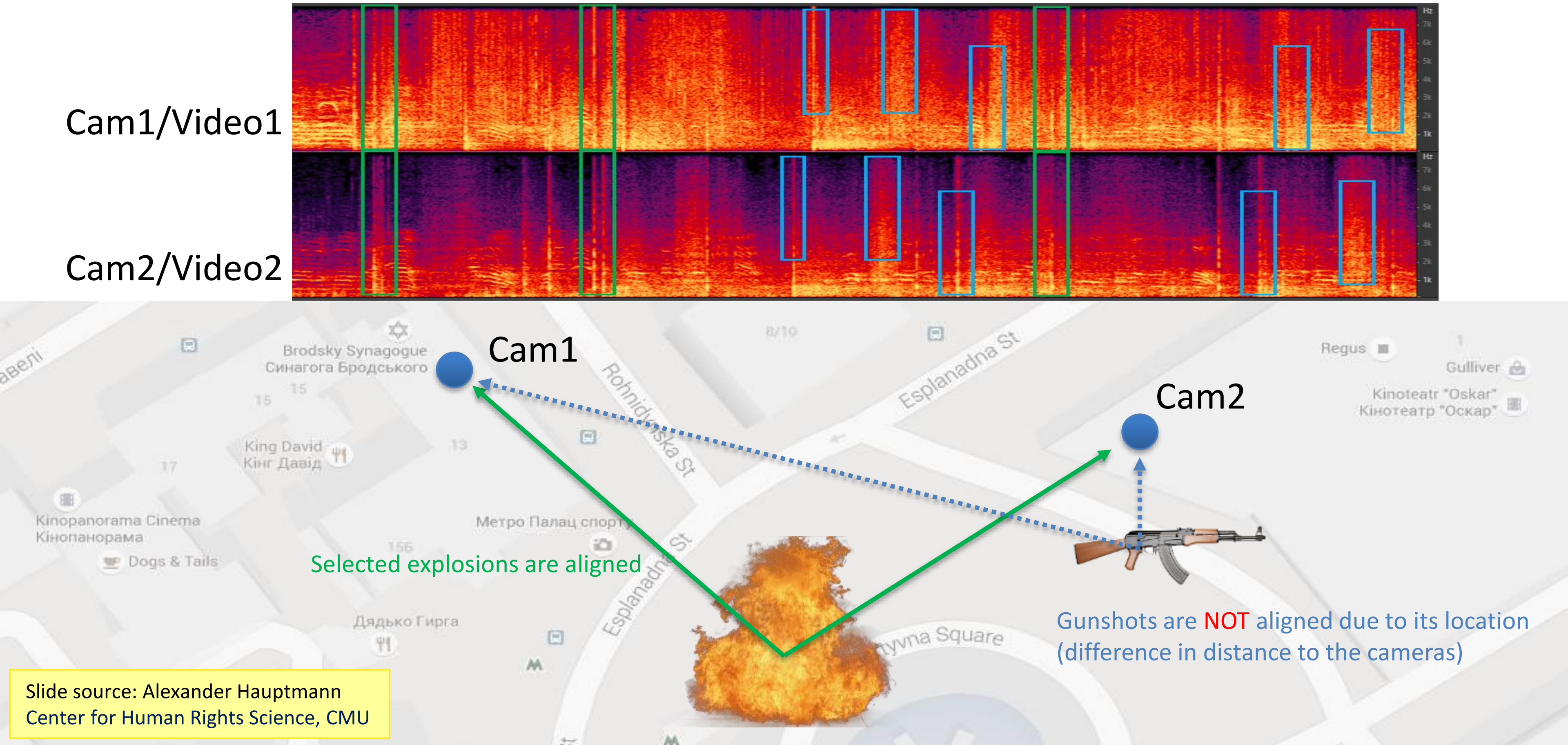
Audio fingerprinting



Slide source: Alexander Hauptmann
Center for Human Rights Science, CMU

Object positions

- Content-based automatic audio alignment
  - Match video pairs based on broadcast, sirens,explosions, gunshots
- Object positions can be determined by triangualtion



Slide source: Alexander Hauptmann
Center for Human Rights Science, CMU

In this lecture

- Digital multimedia: Recording and devices
  - Audio
  - Images
  - Video
  - Text
- Digital multimedia: Processing
  - Audio processing
  - Two generic processing techniques