Machine Learning for Signal Processing
Representing Signals: Images and Sounds
Class 4. 10 Sep 2013
Instructor: Bhiksha Raj

Representer Data
• The first and most important step in processing signals is representing them appropriately

Representing an Elephant
• It was no more than a week. Nothing much had been heard, things were not the same, and all the elephants were beginning to get restless.
• The first approached the elephant, and began to rub himself against its trunk and tail.

Representing
• Describe these images
  – Such that a listener can visualize what you are describing
• More images

Still more images
How do you describe them?
Representation

- Pixel-based descriptions are uninformative
- Feature-based descriptions are infeasible in the general case

Sounds

- Sounds are just sequences of numbers
- When plotted, they just look like blobs
  - Which leads to “natural sounds are blobs”
  - Or more precisely, “sounds are sequences of numbers that, when plotted, look like blobs”
  - Which won’t get us anywhere

Representation

- Representation is description
- But is in compact form
- Must describe the salient characteristics of the data
  - E.g. a pixel-wise description of the two images here will be completely different

- Must allow identification, comparison, storage, reconstruction.

Representing images using a “plain” image

- Most of the figure is a more-or-less uniform shade
  - Dumb approximation – a image is a block of uniform shade
  - Will be mostly right!
- How to compute the “best” description? Projection
  - Represent the images as vectors and compute the projection of the image on the “basis”

\[
BW = \text{Image}
\]
\[
W = \text{pinv}(B) \text{Image}
\]
\[
\text{PROJECTION} = BW = B \left( B^T B \right)^{-1} B^T \text{Image}
\]

Adding more bases

- Lets improve the approximation
- Images have some fast varying regions
  - Dramatic changes
  - Add a second picture that has very fast changes
    - A checkerboard where every other pixel is black and the rest are white

Image = \[ w_1 B_1 + w_2 B_2 \]
\[
W = \left[ \begin{array}{c} w_1 \\ w_2 \end{array} \right]
\]
\[
B = \left[ B_1 \ B_2 \right]
\]
\[
BW = \text{Image}
\]
\[
\text{PROJECTION} = BW = B \left( B^T B \right)^{-1} B^T \text{Image}
\]
Adding still more bases

- Regions that change with different speeds

\[ W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \end{bmatrix} B = [B_1 \ B_2 \ B_3] \]

\[ BW \approx \text{Image} \]

\[ W = \text{pinv}(B) \text{Image} \]

Getting closer at 625 bases!

What about sounds?

- Square wave equivalents of checker boards

Projecting sounds

\[ \text{Signal} = w_1B_1 + w_2B_2 + w_3B_3 \]

\[ W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \]

\[ B = [B_1 \ B_2 \ B_3] \]

\[ BW \approx \text{Signal} \]

\[ W = \text{pinv}(B)\text{Signal} \]

\[ \text{PROJECTION} = BW = (B \cdot \text{pinv}(B)) \cdot \text{Signal} \]

Representation using checkerboards

- A "standard" representation
  - Checker boards are the same regardless of the picture you’re trying to describe
    - As opposed to using “nose shape” to describe faces and “leaf colour” to describe trees.

- Any image can be specified as (for example)
  - 0.8*checkerboard(0) + 0.2*checkerboard(1) + 0.3*checkerboard(2)...

- The definition is sufficient to reconstruct the image to some degree
  - Not perfectly though

General Philosophy of Representation

- Identify a set of standard structures
  - E.g. checkerboards
  - We will call these “bases”

- Express the data as a weighted combination of these bases
  - \( X = w_1B_1 + w_2B_2 + w_3B_3 + \ldots \)

- Chose weights \( w_1, w_2, w_3, \ldots \) for the best representation of \( X \)
  - I.e. the error between \( X \) and \( \sum w_iB_i \) is minimized
  - The error is generally chosen to be \( ||X - \sum w_iB_i||^2 \)

- The weights \( w_1, w_2, \ldots \) fully specify the data
  - Since the bases are known beforehand
  - Knowing the weights is sufficient to reconstruct the data

Bases requirements

- Non-redundancy
  - Each basis must represent information not already represented by other bases
  - I.e. bases must be orthogonal
    - \( \langle B_i, B_j \rangle = 0 \) for \( i \neq j \)
  - Mathematical benefit: can compute \( w_i = \langle B_i, X \rangle \)

- Compactness
  - Must be able to represent most of \( X \) with fewest bases
  - Completeness: For D-dimensional data, need no more than D bases
Bases based representation

\[
\begin{bmatrix}
  b_{11} & b_{12} & b_{13} \\
  b_{21} & b_{22} & b_{23} \\
  b_{31} & b_{32} & b_{33}
\end{bmatrix}
\begin{bmatrix}
  w_1 \\
  w_2 \\
  w_3
\end{bmatrix}
= \begin{bmatrix}
  X_1 \\
  X_2 \\
  X_3
\end{bmatrix}
\]

- Place all bases in basis matrix \( B \)
  \[
  BW = X
  \]
  \[
  W = \text{Pinv}(B)X
  \]
- For orthogonal bases
  \[
  w_j = \frac{<B_j, X>}{\|B_j\|}
  \]

Why checkerboards are great bases

- We cannot explain one checkerboard in terms of another
  - The two are orthogonal to one another!
- This means we can determine the contributions of individual bases separately
  - Joint decomposition with multiple bases gives the same result as separate decomposition with each
    - This never holds true if one basis can explain another

Checker boards are not good bases

- Sharp edges
  - Can never be used to explain rounded curves

Sinusoids ARE good bases

- They are orthogonal
- They can represent rounded shapes nicely
  - Unfortunately, they cannot represent sharp corners

What are the frequencies of the sinusoid?

- Follow the same format as the checkerboard:
  - DC
  - The entire length of the signal is one period
  - The entire length of the signal is two periods.
  - And so on...
- The k-th sinusoid:
  - \( f(n) = \sin(2\pi kn/N) \)
  - \( N \) is the length of the signal
  - \( k \) is the number of periods in \( N \) samples
How many frequencies in all?

- A max of L/2 periods are possible
- If we try to go to (L/2 + X) periods, it ends up being identical to having [L/2 - X] periods
  - With sign inversion
- Example for L = 20
  - Red curve = sin with 9 cycles in a 20 point sequence
  - g(t) = sin(2πt/20)
  - Green curve = sin with 11 cycles in 20 points
  - h(t) = sin(2πt/20)
  - The blue lines show the actual samples obtained
- These are the only numbers stored on the computer
- This set is the same for both sinwaves.

How to compose the signal from sinusoids

\[
\begin{bmatrix}
W_1 \\
W_2 \\
W_3
\end{bmatrix} =
\begin{bmatrix}
\sin\left(\frac{2\pi}{L} k_1\right) & \cdots & \sin\left(\frac{2\pi}{L} k_N\right) \\
\cdots & \cdots & \cdots \\
\sin\left(\frac{2\pi}{L} (L-1) k_1\right) & \cdots & \sin\left(\frac{2\pi}{L} (L-1) k_N\right)
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}
\]

\[W = B W \quad BW \approx \text{Signal} \quad W = \text{pinv}(B)\text{Signal}\]

- The sines form the vectors of the projection matrix
- \(\text{pinv()}\) will do the trick as usual

Interpretation..

- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Interpretation..

- Each sinusoid's amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid
Interpretation:

- Each sinusoid’s amplitude is adjusted until it gives us the least squared error
  - The amplitude is the weight of the sinusoid
- This can be done independently for each sinusoid

Sines by themselves are not enough:

- Every sine starts at zero
  - Can never represent a signal that is non-zero in the first sample!
- Every cosine starts at 1
  - If the first sample is zero, the signal cannot be represented!

The need for phase:

- Allow the sinusoids to move!

\[ \text{signal} = w_1 \sin(2\pi kn / N + \phi_1) + w_2 \sin(2\pi kn / N + \phi_2) + \ldots. \]

- How much do the sines shift?

Determining phase:

- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another
Determining phase

- Least squares fitting: move the sinusoid left / right, and at each shift, try all amplitudes
  - Find the combination of amplitude and phase that results in the lowest squared error
- We can still do this separately for each sinusoid
  - The sinusoids are still orthogonal to one another

Complex Exponential to the rescue

\[ h[n] = \exp(j \times \text{freq} \times n) = \cos(\text{freq} \times n) + j \sin(\text{freq} \times n) \]
\[ j = \sqrt{-1} \]

\[ \exp(j \times \text{freq} \times n + \phi) = \exp(j \times \text{freq} \times n) \exp(j \times \phi) = \cos(\text{freq} \times n + \phi) + j \sin(\text{freq} \times n + \phi) \]

- The cosine is the real part of a complex exponential
- The sine is the imaginary part
- A phase term for the sinusoid becomes a multiplicative term for the complex exponential!!

Complex exponentials are well behaved

- Like sinusoids, a complex exponential of one frequency can never explain one of another
  - They are orthogonal
- They represent smooth transitions
- Bonus: They are complex
  - Can even model complex data!
- They can also model real data
  - \( \exp(j \times x) + \exp(-j \times x) \) is real
    - \( \cos(x) + j \sin(x) + \cos(x) - j \sin(x) = 2\cos(x) \)
- More importantly
  - \( \exp(j \times L/2 \times x) + \exp(j \times L/2 \times x) \) is real
    - The complex exponentials with frequencies equally spaced from \( L/2 \) are complex conjugates

The problem with phase

- This can no longer be expressed as a simple linear algebraic equation
  - The “basis matrix” depends on the unknown phase
    - i.e. it’s a component of the basis itself that must be estimated!
- Linear algebraic notation can only be used if the bases are fully known
  - We can only (pseudo) invert a known matrix

Complex exponentials are well behaved

\[ \exp(j2\pi \times (L/2 - x) n/L) + \exp(j2\pi \times (L/2 + x) n/L) \]

- The complex exponentials with frequencies equally spaced from \( L/2 \) are complex conjugates
  - “Frequency = k numerators k periods in L samples
    - \( \exp(j2\pi \times (L/2 - x) n/L) + \text{conjugate}(n) \exp(j2\pi \times (L/2 + x) n/L) \)
  - Is also real
    - If the two exponentials are multiplied by numbers that are conjugates of one another the result is real
Complex Exponential bases

\[
\begin{bmatrix}
\omega_0 & \omega_{1/2}^1 & \omega_{1/2}^0 & \vdots & \omega_{N/2}^0 \\
\omega_{1/2}^0 & \omega_{1/2}^1 & \omega_{1/2}^2 & \cdots & \omega_{1/2}^N \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\omega_{N/2}^0 & \omega_{N/2}^1 & \omega_{N/2}^2 & \cdots & \omega_{N/2}^N
\end{bmatrix}
\]

- Explain the data using L complex exponential bases
- The weights given to the \([L/2 + k]th\) and \(([L-2]/2 + k)\)th basis should be complex conjugates, to make the result real
  - Because we are dealing with real data
- Fortunately, a least squares fit will give us identical weights to both bases automatically; there is no need to impose the constraint externally

Shorthand Notation

\[
w_L^{k,n} = \frac{1}{N} \exp(j2\pi k n / L) = \frac{1}{\sqrt{L}} (\cos(2\pi k n / L) + j\sin(2\pi k n / L))
\]

\[
\begin{bmatrix}
w_L^{k,0} & w_L^{k,1} & w_L^{k,2} & \cdots & w_L^{k,L-1} \\
w_L^{k+1,0} & w_L^{k+1,1} & w_L^{k+1,2} & \cdots & w_L^{k+1,L-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
w_L^{k,L-1} & w_L^{k,L-2} & w_L^{k,L-3} & \cdots & w_L^{k,0}
\end{bmatrix}
= \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_L \end{bmatrix}
= \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[L-1] \end{bmatrix}
\]

- Note that \(S_{L/2+k} = \text{conjugate}(S_{L/2+k})\) for real \(x\)

A quick detour

- Real Orthonormal matrix:
  - \(XX^T = X^TX = I\)
  - But only if all entries are real
  - The inverse of \(X\) is its own transpose
- Definition: Hermitian
  - \(X^\dagger = \text{Complex conjugate of } X^T\)
  - Conjugate of a number \(a + jb = a - jb\)
  - Conjugate of \(\exp(\alpha) = \exp(\alpha)\)
- Complex Orthonormal matrix
  - \(XX^\dagger = X^\dagger X = I\)
  - The inverse of a complex orthonormal matrix is its own Hermitian

\(W^{-1} = WH\)

\[
W = \begin{bmatrix}
w_L^{k,0} & w_L^{k,1} & w_L^{k,2} & \cdots & w_L^{k,L-1} \\
w_L^{k+1,0} & w_L^{k+1,1} & w_L^{k+1,2} & \cdots & w_L^{k+1,L-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
w_L^{k,L-1} & w_L^{k,L-2} & w_L^{k,L-3} & \cdots & w_L^{k,0}
\end{bmatrix}
= \begin{bmatrix} W_L^{k,0} \\ W_L^{k,1} \\ \vdots \\ W_L^{k,L-1} \end{bmatrix}
= \begin{bmatrix} W_L^{k,0} & W_L^{-k,0} & W_L^{-k,1} & \cdots & W_L^{-k,L-1} \\
W_L^{-k,0} & W_L^{k,0} & W_L^{k,1} & \cdots & W_L^{k,L-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
W_L^{-k,L-1} & W_L^{k,L-2} & W_L^{k,L-3} & \cdots & W_L^{k,0}
\end{bmatrix}
= \frac{1}{\sqrt{L}} \exp(j2\pi k n / L)
\]

- The complex exponential basis is orthogonal
- Its inverse is its own Hermitian

Doing it in matrix form

\[
\begin{bmatrix}
w_L^{k,0} & w_L^{k,1} & w_L^{k,2} & \cdots & w_L^{k,L-1} \\
w_L^{k+1,0} & w_L^{k+1,1} & w_L^{k+1,2} & \cdots & w_L^{k+1,L-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
w_L^{k,L-1} & w_L^{k,L-2} & w_L^{k,L-3} & \cdots & w_L^{k,0}
\end{bmatrix}
= \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_L \end{bmatrix}
= \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[L-1] \end{bmatrix}
\]

- Because \(W^{-1} = WH\)
**The Discrete Fourier Transform**

\[
\begin{bmatrix}
S_0 \\
S_1 \\
\vdots \\
S_{L-1}
\end{bmatrix} =
\begin{bmatrix}
W_L^0 & W_L^{-1} & \cdots & W_L^{-(L-1)} \\
W_L^{-1} & W_L^0 & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
W_L^{-(L-1)} & \vdots & \cdots & W_L^0
\end{bmatrix}
\begin{bmatrix}
s[0] \\
s[1] \\
\vdots \\
s[L-1]
\end{bmatrix}
\]

- The matrix to the right is called the “Fourier Matrix”
- The weights \(S_0, S_1\ldots\text{Etc.}\) are called the Fourier transform

**The Inverse Discrete Fourier Transform**

\[
\begin{bmatrix}
S_0 \\
S_1 \\
\vdots \\
S_{L-1}
\end{bmatrix} =
\begin{bmatrix}
W_L^0 & W_L^{-1} & \cdots & W_L^{-(L-1)} \\
W_L^{-1} & W_L^0 & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
W_L^{-(L-1)} & \vdots & \cdots & W_L^0
\end{bmatrix}^{-1}
\begin{bmatrix}
s[0] \\
s[1] \\
\vdots \\
s[L-1]
\end{bmatrix}
\]

- The matrix to the left is the inverse Fourier matrix
- Multiplying the Fourier transform by this matrix gives us the signal right back from its Fourier transform

**The Fourier Matrix**

- Left panel: The real part of the Fourier matrix
  - For a 32-point signal
- Right panel: The imaginary part of the Fourier matrix

**The FAST Fourier Transform**

- The outcome of the transformation with the Fourier matrix is the DISCRETE FOURIER TRANSFORM (DFT)
- The FAST Fourier transform is an algorithm that takes advantage of the symmetry of the matrix to perform the matrix multiplication really fast
- The FFT computes the DFT

**Images**

- The complex exponential is two dimensional
  - Has a separate X frequency and Y frequency
    - Would be true even for checker boards!
  - The 2-D complex exponential must be unravelled to form one component of the Fourier matrix
    - For a KxL image, we’d have K*L bases in the matrix

**Typical Image Bases**

- Only real components of bases shown
DFT: Properties

- The DFT coefficients are complex
  - Have both a magnitude and a phase
    \[ S_k = S_k \exp(-j\omega_k) \]
- Simple linear algebra tells us that
  - \[ \text{DFT}(A + B) = \text{DFT}(A) + \text{DFT}(B) \]
  - The DFT of the sum of two signals is the DFT of their sum
- A horribly common approximation in sound processing
  - \[ \text{Magnitude}(\text{DFT}(A+B)) = \text{Magnitude}(\text{DFT}(A)) + \text{Magnitude}(\text{DFT}(B)) \]
  - Utterly wrong
  - Absurdly useful

9/10/2013  11-155/10-767  S3

The Discrete Cosine Transform

- Compose a symmetric signal or image
  - Images would be symmetric in two dimensions
- Compute the Fourier transform
  - Since the FT is symmetric, sufficient to store only half the coefficients (quarter for an image)
  - Or as many coefficients as were originally in the signal / image

9/10/2013  11-155/10-767  S7

Representing images

- Most common coding is the DCT
- JPEG: Each 8x8 element of the picture is converted using a DCT
- The DCT coefficients are quantized and stored
  - Degree of quantization = degree of compression
- Also used to represent textures etc for pattern recognition and other forms of analysis

9/10/2013  11-155/10-767  S8

Symmetric signals

- Contributions from points equidistant from L/2 combine to cancel out imaginary terms

9/10/2013  11-155/10-767  S8

DCT

\[
\begin{align*}
&\cos(2\pi(0.5)(0/2L)) & & \cos(2\pi(0.5)(1/2L)) \\
&\cos(2\pi(0.5)(2/2L)) & & \cos(2\pi(0.5)(3/2L)) \\
&\vdots & & \vdots \\
&\cos(2\pi(0.5)(L-1/2L)) & & \cos(2\pi(0.5)(L-1/2L)) \\
\end{align*}
\]

- Not necessary to compute a 2xL sized FFT
  - Enough to compute an L-sized cosine transform
  - Taking advantage of the symmetry of the problem
- This is the Discrete Cosine Transform

9/10/2013  11-155/10-767  S8

Some tricks to computing Fourier transforms

- Direct computation of the Fourier transform can result in poor representations
  - Boundary effects can cause error
    - Solution: Windowing
  - The size of the signal can introduce inefficiency
    - Solution: Zero padding

9/10/2013  11-155/10-767  S8
What does the DFT represent

- The IDFT can be written formulaically as above
- There is no restriction on computing the formula for \( n < 0 \) or \( n > L - 1 \)
  - It's just a formula
  - But computing these terms behind 0 or beyond \( L - 1 \) tells us what the signal composed by the DFT looks like outside our narrow window

The discrete Fourier transform

- The discrete Fourier transform of the above signal actually computes the properties of the periodic signal shown below
  - Which extends from \(-\infty\) to \(+\infty\)
  - The period of this signal is 32 samples in this example

Windowing

- The DFT of one period of the sinusoid shown in the figure computes the spectrum of the entire sinusoid from \(-\infty\) to \(+\infty\)
The DFT of one period of the sinusoid shown in the figure computes the spectrum of the entire sinusoid from -infinity to +infinity.
The DFT of a real sinusoid has only one non-zero frequency.
The second peak in the figure is the "reflection" around \( L/2 \) (for real signals).

The DFT of any sequence computes the spectrum for an infinite repetition of that sequence.
The DFT of a partial segment of a sinusoid computes the spectrum of an infinite repetition of that segment, and not of the entire sinusoid.

This will not give us the DFT of the sinusoid itself.

The difference occurs due to two reasons:
The transform cannot know what the signal actually looks like outside the observed window.
Windowing

- The difference occurs due to two reasons:
  - The transform cannot know what the signal actually looks like outside the observed window
  - The implicit repetition of the observed signal introduces large discontinuities at the points of repetition
    - These are not part of the underlying signal
    - We only want to characterize the underlying signal
  - The discontinuity is an irrelevant detail

10 Sep 2013
11-158/10-797
72

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 attempts to do the following:
    - Keep the windowed signal similar to the original in the central regions

10 Sep 2013
11-158/10-797
72

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 attempts to do the following:
    - Keep the windowed signal similar to the original in the central regions

10 Sep 2013
11-158/10-797
72

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 attempts to do the following:
    - Keep the windowed signal similar to the original in the central regions

10 Sep 2013
11-158/10-797
72
**Window functions**

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

**Zero Padding**

- We can pad zeros to the end of a signal to make it a desired length
  - Useful if the FFT (or any other algorithm we use) requires signals of a specified length
  - E.g., Radix 2 FFTs require signals of length $2^l$, i.e., some power of 2. We must zero pad the signal to increase its length to the appropriate number
- The consequence of zero padding is to change the periodic signal

**Magnitude spectrum**

- The DFT of the zero padded signal is essentially the same as the DFT of the unpadded signal, with additional spectral samples inserted in between
  - It does not contain any additional information over the original DFT
  - It also does not contain less information
Zero Padding

- The DFT of the zero padded signal is essentially the same as the DFT of the unpadded signal, with additional spectral samples inserted in between.
  - It does not contain any additional information over the original DFT.
  - It also does not contain less information.

The Fourier Transform and Perception:

- Sound

  - The Fourier transform represents the signal analogously to a bank of tuning forks.
  - Our ear has a bank of tuning forks.
  - The output of the Fourier transform is perceptually very meaningful.

The Fourier Transform and Perception:

- Sound parameterization

  - The signal is processed in segments of 25-64 ms
    - Because the properties of audio signals change quickly
    - They are “stationary” only very briefly.

Zero padding a speech signal

- 128 samples from a speech signal sampled at 16000 Hz

- The first 65 points of a 128 point DFT. Plot shows log of the magnitude spectrum.

- The first 513 points of a 1024 point DFT. Plot shows log of the magnitude spectrum.
Sound parameterization

- The signal is processed in segments of 25-64 ms
  - Because the properties of audio signals change quickly
  - They are “stationary” only very briefly
- Adjacent segments overlap by 15-48 ms
Sound parameterization

Segments shift every 10-16 milliseconds. Each segment is typically 25-64 milliseconds wide. Audio signals typically do not change significantly within the short time interval.

Sound parameterization

Each segment is windowed and a DFT is computed from it.

Computing a Spectrogram

Compute Fourier Spectra of segments of audio and stack them side-by-side.
Computing a Spectrogram

The result of parameterization

- Each column here represents the FT of a single segment of signal 64ms wide.
  - Adjacent segments overlap by 48 ms.
- DFT details
  - 1024 points (16000 samples a second).
  - 2048 point DFT – 1024 points of zero padding.
  - Only 1025 points of each DFT are shown.
  - The rest are “reflections”
- The value shown is actually the magnitude of the complex spectral values.
  - Most of our analysis / operations are performed on the magnitude.

Magnitude and phase

- All the operations (e.g. the examples shown in the previous class) are performed on the magnitude.
- The phase of the complex spectrum is needed to invert a DFT to a signal.
  - Where does that come from?
- Deriving phase is a serious, not-quite solved problem.

Phase

- Common tricks: Obtain the phase from the original signal
  - $\phi$ = DFT(signal)
  - Phase1 = phase($\phi$)
    - Each term is of the form real + i*imag
      - For each element, compute $\text{atan2}(\text{imag}, \text{real})$
  - Magnitude = magnitude($\phi$)
    - For each element compute $\sqrt{\text{real}^2 + \text{imag}^2}$
  - ProcessedSpectrum = ProcessMagnitude
  - New SFT = ProcessedSpectrum * exp($i$ * Phase1)
  - Recover signal from SFT

- Some other tricks:
  - Compute the FT of a different signal of the same length.
  - Use the phase from that signal.
Returning to the speech signal

- For each complex spectral vector, compute a signal from the inverse DFT
  - Make sure to have the complete FT (including the reflected portion)
- If need be window the retrieved signal
- Overlap signals from adjacent vectors in exactly the same manner as during analysis
  - E.g., if a 46ms (768 sample) overlap was used during analysis, overlap adjacent segments by 768 samples

Additional tricks

- The basic representation is the magnitude spectrum
- Often it is transformed to a log spectrum
  - By computing the log of each entry in the spectrum matrix
  - After processing, the entry is exponentiated to get back the magnitude spectrum
  - To which phase may be factored in to get a signal
- The log spectrum may be "compressed" by a dimensionality reducing matrix
  - Usually a DCT matrix

What about images?

- DCT of small segments
  - Ball
  - Each image becomes a matrix of DCT vectors
- DCT of the image
- Pyramid representations
- haar transform [checkboard]
- Various wavelet representations
  - Gabor wavelets
- Or data-driven representations

Downsampling-based representations

- Downsampling an example
  - Trying to reduce size by factor of 4 each time
    - Select every alternate sample row-wise and column-wise
  - What exactly did we capture?
    - Clue: Results are horrible

Downsampling-based representations

- Nasty aliasing effects!

The Gaussian Kernel

- A two-dimensional image of a Gaussian
- Characterized by
  - Center (mean)
  - Standard deviation σ (assumed same in both directions)
  - I.e. spherical Gaussian
- The image can be represented by a vector
**The Gaussian Kernel matrix**

\[
G = \begin{bmatrix}
g_{11} & g_{12} & \cdots & g_{1N} 
g_{21} & g_{22} & \cdots & g_{2N} 
\vdots & \vdots & \ddots & \vdots 
g_{N1} & g_{N2} & \cdots & g_{NN}
\end{bmatrix}
\]

- Each column is one Gaussian
  - Representing a Gaussian centered at one of the pixels in the image
- As many columns as pixels
  - Also as many rows as pixels

**Downsampling-based representations**

\[
G \times X \Rightarrow G_{i}X_{i}
\]

- Transform with Gaussian kernel matrix
- Then downsample

**The Gaussian Pyramid**

\[
L_{i} = G_{x}
\]

- Successive smoothing and scaling
- The entire collection of images is the Gaussian pyramid

**Laplacians**

\[
X - GX \Rightarrow X_{i} - G_{i}X_{i}
\]

**Laplacian Pyramid**

\[
L_{1} \supset L_{2} \supset \ldots \supset L_{n}
\]
Remember.

- The Gaussian is an anti-aliasing filter
- The Gaussian pyramid is the *low-pass filtered* version of the image
- The Laplacian pyramid is the *high-pass filtered* version of the image

The Gaussian/Laplacian Decomposition

- Each low-pass filtered image is downsampled
- The process is recursively performed

The discrete wavelet transform

- Very similar in structure
- But the bases at each scale are orthogonal to bases at other scales
  - As opposed to a Gaussian kernel matrix

Haar Wavelets

- We have already encountered Haar wavelets

Other characterizations

- Content-based characterizations
  - E.g., Hough transform
    - Captures linear arrangements of pixels
  - Radon transform
  - SIFT features
  - Etc.

- Will revisit in homework.