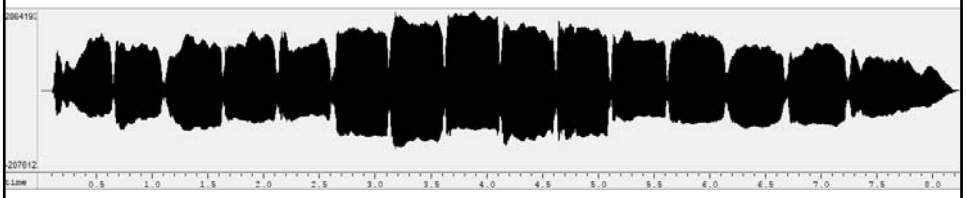
# Speech Processing
# Time-Frequency Representations and dimension reduction

Professor Marie Roch
San Diego State University

SAN DIEGO STATE UNIVERSITY

# What's going on here?



SAN DIEGO STATE UNIVERSITY

2

# Alternative representations

- Frequency domain: spectrum
- Frequency over time:  spectrogram



# Frequency domain

- Result of frequency analysis transform
- Discrete Fourier transform (DFT)

$$X(e^{j\omega_0}) = \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega_0 k}$$

  - x[k] k$^{th}$ time domain sample
  - $X(e^{j\omega_0})$ – content at $\omega_0 = 2\pi freq$, e.g. 30 Hz $\omega_0 = 2\pi 30$
  - $j = \sqrt{-1}$

# DFT

- Euler's formula lets us relate a complex exponential to trigonometric functions

$$Ae^{j\omega n} = A\cos(\omega n) + jA\sin(\omega n)$$

- So for DFT $X(e^{j\omega_0}) = \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega_0 k}$
  - $k$ moves along the unit circle
  - $\omega_0$ controls step speed
  - $\omega_0 = 2\pi * freq$



# DFT output

- $X(e^{j\omega_0})$ is a complex value
- The magnitude of the complex value is an indication of pressure strength at a given frequency.
- Squared value is related to intensity

## DFT Output

$$X(e^{j\omega_0})\overline{X(e^{j\omega_0})} = (a+bj)(a-bj)$$
$$= a^2 - abj + abj + b^2$$
$$= a^2 + b^2$$

where $\bar{X}$, or sometimes $X^*$,
denotes the complex conjugate



## DFT Output

- Convert to dB rel:

$$10\log_{10}\left(X(e^{j\omega_0})\overline{X(e^{j\omega_0})}\right)$$

or $20\log_{10}\left(\left|X(e^{j\omega_0})\right|\right)$ where $|\ |$ is magntiude

# Notes of the Fourier Transform

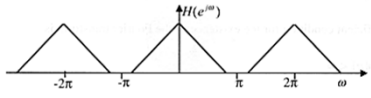
- The Fourier transform for aperiodic signals exists when: $\sum_{n=-\infty}^{\infty} |x[n]| < \infty$
- It is sometimes convenient to normalize the frequency range to [-π, π]
- To move from a normalized [-π, π] frequency axis to Hz:

$$f(\omega) = \frac{\omega}{2\pi} \times SamplingRate, \ \omega \in [0, 2\pi]$$



# Notes on the DFT

- The DFT of real signals is periodic with period 2π (normalized frequency axis).

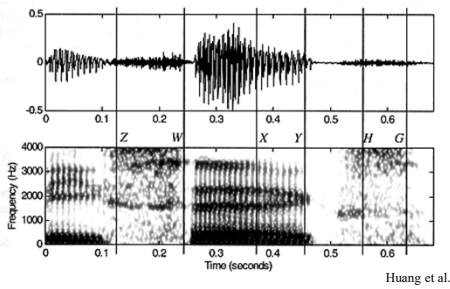



Huang et al. p 209

- An alternate form exists for analyzing periodic signals.

# Short-Time Fourier Analysis

Analyzing the whole signal doesn't make sense, not all regions are similar:




Huang et al. p 277

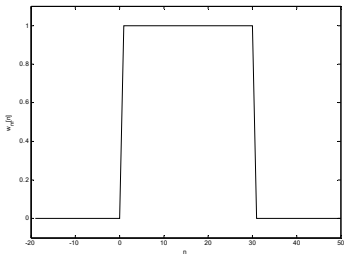


# Discrete Frequency & Short-Time Fourier Transforms

- Computational methods require a discrete frequency domain.
- Speech signals change over time and we want to analyze a "static" portion of the signal.

# Short-Time Fourier Analysis

- Segment the signal into small *frames* and then perform an analysis on each frame.
- Let $w_m[n]$ be a rectangular window for the m'th frame.





# Short Time Discrete Fourier Transform

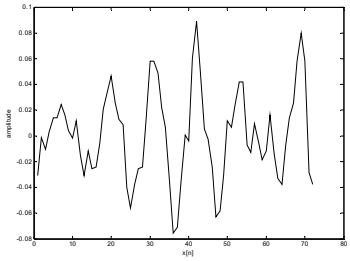
- Assume $x_N[n] = x_N[n+N]$
  - $x_N[n]$ periodic with period N.
- We define the DFT as

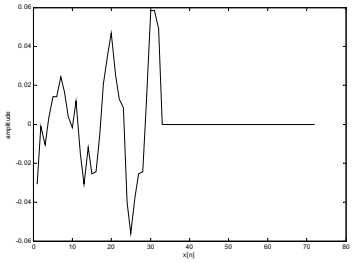
$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi nk/N} \text{ where } 0 \le k < N$$

and will refer to this as Fourier analysis

# Extracting the short-time signal

- x[n]
- $x_m[n]=x[n]w_m[n]$



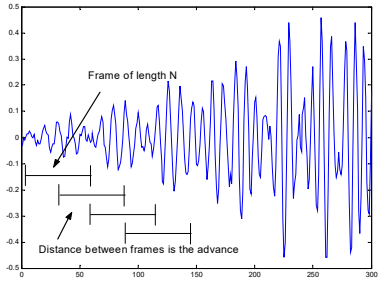




Window function extracts $m^{th}$ frame



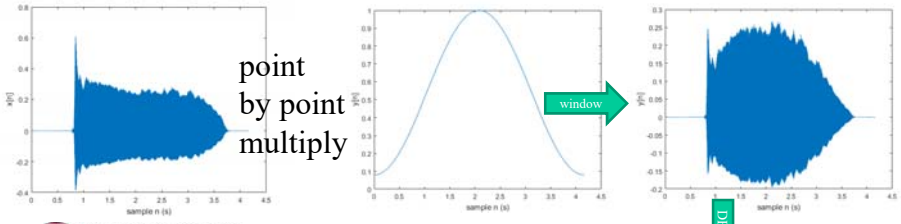
# Short-time Fourier Analysis

- By shifting the region where the window is non-zero we can control what portion of the signal remains for each frame.
- We must choose:
  - Frame length
  - Frame advance



# Window functions

- Rectangular windows causes energy to "leak" from peaks in the spectrum.
- Reduce effect by tapering edges of window function.




better spectral estimate



# Short-Time Fourier Analysis

- Fourier analysis of the frame m:

$$X_m(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]w_m[n]e^{-j\omega n}$$

- Analysis with discrete frequencies:

$$X_m[k] = \sum_{n=0}^{N-1} x[n]w_m[n]e^{-j2\pi nk/N} \text{ where } 0 \le k < N$$

frequency associated with bin $k$ in $X_m[k]$ is $\frac{k}{N}Fs$

# Python DFT

```
# spyder:
# To avoid tiny inline plots...
# Tools/Preferences/Ipython Graphics/Backend automatic
# (kernal must be restarted)
import numpy as np
import matplotlib.pyplot as plt

frame_s = 0.020 # frame length
Fs = 8000  # sample rate

frame_N = int(np.round(Fs * frame_s))  # Num samples
tidx = np.arange(frame_N) / Fs  # time in s

# sinusoid components for signal
omega = np.array([1000, 2000, 2500])  # frequencies at Hz
amplitudes = np.array([1, .5, .25])
```

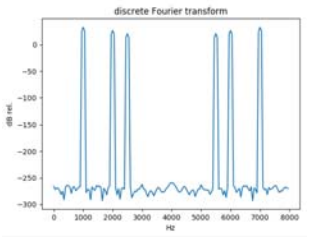


# Python DFT

```
# build a sine wave
x = np.zeros([frame_N])
for idx in range(len(omega)):
    x = x + (amplitudes[idx] * np.sin(2 * np.pi *
   omega[idx] * tidx))

# plot the signal
plt.figure()
plt.plot(tidx, x)
plt.title('Signal')
plt.xlabel('time (s)')
plt.ylabel('Pressure (counts)')
# Use plt.show() if not running ipython
```

# Python DFT

```
# Compute discrete Fourier transform
bins_Hz = np.arange(frame_N)/frame_N*Fs
window = signal.get_window("hamming", frame_N)
windowed_x = x * window
X = np.fft.fft(windowed_x)

# spectrum X is complex, convert to dB
magX = np.abs(X)
mag_dB = 20 * np.log10(magX)

plt.figure()
plt.plot(bins_Hz, mag_dB)
plt.title("discrete Fourier transform")
plt.xlabel("Hz")
plt.ylabel('dB rel.')
```




SAN DIEGO STATE UNIVERSITY

21

# Typical framing parameters

- Window: Hamming.
- Frame/Window length: 20-30 ms.
- Frame advance: 10-20 ms.

SAN DIEGO STATE UNIVERSITY

22

# Spectrograms

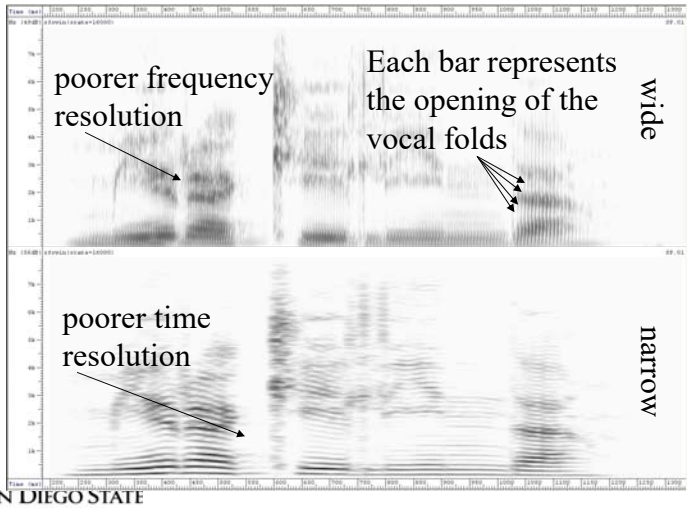
- Spectrograms are a series of short-time DFTs on windowed segments of a signal
- There are two types of spectrograms:
  - Narrow-band spectrogram
  - Wide-band spectrogram
- The type depends upon the window length.



# Window length

- Short ≈ 10 ms or less
  - Better time analysis
  - Frequency bands are widely spaced.
  - F0 can be estimated by determining time between vertical striations
- Long ≈ 20 ms or more
  - Poor time analysis
  - Frequency bands narrowly spaced.

# Narrow vs. Wide-band Spectrogram






# Feature vectors

- A set of feature measurements in a $D$ dimensional space

$$\vec{x} = \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_{D-1} \\ f_D \end{bmatrix}$$

- More is better?

# Feature vectors

- What if $f_i$ and $f_j$ vary in predictable ways?

  Then they are not independent: $P(f_i, f_j) \neq P(f_i)P(f_j)$
  and their covariance is non-zero

$$\text{cov}(f_i, f_j) = E[(f_i - \mu_{f_i})(f_j - \mu_{f_j})] \neq 0$$

- Strong dependence is not useful

What is the $\text{cov}(f_i, f_i)$?



# Covariance of features

$$\text{cov}(f_i, f_j) = E[(f_i - \mu_{f_i})(f_j - \mu_{f_j})]$$

$$= E[f_i f_j - f_i \mu_{f_j} - \mu_{f_j} f_j + \mu_{f_i} \mu_{f_j}] \qquad E[\cdot] \text{ is a linear operator}$$

$$= E[f_i f_j] - E[f_i \mu_{f_j}] - E[\mu_{f_j} f_j] + E[\mu_{f_i} \mu_{f_j}]$$

$$= E[f_i f_j] - E[f_i]\mu_{f_j} - \mu_{f_j} E[f_j] + \mu_{f_i} \mu_{f_j}$$

$$= E[f_i f_j] - \mu_{f_i} \mu_{f_j} - \mu_{f_j} \mu_{f_j} + \mu_{f_i} \mu_{f_j}$$

$$= E[f_i f_j] - \mu_{f_i} \mu_{f_j} = \int_{f_i} \int_{f_j} f_i f_j P(f_i, f_j) df_i df_j - \mu_{f_i} \mu_{f_j}$$

Sample covariance: $\widehat{cov}(f_i, f_j) = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \mu_{f_i})(f_j - \mu_{f_j})$

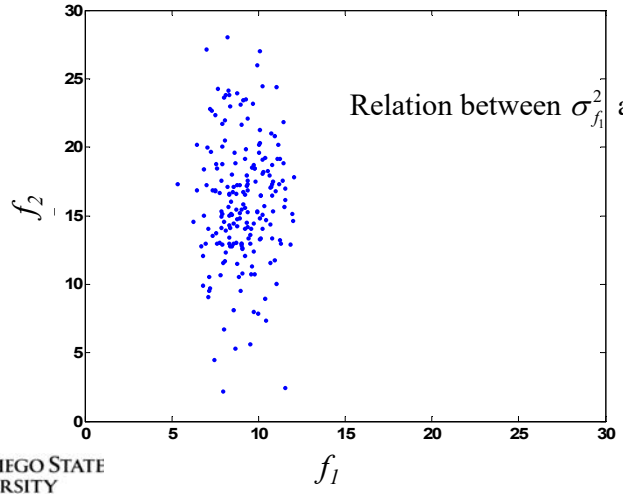
# Covariance of features

- Ideally, we would like features to be independent.
- Sometimes useful to build a variance-covariance matrix $\sum$

$$\Sigma = \begin{bmatrix} \mathrm{cov}(f_1, f_1) & \mathrm{cov}(f_2, f_1) & \cdots & \mathrm{cov}(f_{D-1}, f_1) & \mathrm{cov}(f_D, f_1) \\ \mathrm{cov}(f_1, f_2) & \mathrm{cov}(f_2, f_2) & & \mathrm{cov}(f_{D-1}, f_2) & \mathrm{cov}(f_D, f_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{cov}(f_1, f_{D-1}) & \mathrm{cov}(f_2, f_{D-1}) & & \mathrm{cov}(f_{D-1}, f_{D-1}) & \mathrm{cov}(f_D, f_{D-1}) \\ \mathrm{cov}(f_1, f_D) & \mathrm{cov}(f_2, f_D) & & \mathrm{cov}(f_{D-1}, f_D) & \mathrm{cov}(f_D, f_D) \end{bmatrix}$$



# Does $\sum$ always make sense?




Relation between $\sigma^2_{f_1}$ and $\sigma^2_{f_2}$ ?

# Correlation matrix $R$ (normalized $\sum$)

$$R = \begin{bmatrix} \frac{\text{cov}(f_1, f_1)}{\sigma_{f_1}\sigma_{f_1}} & \frac{\text{cov}(f_2, f_1)}{\sigma_{f_2}\sigma_{f_1}} & & \frac{\text{cov}(f_{D-1}, f_1)}{\sigma_{f_{D-1}}\sigma_{f_1}} & \frac{\text{cov}(f_D, f_1)}{\sigma_{f_D}\sigma_{f_1}} \\ \frac{\text{cov}(f_1, f_2)}{\sigma_{f_1}\sigma_{f_2}} & \frac{\text{cov}(f_2, f_2)}{\sigma_{f_2}\sigma_{f_2}} & & \frac{\text{cov}(f_{D-1}, f_2)}{\sigma_{f_{D-1}}\sigma_{f_2}} & \frac{\text{cov}(f_D, f_2)}{\sigma_{f_D}\sigma_{f_2}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\text{cov}(f_1, f_{D-1})}{\sigma_{f_1}\sigma_{f_{D-1}}} & \frac{\text{cov}(f_2, f_{D-1})}{\sigma_{f_2}\sigma_{f_{D-1}}} & & \frac{\text{cov}(f_{D-1}, f_{D-1})}{\sigma_{f_{D-1}}\sigma_{f_{D-1}}} & \frac{\text{cov}(f_D, f_{D-1})}{\sigma_{f_D}\sigma_{f_{D-1}}} \\ \frac{\text{cov}(f_1, f_D)}{\sigma_{f_1}\sigma_{f_D}} & \frac{\text{cov}(f_2, f_D)}{\sigma_{f_2}\sigma_{f_D}} & & \frac{\text{cov}(f_{D-1}, f_D)}{\sigma_{f_{D-1}}\sigma_{f_D}} & \frac{\text{cov}(f_D, f_D)}{\sigma_{f_D}\sigma_{f_D}} \end{bmatrix}$$



# correlation matrix

- Values in [-1, 1]
- Diagonals ($R_{ii}$) always 1
- Off-diagonal $R_{ij}$

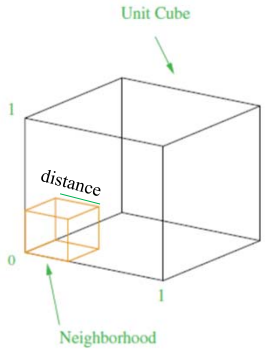
  $> 0 \rightarrow f_i$ and $f_j$ tend to change in same direction

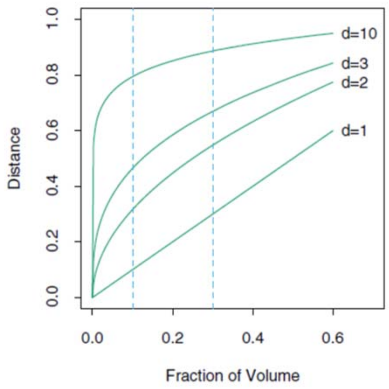
  $< 0 \rightarrow f_i$ and $f_j$ tend to change in opposite directions

  The closer to -1, 1, the stronger the relationship

Munch's *The Scream*

# Curse of dimensionality



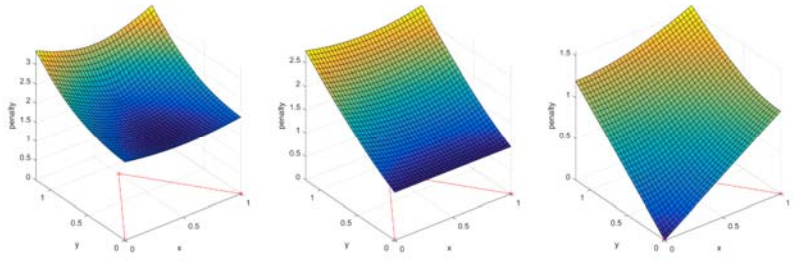

Hastie et al. (2009), p. 23

Training features should cover much of the space…



# Manifolds

- Frequently what we measure can be expressed more compactly.
- Low dimensional representation of higher dimension object



# Manifolds

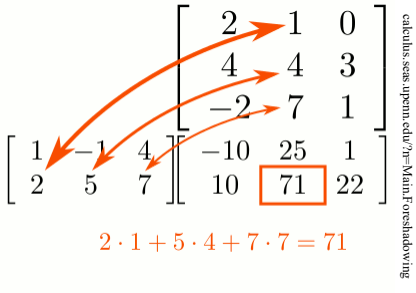
- We'll learn how to do this in a non-linear fashion later in the semester.
- For now, we will use principle components analysis, but we'll need a brief review of linear algebra



# Linear algebra review

Goodfellow et al. 2.2 for details

- Matrix multiplication
  - inner dimension must match:
    [2 x 3][3 x 3]=[2 x 3]



# Eigen vectors

Goodfellow et al. 2.7 for details

- Special vectors such that in

$$Ax = b$$

$$\lambda x = b \text{ for some } \lambda \in \Re$$

- These are the directions in which the matrix merely scales vectors.
- Directions are uncorrelated.
- When the vector is a unit vector, the scale factor is the eigen value.



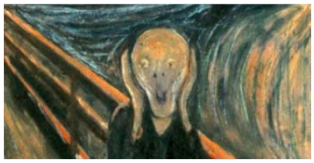
numpy.linalg.eig

see demo



# Principal components analysis (PCA)

- Finds new basis set to represent data
- Relies on eigen vectors and values
- Bases account for different amounts of variance in data and low contributors can be discarded



Munch using PCA ☺

# PCA – Let's get our hands dirty

USFWSmidwest CC 2.0

- Let X be an *NxD* data matrix
- Assume expected value has been subtracted (no loss of generality)
- Can think of feature space as
  - having basis vectors $u_1, u_2, \ldots, u_D$ along axes
  - each row of X is a combination of those vectors



# PCA

- Goal: Pick a new set of basis vectors
  - First vector

$$X \begin{bmatrix} w_{(1),1} \\ w_{(1),\ldots} \\ w_{(1),D} \end{bmatrix} = y$$

produces vector of y values in direction $w_{(1)}$

  - Select such that var(y) is maximized subject to

$$\sum_{i=1}^{D} w_{(1),i}^2 = 1$$

  - Repeat finding next largest uncorrelated basis

# PCA

- In practice, this becomes an eigenvector problem on the variance-covariance matrix
- Principal components are eigenvectors ordered by descending eigenvalue.



# Computing PCA

- Estimate covariance matrix $\sum$ of X
  X is an NxD data matrix
- Compute eigen vectors $e_i$ and values $\lambda_i$ of $\sum$ and arrange by largest eigen value to smallest: $e_1, e_2, \ldots, e_D$
  $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$

# Using PCA

- To project data, multiply by the number of bases desired, e.g. for data matrix X

$$\underbrace{\begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,D} \\ x_{2,1} & x_{2,2} & \dots & x_{2,D} \\ \dots & & & \\ x_{N-1,1} & x_{N-1,2} & \dots & x_{N-1,D} \\ x_{N,1} & x_{N,2} & \dots & x_{N,D} \end{bmatrix}}_{NxD} \underbrace{\begin{bmatrix} e_{1,1} & \dots & e_{m,1} \\ \vdots & & \vdots \\ e_{1,D} & & e_{m,D} \end{bmatrix}}_{Dxm} = \underbrace{\begin{bmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,m} \\ b_{2,1} & b_{2,2} & \dots & b_{2,m} \\ \dots & & & \\ b_{N-1,1} & b_{N-1,2} & \dots & b_{N-1,m} \\ b_{N,1} & b_{N,2} & \dots & b_{N,m} \end{bmatrix}}_{Nxm}$$

$1 \le m \le D$
project to m dimensions



Use numpy.dot to multiply matrices



# How much of the variance is captured?

- $\sum(tr(\Sigma)) = \sum_{i=1}^{D} \lambda_i$ Sum of variances (sum of trace) is the same as the sum of the eigen values
- The first m dimensions contain the variance represented by the sum of their eigen values: $\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{j=1}^{D} \lambda_j}$

# Component loadings

Loadings give the correlation between the bases and the features, e.g. for eigen vector $e_i$:

$$\frac{e_{i,1}\sqrt{\lambda_i}}{\sigma_{f_1}} \quad \frac{e_{i,2}\sqrt{\lambda_i}}{\sigma_{f_2}} \quad \cdots \quad \frac{e_{i,D-1}\sqrt{\lambda_i}}{\sigma_{f_{D-1}}} \quad \frac{e_{i,D}\sqrt{\lambda_i}}{\sigma_{f_D}}$$



# PCA of correlation matrix

- The same analysis can be done on the sample correlation matrix R
- Eigen values will add up to D. Why?
- What is the qualitative difference with this type of analysis?