Features Part I – Principal Component Analysis
Today’s lecture

- Adaptive Feature Extraction
- Principal Component Analysis
  - How, why, when, which
A dual goal

- Find a good representation
  - The features part

- Reduce redundancy in the data
  - A side effect of “proper” features
Example case

- Describe this input
A “good feature”

• “Simplify” the explanation of the input
  • Represent repeating patterns
  • Once defined, makes the input description simpler

• How do we define these abstract qualities?
  • On to the math ...
Linear features

\[ Z = W \cdot X \]

- **Feature Matrix**
- **Weight Matrix**
- **Input Matrix**
A 2D case

\[ Z = W \cdot X = \]

\[
\begin{bmatrix}
    z_1^T \\
    z_2^T
\end{bmatrix}
\begin{bmatrix}
    w_1^T \\
    w_2^T
\end{bmatrix}
\begin{bmatrix}
    x_1^T \\
    x_2^T
\end{bmatrix}
\]

Matrix representation of data

Scatter plot of same data

\[ x1 \]

\[ x2 \]

\[ x1 \]

\[ x2 \]
Defining a goal

- Desirable feature features
  - Give “simple” weights
  - Avoid feature similarity

- How do we define these?
One way to proceed

• “Simple weights”
  • Minimize similarity of the two weight dimensions
    • i.e. $z_1$ and $z_2$ should be “different”

• Avoiding “feature similarity”
  • Same thing for features!
    • $w_1$ and $w_2$ should be “different”
One way to proceed

• “Simple weights”
  • Minimize similarity of the two weight dimensions
    • Decorrelate: $\mathbf{z}_1^\top \cdot \mathbf{z}_2 = 0$

• “Feature similarity”
  • Same thing for features!
    • Decorrelate: $\mathbf{w}_1^\top \cdot \mathbf{w}_2 = 0$
Doing it in a single expression

- Use the covariance matrix

\[
\text{Cov}(\mathbf{z}_1, \mathbf{z}_2) = \begin{bmatrix}
\mathbf{z}_1^\top \cdot \mathbf{z}_1 & \mathbf{z}_1^\top \cdot \mathbf{z}_2 \\
\mathbf{z}_2^\top \cdot \mathbf{z}_1 & \mathbf{z}_2^\top \cdot \mathbf{z}_2
\end{bmatrix} / N = \mathbf{Z} \cdot \mathbf{Z}^\top / N
\]

- \{i,j\} element is the dot product of \{i,j\}-dimension pair
  - Represents all dimension combinations at once

Technically, we divide by \(N-1\) but we simplify for readability.
Diagonalizing the covariance

- Diagonalizing the covariance suppresses co-activation across dimensions
  - if $z_1$ is high, $z_2$ won’t be, etc.

$$\text{Cov}(z_1,z_2) = \begin{bmatrix} z_1^\top \cdot z_1 & z_1^\top \cdot z_2 \\ z_2^\top \cdot z_1 & z_2^\top \cdot z_2 \end{bmatrix} / N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$
Problem definition

• For a given input $X$

• Find a feature matrix $W$

• So that the weights *decorrelate*

\[
(W \cdot X) \cdot (W \cdot X)^T = NI \implies Z \cdot Z^T = NI
\]
How do we solve this?

- Any ideas?

\[
(W \cdot X) \cdot (W \cdot X)^\top = NI
\]
Solving for diagonalization

\[
(W \cdot X) \cdot (W \cdot X)^\top = NI \Rightarrow
\]
\[
\Rightarrow W \cdot X \cdot X^\top \cdot W^\top = NI \Rightarrow
\]
\[
\Rightarrow W \cdot \text{Cov}(X) \cdot W^\top = I
\]
Solving for diagonalization

- Covariance matrices are positive definite
  - And symmetric
    - have orthogonal eigenvectors and real eigenvalues
  - and are factorized by:

\[
A \cdot U = U \cdot \Lambda \Rightarrow U^\top \cdot A \cdot U = \Lambda
\]

- Where \(U\) has the eigenvectors of \(A\) in its columns
- \(\Lambda = \text{diag}(\lambda_i)\), where \(\lambda_i\) are the eigenvalues of \(A\)
Solving for diagonalization

What we have:

\[ U^\top \cdot A \cdot U = \Lambda \quad \text{W} \cdot \text{Cov}(X) \cdot \text{W}^\top = I \]

Substitute:

\[ U_{\text{Cov}(x)}^\top \cdot \text{Cov}(X) \cdot U_{\text{Cov}(x)} = \Lambda_{\text{Cov}(x)} \]

Get the identity:

\[ \left( \Lambda_{\text{Cov}(x)} \right)^{-\frac{1}{2}} U_{\text{Cov}(x)}^\top \cdot \text{Cov}(X) \cdot U_{\text{Cov}(x)} \left( \Lambda_{\text{Cov}(x)} \right)^{-\frac{1}{2}} = I \]

Solve for \( \text{W} \):

\[ \text{W} = \left( \Lambda_{\text{Cov}(x)} \right)^{-\frac{1}{2}} U_{\text{Cov}(x)}^\top = \begin{bmatrix} \sqrt{\lambda_1_{\text{Cov}(x)}} & 0 \\ 0 & \sqrt{\lambda_2_{\text{Cov}(x)}} \end{bmatrix}^{-1} \cdot U_{\text{Cov}(x)}^\top \]
Solving for diagonalization

- The solution is a product of the eigenvectors $U$ and the square root of the eigenvalues $\Lambda$ of $\text{Cov}(X)$

$$W \cdot \text{Cov}(X) \cdot W^\top = I \Rightarrow$$

$$\Rightarrow W = \begin{bmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \end{bmatrix}^{-1} \cdot U^\top$$
Another solution

• That was not the only solution

• Consider this one:  \( \mathbf{W} \cdot \mathbf{X} \cdot (\mathbf{W} \cdot \mathbf{X})^\top = NI \Rightarrow \)
  
  \[
  \mathbf{W} \cdot \mathbf{X} \cdot \mathbf{X}^\top \cdot \mathbf{W}^\top = NI \Rightarrow \\
  \mathbf{W} = \sqrt{N} \left( \mathbf{X} \cdot \mathbf{X}^\top \right)^{-1/2}
  \]
Another solution

- That was not the only solution
  - Consider this one: \( W \cdot X \cdot (W \cdot X)^T = N I \Rightarrow \)
    \[
    W \cdot X \cdot X^T \cdot W^T = N I \Rightarrow \\
    W = \sqrt{N} (X \cdot X^T)^{-1/2}
    \]

- How do you get the matrix square root?
  \[
  W = U \cdot S^{-1/2} \cdot V^T \\
  [U, S, V] = \text{SVD}(X \cdot X^T)
  \]
Decorrelation in pictures

- An implicit Gaussian assumption
- $N$-dimensional data has $N$ directions of variance
Undoing the variance

* The decorrelating matrix $W$ contains two vectors that normalize the input’s variance
Resulting transform

- Input gets scaled to a well-behaved Gaussian with unit variance in all dimensions
A more complex case

- Having correlation between two dimensions
  - We still find the directions of maximal variance
  - But we also rotate in addition to scaling
One more detail

- So far we considered zero-mean inputs
- The transforming operation was a rotation
- If the input mean is not zero bad things happen!
- Make sure that your data is zero-mean!
Principal Component Analysis

• This transform is known as PCA
  • The features are the principal components
    • They are orthogonal to each other
    • And produce orthogonal (white) weights
  • Major tool in statistics
    • Removes dependencies from multivariate data

• Also known as the KLT
  • Karhunen-Loève Transform
A better way to compute PCA

- The Singular Value Decomposition way

\[
[U, S, V] = \text{SVD}(A) \Rightarrow A = U \cdot S \cdot V^\top
\]

- Relationship to eigendecomposition
  - In our case the input will be the $A$ matrix
  - $U$ and $S$ will be like the eigenvectors/values of $A$

- Why the SVD?
  - More stable, more robust, fancy extensions
PCA through the SVD

Feature Matrix → features → Eigenvalue matrix
Dimensions

Input Covariance

Feature Matrix → dimensions → Feature Matrix

Weight Matrix

Input Matrix

= SVD

Feature Matrix

√Eigenvalue matrix

Samples

Feature Matrix

Dimensions

Samples

Dimensions

= SVD
Dimensionality reduction

- PCA is great for high dimensional data

- Allows us to perform dimensionality reduction
  - Helps us find relevant structure in data
  - Helps us throw away things that won’t matter
A simple example

- Two very correlated dimensions
  - e.g. size and weight of fruit
  - One effective variable

- Whitening matrix is:

\[ W = \begin{pmatrix}
-0.36 & -0.34 \\
-14.1 & 14.8 \\
\end{pmatrix} \]

- Large variance between the two components
  - Implies that only one dimension is useful
A simple example

- Second principal component needs to be super-boosted to whiten the weights
  - maybe is it useless?

- Keep only high variance
  - Throw away components with minor variance contributions
What is the number of dimensions?

• If the input was $M$ dimensional, how many dimensions do we keep?
  • No solid answer (estimators exist but can be flaky)

• Look at the singular values/eigenvalues
  • They will show the variance of each component, at some point it will be small
Example

- Eigenvalues of 4800-dimensional video data
  - Little variance after component 20
  - We don’t need to keep the rest of the data
So where are the features?

- We strayed off-subject
  - What happened to the features?
  - We only mentioned that they are orthogonal

- We talked about the weights so far, let’s talk about the principal components
  - They should encapsulate structure
  - How do they look like?
Face analysis

• Analysis of a face database
  • What are good features for faces?

• Is there anything special there?
  • What will PCA give us?
  • Any extra insight?

• Lets see some code in action
The Eigenfaces
Low-rank decompositions

- Big data is painful to deal with
  - We want to reduce their quantity without losing information

- For many operations we will use the low-rank weights
  - More versions of that later
Low-rank projection with PCA

• Make low-rank data with:

\[ z = U^\top \cdot (x - m) \]

• Why not use eigenvalues here? We do not want to amplify lesser info
  • Covariance stays diagonal, but the dimension variances are decreasing

• Approximate data from low-rank representation with:

\[ \hat{x} = U \cdot z + m \]
Low-rank advantage with face data

• Full data
  • 600 faces with $30 \times 26$ pixels $= 600 \times 780 = 468,000$ values

• Low-rank version
  • $K=50$: 600 weights with 50 values + $50 \times 780$ bases $= 69,000$ values
  • $K=15$: 600 weights with 15 values + $15 \times 780$ bases $= 11,700$ values
PCA for large dimensionalities

- Sometimes the data is very high dimensional
  - e.g. 720p videos: $1280 \times 720 \times T = 921,600-D \times T$ frames

- You will not do an SVD that big!
  - Complexity is $O(4m^2n + 8mn^2 + 9n^3)$

- We can now use approximate methods
  - Faster, but sloppier
**EM-PCA**

- Alternate between successive approximations
  - Start with random $W$ and loop over:
    
    $$Z = W^+ \cdot X, \quad W = X \cdot Z^+$$

- After convergence $W$ spans the PCA space
- For low rank $W$ the computations are much faster!
  - More later when we cover EM
Approximate eig/SVD methods

- Rich literature on approximate methods
  - E.g. Lanczos, conjugate gradient, etc.
  - Much faster than regular eig/SVD
    - Especially for estimating a few components

- What to use:
  - MATLAB: `eigs()` and/or `svds()`
  - Python: `scipy.sparse.linalg.eigs()`, `scipy.sparse.linalg.svds()`
    - For symmetric inputs use `scipy.sparse.linalg.eigsh()`
PCA for online data

- Sometimes we have too many data samples
  - Irrespective of the dimensionality
  - e.g. long video recordings

- Incremental SVD algorithms
  - Update the $U$, $S$, $V$ matrices using only a small data subset or even a single sample point
  - Very efficient updates
PCA for online data II

• “Neural net” algorithms
  • Algorithms that update $\mathbf{W}$ one sample at a time
  • Gradient methods

• E.g. Sanger’s rule:

$$\Delta \mathbf{W} = \mathbf{W} \cdot x_i \cdot \left( x_i - \mathbf{W}^\top \cdot \mathbf{W} \cdot x_i \right)^\top$$

$$\mathbf{W} = \mathbf{W} + \mu \Delta \mathbf{W}$$
Back to signals and human perception

- We already talked about perceptual features
  - And how they correlate with classical DSP

- Is there a connection to make with PCA?
An example

- Let’s take a time series which is not “white”
  - Each sample is somewhat correlated with the previous one (Markov process)

\[
\begin{bmatrix}
  x(t), \ldots, x(t+T)
\end{bmatrix}
\]

- We’ll make it multidimensional

\[
\begin{bmatrix}
  x(t) & x(t+1) \\
  \vdots & \vdots \\
  x(t+N) & x(t+1+N)
\end{bmatrix}
\]
An example

- In this context, features will be repeating temporal patterns smaller than $N$

$$Z = W \cdot \begin{bmatrix} x(t) & x(t+1) \\ \vdots & \vdots & \ddots \\ x(t+N) & x(t+1+N) \end{bmatrix}$$

- If $W$ is the Fourier matrix then we are performing a time/frequency analysis
PCA on the time series

• By our definition there is a correlation between successive samples:

\[
\text{Cov}(\mathbf{X}) \approx \begin{bmatrix}
1 & 1-e & \ldots & 0 \\
1-e & 1 & 1-e & \vdots \\
\vdots & 1-e & 1 & 1-e \\
0 & \ldots & 1-e & 1 \\
\end{bmatrix}
\]

• Resulting covariance matrix will be symmetric Toeplitz, tapering from one towards zero
The resulting eigenvectors

- What does this look like? (approximately)
From PCA to a frequency transform

- The eigenvectors of Toeplitz matrices are (approximately) sinusoids of varying frequencies
What about images?

- Take lots of pictures, pick random local patches
- What are the features of this data set?
The components of image patches

• Do PCA on patch matrix and rewrap the components
  • What is this?
2D bases to 1D

- Remember the 2D Fourier transform?
  - We left/right multiply with the Fourier matrix
    \[ \mathbf{Y} = \mathbf{F} \cdot \mathbf{X} \cdot \mathbf{F} \]
  - Which in 1D translates to
    \[ \text{vec}(\mathbf{Y}) = \text{vec}(\mathbf{F} \cdot \mathbf{X} \cdot \mathbf{F}) = (\mathbf{F}^\top \otimes \mathbf{F}) \cdot \text{vec}(\mathbf{X}) \]
- So what does that matrix look like?
2D sinusoidal bases on 1D

- Looks like the PCA components from images!
Not quite a Fourier transform

- Real-valued bases are sinusoids of various frequencies
  - Why not complex-valued? You get these with circulant covariance
- This is known as the Discrete Cosine Transform (DCT)
So now you know

• A sinusoidal transform is an “optimal” decomposition for time series
  • In fact, you will often not do PCA and do a DCT

• There is also a connection with our perceptual system
  • We employ similar filters in our ears and eyes (but we’ll make that connection later)
Comparing the DCT with PCA

• How to compress a picture
  • Chop into small pieces
  • Perform either:
    • DCT transform
    • PCA transform
  • Keep only large weights
  • Resynthesize using approximation
Comparing the DCT with PCA

Original Input

JPEG version

PCA version

0.14 sec

2.1 sec
Recap

• Principal Component Analysis
  • Get used to it!
  • Decorrelates multivariate data, finds useful components, reduces dimensionality

• Many ways to get to it
  • Knowing what to use with your data helps

• Interesting connection to Fourier transform
Next lecture

- Continuing with the features theme
- Independent Component Analysis
  - “PCA on steroids”
  - Strong statistical tool, very interesting properties and connections
- Other useful feature transforms
Reading

• Textbook sections 6.1-6.4
• Eigenfaces (optional)
  • http://en.wikipedia.org/wiki/Eigenface
  • http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf
  • http://www.cs.ucsb.edu/~mturk/Papers/jcn.pdf
• Incremental SVD (optional)
  • http://www.merl.com/publications/TR2002-024/
• EM-PCA (optional)
  • http://cs.nyu.edu/~roweis/papers/empca.ps.gz
Reminder

• There is homework sent out already

• See your TA for questions early!