Today

- Motivation
- PCA algorithms
- Applications
- PCA shortcomings
- Kernel PCA

PCA Applications

- Data Visualization
- Data Compression
- Noise Reduction
- Learning
- Anomaly detection
Example:

- Given 53 blood and urine samples (features) from 65 people.
- How can we visualize the measurements?

• Matrix format (65x53)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>8.0000</td>
<td>4.8200</td>
<td>14.1000</td>
<td>41.0000</td>
<td>85.0000</td>
<td>29.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A2</td>
<td>7.3000</td>
<td>5.0200</td>
<td>14.7000</td>
<td>43.0000</td>
<td>86.0000</td>
<td>29.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A3</td>
<td>4.3000</td>
<td>4.8600</td>
<td>14.9000</td>
<td>42.0000</td>
<td>91.0000</td>
<td>32.0000</td>
<td>35.0000</td>
</tr>
<tr>
<td>A4</td>
<td>7.5000</td>
<td>4.7000</td>
<td>14.9000</td>
<td>45.0000</td>
<td>101.0000</td>
<td>33.0000</td>
<td>33.0000</td>
</tr>
<tr>
<td>A5</td>
<td>7.3000</td>
<td>5.5200</td>
<td>15.4000</td>
<td>46.0000</td>
<td>84.0000</td>
<td>28.0000</td>
<td>33.0000</td>
</tr>
<tr>
<td>A6</td>
<td>8.0000</td>
<td>4.8600</td>
<td>16.0000</td>
<td>47.0000</td>
<td>97.0000</td>
<td>33.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A7</td>
<td>7.8000</td>
<td>4.6800</td>
<td>14.7000</td>
<td>42.0000</td>
<td>92.0000</td>
<td>31.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A8</td>
<td>8.6000</td>
<td>4.8200</td>
<td>15.8000</td>
<td>42.0000</td>
<td>88.0000</td>
<td>33.0000</td>
<td>37.0000</td>
</tr>
<tr>
<td>A9</td>
<td>5.1000</td>
<td>4.7100</td>
<td>14.0000</td>
<td>43.0000</td>
<td>92.0000</td>
<td>30.0000</td>
<td>32.0000</td>
</tr>
</tbody>
</table>

• Spectral format (65 curves, one for each person)

Difficult to compare the different patients...

Difficult to see the correlations between the features...

• Spectral format (53 pictures, one for each feature)

Difficult to see the correlations between the features...
How can we visualize the other variables???
... difficult to see in 4 or higher dimensional spaces...

• Is there a representation better than the coordinate axes?

• Is it really necessary to show all the 53 dimensions?
  - ... what if there are strong correlations between the features?

• How could we find the smallest subspace of the 53-D space that keeps the most information about the original data?

• A solution: Principal Component Analysis

Today

• Motivation
• PCA algorithms
• Applications
• PCA Shortcomings
• Kernel PCA

Principal Component Analysis

PCA:
Orthogonal projection of the data onto a lower-dimension linear space that...
• maximizes variance of projected data (purple line)

• minimizes mean squared distance between
  - data point and
  - projections (sum of blue lines)
Principal Component Analysis

Idea:
- Given data points in a d-dimensional space, project them into a lower dimensional space while preserving as much information as possible.
  - Find best planar approximation to 3D data
  - Find best 12-D approximation to 10^4-D data
- In particular, choose projection that minimizes squared error in reconstructing the original data.

PCA Vectors originate from the center of mass.
- Principal component #1: points in the direction of the largest variance.
- Each subsequent principal component
  - is orthogonal to the previous ones, and
  - points in the directions of the largest variance of the residual subspace

2D Gaussian dataset

1st PCA axis
PCA algorithm I (sequential)

Given the centered data \( \{x_1, \ldots, x_m\} \), compute the principal vectors:

\[
w_k = \arg \max_{w} \frac{1}{m} \sum_{i=1}^{m} (w^T x_i)^2 \quad \text{1st PCA vector}
\]

We maximize the variance of projection of \( x \)

\[
w_2 = \arg \max_{w} \frac{1}{m} \sum_{i=1}^{m} (w^T (x_i - w_1^T x_i))^2 \quad \text{2nd PCA vector}
\]

We maximize the variance of the projection in the residual subspace

\[
x' = w_1(w_1^T x) + w_2(w_2^T x)
\]

PCA algorithm II (sample covariance matrix)

- Given data \( \{x_1, \ldots, x_m\} \), compute covariance matrix \( \Sigma \)

\[
\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x_i - \bar{x})(x_i - \bar{x})^T
\]

where \( \bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i \)

- PCA basis vectors = the eigenvectors of \( \Sigma \)

- Larger eigenvalue \( \Rightarrow \) more important eigenvectors
PCA algorithm II
(sample covariance matrix)

PCA algorithm($X$, $k$): top $k$ eigenvalues/eigenvectors

% $X = N \times m$ data matrix,
% … each data point $x_i = \text{column vector, } i=1..m$

• $x = \frac{1}{m} \sum_{i=1}^{m} x_i$
• $X \leftarrow \text{subtract mean } \bar{x} \text{ from each column vector } x_i \text{ in } X$
• $\Sigma \leftarrow XX^T$ … covariance matrix of $X$
• $\{ \lambda_i, u_i \}_{i=1..N} = \text{eigenvectors/eigenvalues of } \Sigma$
  … $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$
• Return $\{ \lambda_i, u_i \}_{i=1..k}$
% top $k$ PCA components

PCA algorithm III
(SVD of the data matrix)

Singular Value Decomposition of the centered data matrix $X$.

$X = [x_1, \ldots, x_m] \in \mathbb{R}^{N \times m}$, $m$: number of instances, $N$: dimension

$X_{\text{features} \times \text{samples}} = USV^T$

• Columns of $U$
  • the principal vectors, $\{ u^{(1)}, \ldots, u^{(k)} \}$
  • orthogonal and has unit norm – so $U^TU = I$
  • Can reconstruct the data using linear combinations of $\{ u^{(1)}, \ldots, u^{(k)} \}$

• Matrix $S$
  • Diagonal
  • Shows importance of each eigenvector

• Columns of $V^T$
  • The coefficients for reconstructing the samples

Today

• Motivation
• PCA algorithms
• Applications
• PCA Shortcomings
• Kernel PCA
Today

- Motivation
- PCA algorithms
- Applications
  - Face Recognition
  - Image Compression
  - Noise Filtering
- PCA Shortcomings
- Kernel PCA

Face Recognition

- Want to identify specific person, based on facial image
- Robust to glasses, lighting, ...
  - Can’t just use the given 256 x 256 pixels

Applying PCA: Eigenfaces

**Method A:** Build a PCA subspace for each person and check which subspace can reconstruct the test image the best

**Method B:** Build one PCA database for the whole dataset and then classify based on the weights.

Example data set: Images of faces
- Famous Eigenface approach [Turk & Pentland], [Sirovich & Kirby]
- Each face $x$ is ...
  - $256 \times 256$ values (luminance at location)
  - $x$ in $\mathbb{R}^{256 \times 256}$ (view as 64K dim vector)
- Form $X = [x_1, \ldots, x_m]$ centered data mtx
- Compute $\Sigma = XX^T$
- Problem: $\Sigma$ is 64K x 64K ... HUGE!!

Computational Complexity

- Suppose $m$ instances, each of size $N$
  - Eigenfaces: $m=500$ faces, each of size $N=64K$
- Given $N \times N$ covariance matrix $\Sigma$, can compute
  - all $N$ eigenvectors/eigenvalues in $O(N^3)$
  - first $k$ eigenvectors/eigenvalues in $O(k N^2)$
- But if $N=64K$, EXPENSIVE!
A Clever Workaround

- Note that $m < 64K$
- Use $L = XX^T$ instead of $XX^T$
- If $v$ is eigenvector of $L$
  then $Xv$ is eigenvector of $\Sigma$

Proof: $L v = \gamma v$

$X^T X v = \gamma v$

$X (X^T X v) = X(\gamma v) = \gamma X v$

$(XX^T) X v = \gamma (Xv)$

$\Sigma (Xv) = \gamma (Xv)$

Principle Components (Method B)

- Requires carefully controlled data:
  - All faces centered in frame
  - Same size
  - Some sensitivity to angle

- Method is completely knowledge free
  - (sometimes this is good!)
  - Doesn’t know that faces are wrapped around 3D objects (heads)
  - Makes no effort to preserve class distinctions

... faster if train with ...
- only people w/out glasses
- same lighting conditions

Reconstructing (Method B)

Shortcomings

• Requires carefully controlled data:
  - All faces centered in frame
  - Same size
  - Some sensitivity to angle

• Method is completely knowledge free
  - (sometimes this is good!)
  - Doesn’t know that faces are wrapped around 3D objects (heads)
  - Makes no effort to preserve class distinctions
Happiness subspace (method A)

Disgust subspace (method A)

Facial Expression Recognition
Movies

Surprise

Happiness
Facial Expression Recognition
Movies

Disgust

Today

- Motivation
- PCA algorithms
- Applications
  - Face Recognition
  - Image Compression
  - Noise Filtering
- PCA Shortcomings
- Kernel PCA

Original Image

- Divide the original 372x492 image into patches:
  - Each patch is an instance
- View each as a 144-D vector

L₂ error and PCA dim

Relative rec. error

PCA dim
PCA compression: 144D => 60D

PCA compression: 144D => 16D

16 most important eigenvectors

PCA compression: 144D => 6D
6 most important eigenvectors

PCA compression: 144D => 3D

3 most important eigenvectors

PCA compression: 144D => 1D
60 most important eigenvectors

• Looks like the discrete cosine bases of JPG!

2D Discrete Cosine Basis


Today

• Motivation
• PCA algorithms
• Applications
  - Face Recognition
  - Image Compression
  - Noise Filtering
• PCA Shortcomings
• Kernel PCA

Noise Filtering

\[ x \xrightarrow{U} x' \]
Today

• Motivation
• PCA algorithms
• Applications
• PCA Shortcomings
• Kernel PCA

Problematic Data Set for PCA

• PCA doesn’t know labels!
PCA vs. Fisher Linear Discriminant

- PCA maximizes variance, *independent of class*
  => magenta
- FLD attempts to separate classes
  => green line

Problematic Data Set for PCA

- PCA cannot capture NON-LINEAR structure!

PCA Conclusions

- PCA
  - Finds orthonormal basis for data
  - Sorts dimensions in order of “importance”
  - Discard low significance dimensions
- Uses:
  - Get compact description
  - Ignore noise
  - Improve classification (hopefully)
- Not magic:
  - Doesn’t know class labels
  - Can only capture linear variations
- One of many tricks to reduce dimensionality!

Today

- Motivation
- PCA algorithms
- Applications
- PCA Shortcomings
- Kernel PCA
**Dimensionality Reduction**

- Data representation
  - Inputs are real-valued vectors in a high dimensional space.
- Linear structure
  - Does the data live in a low dimensional subspace?
- Nonlinear structure
  - Does the data live on a low dimensional submanifold?

**The “magic” of high dimensions**

- Given some problem, how do we know what classes of functions are capable of solving that problem?

- VC (Vapnik-Chervonenkis) theory tells us that often mappings which take us into a higher dimensional space than the dimension of the input space provide us with greater classification power.

**Example in $\mathbb{R}^2$**

These classes are linearly inseparable in the input space.

**Example: High-Dimensional Mapping**

We can make the problem linearly separable by a simple mapping

$$
\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3
$$

$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$
Kernel Trick

- High-dimensional mapping can seriously increase computation time.
- Can we get around this problem and still get the benefit of high-D?
- Yes! Kernel Trick
  \[ K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \]
- Given any algorithm that can be expressed solely in terms of dot products, this trick allows us to construct different nonlinear versions of it.

Kernel Principle Component Analysis (KPCA)

- Extends conventional principal component analysis (PCA) to a high dimensional feature space using the “kernel trick”.
- Can extract up to \( n \) (number of samples) nonlinear principal components without expensive computations.

Popular Kernels

- Gaussian
  \[ K(x, \tilde{x}) = \exp(-\beta \|x - \tilde{x}\|^2) \]
- Polynomial
  \[ K(x, \tilde{x}) = (1 + \tilde{x} \cdot x)^d \]
- Hyperbolic tangent
  \[ K(x, \tilde{x}) = \tanh(x \cdot \tilde{x} + \delta) \]

Making PCA Non-Linear

- Suppose that instead of using the points \( x_i \) we would first map them to some nonlinear feature space \( \phi(x_i) \)
  - E.g. using polar coordinates instead of cartesian coordinates would help us deal with the circle.
- Extract principal component in that space (PCA)
- The result will be non-linear in the original data space!
Derivation

• Suppose that the mean of the data in the feature space is
  \[ \mu = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) = 0 \]

• Covariance:
  \[ C = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T \]

• Eigenvectors
  \[ CV = \lambda V \]

Derivation (cont’d.)

• Eigenvectors can be expressed as linear combination of features:
  \[ v = \sum_{i=1}^{n} \alpha_i \phi(x_i) \]

• Proof:
  \[ CV = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T v = \lambda v \]

  thus
  \[ v = \frac{1}{\lambda n} \sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T v = \frac{1}{\lambda n} \sum_{i=1}^{n} (\phi(x_i) \cdot v)\phi(x_i)^T \]

Showing that \( xx^Tv = (x \cdot v)x^T \)

\[
(x x^T)v = \begin{pmatrix}
  x_1 x_1 & x_1 x_2 & \ldots & x_1 x_M \\
  x_2 x_1 & x_2 x_2 & \ldots & x_2 x_M \\
  \vdots & \vdots & \ddots & \vdots \\
  x_M x_1 & x_M x_2 & \ldots & x_M x_M
\end{pmatrix}
\begin{pmatrix}
  v_1 \\
  v_2 \\
  \vdots \\
  v_M
\end{pmatrix}
\]

\[ = \begin{pmatrix}
  x_1 x_1 v_1 + x_1 x_2 v_2 + \ldots + x_1 x_M v_M \\
  x_2 x_1 v_1 + x_2 x_2 v_2 + \ldots + x_2 x_M v_M \\
  \vdots \\
  x_M x_1 v_1 + x_M x_2 v_2 + \ldots + x_M x_M v_M
\end{pmatrix}
\]

\[ = (x \cdot v)x \]

Showing that \( xx^Tv = (x \cdot v)x^T \)

\[
(x x^T)v = \begin{pmatrix}
  x_1 v_1 & x_1 v_2 & \ldots & x_1 v_M \\
  x_2 v_1 & x_2 v_2 & \ldots & x_2 v_M \\
  \vdots & \vdots & \ddots & \vdots \\
  x_M v_1 & x_M v_2 & \ldots & x_M v_M
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_M
\end{pmatrix}
\]

\[ = \begin{pmatrix}
  x_1 v_1 + x_2 v_2 + \ldots + x_M v_M \cdot x_1 \\
  (x_1 v_1 + x_2 v_2 + \ldots + x_M v_M) \cdot x_2 \\
  \vdots \\
  (x_1 v_1 + x_2 v_2 + \ldots + x_M v_M) \cdot x_M
\end{pmatrix}
\]

\[ = (x \cdot v)x \]
Derivation (cont’d.)

• So, from before we had,
  \[ v = \frac{1}{n\lambda} \sum_{i=1}^{n} \phi(x_i)\phi(x_i)^Tv = \frac{1}{n\lambda} \sum_{i=1}^{n} (\phi(x_i)\cdot v)\phi(x_i)^T \]
  
  just a scalar

• this means that all solutions \( v \) with \( \lambda = 0 \) lie in the span of \( \phi(x_1),...\phi(x_n) \), i.e.,
  \[ v = \sum_{i=1}^{n} \alpha_i \phi(x_i) \]

• Finding the eigenvectors is equivalent to finding the coefficients \( \alpha_i \)

Derivation (cont’d.)

• By substituting this back into the equation we get:
  \[ \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)\phi(x_i)^T \left( \sum_{j=1}^{n} \alpha_j \phi(x_j) \right) = \lambda_j \sum_{j=1}^{n} \alpha_j \phi(x_j) \]

• We can rewrite it as
  \[ \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \left( \sum_{j=1}^{n} \alpha_j K(x_i,x_j) \right) = \lambda_j \sum_{j=1}^{n} \alpha_j \phi(x_j) \]

• Multiply this by \( \phi(x_k) \) from the left:
  \[ \frac{1}{n} \sum_{i=1}^{n} \phi(x_k)^T \phi(x_i) \left( \sum_{j=1}^{n} \alpha_j K(x_i,x_j) \right) = \lambda_j \sum_{j=1}^{n} \alpha_j \phi(x_k)^T \phi(x_j) \]

Derivation (cont’d.)

• By plugging in the kernel and rearranging we get:
  \[ K^2 \alpha_j = n\lambda_j K \alpha_j \]

  We can remove a factor of \( K \) from both sides of the matrix (this will only affects the eigenvectors with zero eigenvalue, which will not be a principle component anyway):
  \[ K \alpha_j = n\lambda_j \alpha_j \]

• We have a normalization condition for \( \alpha_j \) vectors:
  \[ v_j^Tv_j = 1 \Rightarrow \sum_{k=1}^{n} \sum_{l=1}^{n} \alpha_j \alpha_k \phi(x_i)^T \phi(x_k) = 1 \Rightarrow \alpha_j^T K \alpha_j = 1 \]

Derivation (cont’d.)

• By multiplying \( K \alpha_j = n\lambda_j \alpha_j \) by \( \alpha_j \) and using the normalization condition we get:
  \[ \lambda_j n \alpha_j^T \alpha_j = 1, \quad \forall j \]

• For a new point \( x \), its projection onto the principal components is:
  \[ \phi(x)^Tv_j = \sum_{j=1}^{n} \alpha_j \phi(x)^T \phi(x_i) = \sum_{i=1}^{n} \alpha_j K(x,x_i) \]
Summary of kernel PCA

- In general, $\phi(x_i)$ may not be zero mean.
- Centered features:
  \[ \tilde{\phi}(x_i) = \phi(x_i) - \frac{1}{n} \sum_{k=1}^{n} \phi(x_k) \]
- The corresponding kernel is:
  \[ \tilde{K}(x_i, x_j) = \tilde{\phi}(x_i)^T \tilde{\phi}(x_j) \]
  \[ = \left( \phi(x_i) - \frac{1}{n} \sum_{k=1}^{n} \phi(x_k) \right)^T \left( \phi(x_j) - \frac{1}{n} \sum_{l=1}^{n} \phi(x_l) \right) \]
  \[ = K(x_i, x_j) - \frac{1}{n} \sum_{k=1}^{n} K(x_i, x_k) - \frac{1}{n} \sum_{l=1}^{n} K(x_j, x_l) + \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} K(x_k, x_l) \]

Normalizing the feature space

- In general, $\phi(x_i)$ may not be zero mean.
- Centered features:
  \[ \tilde{\phi}(x_i) = \phi(x_i) - \frac{1}{n} \sum_{k=1}^{n} \phi(x_k) \]
- The corresponding kernel is:
  \[ \tilde{K}(x_i, x_j) = \tilde{\phi}(x_i)^T \tilde{\phi}(x_j) \]
  \[ = \left( \phi(x_i) - \frac{1}{n} \sum_{k=1}^{n} \phi(x_k) \right)^T \left( \phi(x_j) - \frac{1}{n} \sum_{l=1}^{n} \phi(x_l) \right) \]
  \[ = K(x_i, x_j) - \frac{1}{n} \sum_{k=1}^{n} K(x_i, x_k) - \frac{1}{n} \sum_{l=1}^{n} K(x_j, x_l) + \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} K(x_k, x_l) \]

Normalizing the feature space (cont’d.)

- In a matrix form
  \[ \tilde{K} = K - 21_{1/n} K + 1_{1/n} K 1_{1/n} \]
  where $1_{1/n}$ is a matrix with all elements $1/n$.  

Summary of Kernel PCA

- Pick a kernel
- Construct the normalized kernel matrix of the data (dimension $m \times m$):
  \[ \tilde{K} = K - 21_{1/n} K + 1_{1/n} K 1_{1/n} \]
- Solve an eigenvalue problem:
  \[ \tilde{K} \alpha_j = \lambda_j \alpha_j \]
- For any data point (new or old), we can represent it as
  \[ y_j = \sum_{i=1}^{n} \alpha_j K(x, x_i), \ j = 1, \ldots, d \]

Input points before kernel PCA

http://en.wikipedia.org/wiki/Kernel_principal_component_analysis
Output after kernel PCA

- The three groups are distinguishable using the first component only: \( \kappa(x, y) = (x^T y + 1)^2 \)

Example: De-noising images

- De-noising images

Properties of KPCA

- Kernel PCA can give a good re-encoding of the data when it lies along a non-linear manifold.
- The kernel matrix is \( n \times n \), so kernel PCA will have difficulties if we have lots of data points.