Introduction to Machine Learning
CMU-10701

14. Principal Component Analysis

Barnabás Póczos
Contents

- Motivation
- PCA algorithms
- Applications

Some of these slides are taken from
- Karl Booksh Research group
- Tom Mitchell
- Ron Parr
Motivation
PCA Applications

- Data Visualization
- Data Compression
- Noise Reduction
Example:

• Given 53 blood and urine samples (features) from 65 people.

• How can we visualize the measurements?
### Data Visualization

- **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>4.4800</td>
<td>14.1000</td>
<td>41.0000</td>
<td>91.0000</td>
<td>32.0000</td>
<td>35.0000</td>
</tr>
<tr>
<td>A4</td>
<td>7.5000</td>
<td>4.4700</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>6.9000</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>43.0000</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>

Difficult to see the correlations between the features...
Data Visualization

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

Difficult to compare the different patients...
Data Visualization

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

Difficult to see the correlations between the features...
Data Visualization

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**
PCA Algorithms
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)
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
1\textsuperscript{st} PCA axis
2nd PCA axis
Given the **centered** data \( \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \), compute the principal vectors:

\[
\mathbf{w}_1 = \arg \max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{ (\mathbf{w}^T \mathbf{x}_i)^2 \} \quad \text{1st PCA vector}
\]

We maximize the variance of projection of \( \mathbf{x} \)

\[
\mathbf{w}_2 = \arg \max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{ [\mathbf{w}^T (\mathbf{x}_i - \mathbf{w}_1 \mathbf{w}_1^T \mathbf{x}_i)]^2 \} \quad \text{\( k \)th PCA vector}
\]

\( \mathbf{x}' \) PCA reconstruction

We maximize the variance of the projection in the residual subspace

\[
\mathbf{x}' = \mathbf{w}_1 (\mathbf{w}_1^T \mathbf{x})
\]
PCA algorithm I (sequential)

Given $\mathbf{w}_1, \ldots, \mathbf{w}_{k-1}$, we calculate $\mathbf{w}_k$ principal vector as before:

Maximize the variance of projection of $\mathbf{x}$

$$
\mathbf{w}_k = \arg \max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{ [\mathbf{w}^T (\mathbf{x}_i - \sum_{j=1}^{k-1} \mathbf{w}_j \mathbf{w}_j^T \mathbf{x}_i)]^2 \}
$$

$k^{th}$ PCA vector

$x'$ PCA reconstruction

We maximize the variance of the projection in the residual subspace

$$
x' = \mathbf{w}_1 (\mathbf{w}_1^T \mathbf{x}) + \mathbf{w}_2 (\mathbf{w}_2^T \mathbf{x})
$$
Animation
PCA algorithm II
(sample covariance matrix)

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

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

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 $x_i = \text{column vector}$, $i=1..m$

- $\mathbf{x} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$
- $X \leftarrow \text{subtract mean } \mathbf{x} \text{ from each column vector } \mathbf{x}_i \text{ in } X$
- $\Sigma \leftarrow XX^T \quad \text{... covariance matrix of } X$
- $\{ \lambda_i, u_i \}_{i=1..N} = \text{eigenvectors/eigenvalues of } \Sigma$
  ... $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$
- 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 $X$.

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

$$N: \text{dimension}$$

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

$$X = U \quad S \quad V^T$$

- $U$: significant
- $S$: noise
- $V^T$: significant

samples
PCA algorithm III

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

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

• **Columns of V^T**
  • The coefficients for reconstructing the samples
Applications
Face Recognition

- Want to identify specific person, based on facial image
- Robust to glasses, lighting,

  $\Rightarrow$ 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 \( \mathbf{x} \) is ...
  - 256 \( \times \) 256 values (luminance at location)
  - \( \mathbf{x} \) in \( \mathbb{R}^{256 \times 256} \) (view as 64K dim vector)

- Form \( \mathbf{X} = [ \mathbf{x}_1, \ldots, \mathbf{x}_m ] \) centered data mtx

- Compute \( \Sigma = \mathbf{X} \mathbf{X}^\top \)

- Problem: \( \Sigma \) is 64K \( \times \) 64K ... HUGE!!!
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!
• Note that $m << 64K$
• Use $L = XX^T$ instead of $\Sigma = XX^T$
• If $v$ is eigenvector of $L$
  then $Xv$ is eigenvector of $\Sigma$

Proof: 

\[
\begin{align*}
L \ v &= \gamma \ v \\
X^T X \ v &= \gamma \ v \\
X (X^T X \ v) &= X(\gamma \ v) = \gamma X v \\
(XX^T)X \ v &= \gamma (Xv) \\
\Sigma (Xv) &= \gamma (Xv)
\end{align*}
\]
Principle Components (Method B)
... faster if train with...

- only people w/out glasses
- same lighting conditions
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
Facial Expression Recognition
Movies
Facial Expression Recognition
Movies

Disgust
Image Compression
• Divide the original 372x492 image into patches:
  • Each patch is an instance that contains 12x12 pixels on a grid
  • View each as a 144-D vector
L₂ error and PCA dim
PCA compression: 144D $\Rightarrow$ 60D
PCA compression: 144D $\Rightarrow$ 16D
16 most important eigenvectors
PCA compression: 144D $\Rightarrow$ 6D
6 most important eigenvectors
PCA compression: 144D $\Rightarrow$ 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

Noise Filtering
Noise Filtering
Noisy image
Denoised image using 15 PCA components
PCA Shortcomings
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!
Kernel PCA
Performing PCA in the feature space

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

Lemma

$u$ is eigenvector of $\Sigma \Rightarrow u$ is a linear combination of the samples

Proof:

$\lambda u = \Sigma u = \left( \frac{1}{m} \sum_{i=1}^{m} x_i x_i^T \right) u = \frac{1}{m} \sum_{i=1}^{m} (x_i^T u)x_i$

$\Rightarrow u = \sum_{i=1}^{m} \frac{(x_i^T u)}{\lambda m} x_i = \sum_{i=1}^{m} \alpha_i x_i$
Kernel PCA

\[ u = \sum_{i=1}^{m} \frac{(x_i^T u)}{\lambda m} x_i = \sum_{i=1}^{m} \alpha_i x_i \quad X = [x_1, \ldots, x_m] \in \mathbb{R}^{N \times m}, \]

Lemma

To calculate \( \alpha \in \mathbb{R}^m \)

- just use inner products (Gram matrix): \( K_{ij} = x_i^T x_j \)
- don’t need the actual values of \( x_i \)
Kernel PCA

Proof

\[ \Sigma u = \lambda u, \quad u = \sum_{j=1}^{m} \alpha_j x_j \]

\[ \Rightarrow x_i^T \Sigma u = \lambda x_i^T u \]

\[ \Rightarrow x_i^T \left( \frac{1}{m} \sum_{k=1}^{m} x_k x_k^T \right) \left( \sum_{j=1}^{m} \alpha_j x_j \right) = \lambda x_i^T \left( \sum_{j=1}^{m} \alpha_j x_j \right) \]

\[ \Rightarrow \frac{1}{m} \sum_{k=1}^{m} \sum_{j=1}^{m} (x_i^T x_k)(x_k^T x_j) \alpha_j = \lambda \sum_{j=1}^{m} (x_i^T x_j) \alpha_j \]

\[ \Rightarrow \frac{1}{m} K^2 \alpha = \lambda K \alpha \quad \text{where } K \in \mathbb{R}^{m \times m} \]

\[ \Rightarrow K \alpha = m \lambda \alpha \quad \text{If } K \text{ is invertible (strictly pos def)} \]
Kernel PCA

How to use $\alpha$ to calculate the projection of a new sample $t$?

$$u^T t = \left( \sum_{j=1}^{m} \alpha_j x_j \right)^T t = \sum_{j=1}^{m} \alpha_j K(x_j, t)$$

Again, we don’t need values of $x_j$!
Let $K_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle$

Where was I cheating? 😊

The data should be centered in the feature space, too!
But this is manageable...

$$\tilde{K}_{i,j} = \left\langle \phi(x_i) - \frac{1}{m} \sum_{k=1}^{m} \phi(x_k), \phi(x_j) - \frac{1}{m} \sum_{k=1}^{m} \phi(x_k) \right\rangle$$
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

\[ k(x, y) = (x^T y + 1)^2 \]
PCA Theory
Justification of Algorithm II

Let $x \in \mathbb{R}^N$

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

Let $U = \begin{pmatrix} u_1^T \\ \vdots \\ u_N^T \end{pmatrix} \in \mathbb{R}^{N \times N}$ orthogonal matrix, $U^T U = I_N$

$y = Ux$, $x = U^T y = \sum_{i=1}^{N} u_i y_i$

$\hat{x} = \sum_{i=1}^{M} u_i y_i$, $(M \leq N)$ approximation of $x$ using $M$ basis vectors only.

$\varepsilon^2 = \mathbb{E}\{\|x - \hat{x}\|^2\} = \frac{1}{m} \sum_{j=1}^{m} \|x_j - \hat{x}_j\|$, average error

GOAL:

$$\arg\min_{U} \varepsilon^2, \text{ s.t. } U^T U = I_N$$
\[ \varepsilon^2 = \mathbb{E}\{\|x - \hat{x}\|^2\} = \mathbb{E}\{\| \sum_{i=1}^{N} u_i y_i - \sum_{i=1}^{M} u_i y_i \|^2\} \]

\[ = \mathbb{E}\{ \sum_{i=M+1}^{N} y_i u_i^T u_i y_i \} = \sum_{i=M+1}^{N} \mathbb{E}\{ y_i^2 \} \]

\[ = \sum_{i=M+1}^{N} \mathbb{E}\{(u_i^T x)(x^T u_i)\} \]

\[ = \sum_{i=M+1}^{N} u_i^T \mathbb{E}\{xx^T\} u_i \quad \text{x is centered!} \]

\[ = \sum_{i=M+1}^{N} u_i^T \Sigma u_i \]
Justification of Algorithm II

**GOAL:** \( \text{arg} \min_{u_{M+1}, \ldots, u_N} \varepsilon^2 \)

Use Lagrange-multipliers for the constraints.

\[
L = \varepsilon^2 - \sum_{i=M+1}^{N} \lambda_i (u_i^T u_i - 1)
\]

\[
= \sum_{i=M+1}^{N} u_i^T \Sigma u_i - \sum_{i=M+1}^{N} \lambda_i (u_i^T u_i - 1)
\]

\[
\frac{\partial L}{\partial u_i} = [2\Sigma u_i - 2\lambda_i u_i] = 0
\]
Justification of Algorithm II

$$\frac{\partial L}{\partial u_i} = [2\Sigma u_i - 2\lambda_i u_i] = 0 \Rightarrow \Sigma u_i = \lambda_i u_i$$

$$\Rightarrow [u_i, \lambda_i] = \text{eigenvector/eigenvalue of } \Sigma.$$}

$$\varepsilon^2 = \sum_{i=M+1}^N u_i^T \Sigma u_i = \sum_{i=M+1}^N u_i^T \lambda_i u_i = \sum_{i=M+1}^N \lambda_i$$

The error $\varepsilon^2$ is minimal if $\lambda_{M+1}, \ldots, \lambda_N$ are the smallest eigenvalues of $\Sigma$, and $u_{M+1}, \ldots, u_N$ are the corresponding eigenvectors.
Thanks for the Attention! 😊