Principal Component Analysis

Nuno Vasconcelos

ECE Department, UCSD
Curse of dimensionality

- typical observation in Bayes decision theory:
  - error increases when number of features is large

- problem: even for simple models (e.g. Gaussian) we need large # of examples $n$ to have good estimates

- Q: what does “large” mean? This depends on the dimension of the space

- the best way to see this is to think of an histogram
  - suppose you have 100 points and you need at least 10 bins per axis in order to get a reasonable quantization

- for uniform data you get, on average,

- decent in 1D, bad in 2D, terrible in 3D

<table>
<thead>
<tr>
<th>dimension</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>points/bin</td>
<td>10</td>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Curse of dimensionality

- this is the curse of dimensionality
  - for a given classifier
  - number of examples required to maintain classification accuracy
  - increases exponentially with the dimension of the space

- in higher dimensions the classifier has more parameters
  - higher complexity, harder to learn
Dimensionality reduction

- what do we do about this? we **avoid unnecessary dimensions**
- unnecessary can be measured in two ways:
  1. features are not discriminant
  2. features are not independent
- non-discriminant means that they **do not separate the classes well**
Dimensionality reduction

Q: how do we detect the presence of feature correlations?

A: the data “lives” in a low dimensional subspace (up to some amounts of noise). E.g.

if we can find this hyper-plane we can

- project the data onto it
- get rid of half of the dimensions without introducing significant error
Principal component analysis

Basic idea:

- If the data lives in a subspace, it is going to look very flat when viewed from the full space, e.g.

1D subspace in 2D

2D subspace in 3D

This means that

- If we fit a Gaussian to the data
- The iso-probability contours are going to be highly skewed ellipsoids
Principal component analysis

how do we find these ellipsoids?

when we talked about metrics we said that

• the Mahalanobis distance
• measures the “natural” units for the problem
• because it is “adapted” to the covariance of the data

we also know that

• what is special about it
• is that it uses $\Sigma^{-1}$

hence, the information must be in $\Sigma$

$$d(x, y) = (x - y)^T \Sigma^{-1} (x - y)$$
Gaussian review

the equiprobability contours of a Gaussian are the points such that

\[(x - \mu)^T \Sigma^{-1} (x - \mu) = K\]

let’s consider the change of variable \( z = x - \mu \), which only moves the origin by \( \mu \). The equation

\[z^T \Sigma^{-1} z = K\]

is the equation of an ellipse.

this is easy to see when \( \Sigma \) is diagonal:

\[
\Sigma = \Lambda = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2) \Rightarrow z^T \Sigma^{-1} z = \sum_i \frac{z_i^2}{\sigma_i^2} = K
\]
Gaussian review

- this is the equation of an ellipse with principal lengths $\sigma_i$
  - e.g. when $d = 2$

$$\frac{z_1^2}{\sigma_1^2} + \frac{z_2^2}{\sigma_2^2} = 1$$

- is the ellipse

- introduce the transformation $y = \Phi z$
Gaussian review

- Introduce the transformation $y = \Phi z$
- Then $y$ has covariance $\Sigma_y = \Phi \Sigma_z \Phi^T = \Phi \Lambda \Phi^T$
- If $\Phi$ is orthonormal this is just a rotation and we have

![Diagram of rotated ellipse]

- We obtain a rotated ellipse with principal components $\phi_1$ and $\phi_2$ which are the columns of $\Phi$
- Note that $\Sigma_y = \Phi \Lambda \Phi^T$ is the eigen-decomposition of $\Sigma_y$
Principal component analysis

- If $y$ is Gaussian with covariance $\Sigma$, the equiprobability contours are the ellipses whose principal components $\phi_i$ are the eigenvectors of $\Sigma$.
- Principal lengths $\lambda_i$ are the eigenvalues of $\Sigma$.

- By computing the eigenvalues we know if the data is flat:
  - $\lambda_1 >> \lambda_2$: flat
  - $\lambda_1 = \lambda_2$: not flat
Principal component analysis (learning)

- Given sample $D = \{x_1, \ldots, x_n\}, \ x_i \in \mathbb{R}^d$

  - compute sample mean: $\hat{\mu} = \frac{1}{n} \sum_i (x_i)$

  - compute sample covariance: $\hat{\Sigma} = \frac{1}{n} \sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T$

  - compute eigenvalues and eigenvectors of $\hat{\Sigma}$
    $$\hat{\Sigma} = \Phi \Lambda \Phi^T, \ \Lambda = diag(\sigma_1^2, \ldots, \sigma_n^2) \quad \Phi^T \Phi = I$$

  - order eigenvalues $\sigma_1^2 > \ldots > \sigma_n^2$

  - if, for a certain $k$, $\sigma_k << \sigma_1$ eliminate the eigenvalues and eigenvectors above $k$. 


Principal component analysis

- Given principal components \( \phi_i, i \in 1,\ldots, k \) and a test sample \( T = \{t_1,\ldots, t_n\}, \; t_i \in \mathcal{R}^d \)

  - subtract mean to each point \( t_i' = t_i - \mu \)

  - project onto eigenvector space \( y_i = A t_i' \) where

    \[
    A = \begin{bmatrix}
    \phi_1^T \\
    \vdots \\
    \phi_k^T
    \end{bmatrix}
    \]

  - use \( T' = \{y_1,\ldots, y_n\} \) to estimate class conditional densities and do all further processing on \( y \).
Principal component analysis

how do I determine the number of eigenvectors to keep?

• one possibility is to plot eigenvalue magnitudes
• this is a scree plot
• usually there is a fast decrease in the eigenvalue magnitude
• followed by a flat area
• one good choice is the knee of this curve
Principal component analysis

- another possibility is the percentage of explained variance
  - remember that eigenvalues are a measure of variance

- ratio \( r_k \) measures % of total variance contained in the top \( k \) eigenvalues
- measure of the fraction of data variability along the associated eigenvectors

\[
y = \Phi z
\]

\[
\sum_{i=1}^{k} \lambda_i^2 = \sum_{i=1}^{n} \lambda_i^2
\]

\[
r_k = \frac{\sum_{i=1}^{k} \lambda_i^2}{\sum_{i=1}^{n} \lambda_i^2}
\]
Principal component analysis

- A natural measure is to pick the eigenvectors that explain \( p \% \) of the data variability.
  - Can be done by plotting the ratio \( r_k \) as a function of \( k \).

\[
\sum_{i=1}^{k} \lambda_i^2 = \sum_{i=1}^{n} \lambda_i^2
\]

- E.g. we need 3 eigenvectors to cover 70% of the variability of this dataset.
Principal component analysis

▶ there is an alternative manner to compute the principal components, based on singular value decomposition

▶ SVD:

• any real \( n \times m \) matrix \((n>m)\) can be decomposed as

\[
A = M \Sigma N^T
\]

• where \( M \) is a \( n \times m \) column orthonormal matrix of left singular vectors (columns of \( M \))
• \( \Sigma \) a \( m \times m \) diagonal matrix of singular values
• \( N^T \) a \( m \times m \) row orthonormal matrix of right singular vectors (columns of \( N \))

\[
M^T M = I \quad N^T N = I
\]
PCA by SVD

- to relate this to PCA, we consider the data matrix

\[
X = \begin{bmatrix}
  x_1 & \ldots & x_n \\
\end{bmatrix}
\]

- the sample mean is

\[
\mu = \frac{1}{n} \sum_i x_i = \frac{1}{n} \begin{bmatrix}
  x_1 & \ldots & x_n \\
\end{bmatrix} \begin{bmatrix}
  1 \\
  \vdots \\
  1 \\
\end{bmatrix} = \frac{1}{n} X 1
\]
PCA by SVD

- and we can center the data by subtracting the mean to each column of $X$
- this is the centered data matrix

$$X_c = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} - \begin{bmatrix} \mu & \ldots & \mu \end{bmatrix}$$

$$= X - \mu 1^T = X - \frac{1}{n} X 1 1^T = X \left( I - \frac{1}{n} 1 1^T \right)$$
PCA by SVD

- The sample covariance is

\[
\Sigma = \frac{1}{n} \sum_{i} (x_i - \mu)(x_i - \mu)^T = \frac{1}{n} \sum_{i} x_i^c (x_i^c)^T
\]

where \( x_i^c \) is the \( i \)th column of \( X_c \)

- This can be written as

\[
\Sigma = \frac{1}{n} \begin{bmatrix}
- & x_1^c & - \\
- & \cdots & - \\
- & x_n^c & - \\
\end{bmatrix} = \frac{1}{n} X_c X_c^T
\]
PCA by SVD

the matrix

\[ X_c^T = \begin{bmatrix} - & X_1^c & - \\ & \vdots & \\ - & X_n^c & - \end{bmatrix} \]

is real \( n \times d \). Assuming \( n > d \) it has SVD decomposition

\[ X_c^T = \mathbf{M} \mathbf{N} \mathbf{N}^T \]

and

\[ \Sigma = \frac{1}{n} X_c X_c^T = \frac{1}{n} \mathbf{N} \mathbf{M} \mathbf{M}^T \mathbf{M} \mathbf{N} \mathbf{N}^T = \frac{1}{n} \mathbf{N} \mathbf{N}^2 \mathbf{N}^T \]
PCA by SVD

\[ \Sigma = N \left( \frac{1}{n} \Pi^2 \right) N^T \]

- noting that \( N \) is \( d \times d \) and orthonormal, and \( \Pi^2 \) diagonal, shows that this is just the eigenvalue decomposition of \( \Sigma \)

- it follows that
  - the eigenvectors of \( \Sigma \) are the columns of \( N \)
  - the eigenvalues of \( \Sigma \) are

\[ \lambda_i = n \sqrt{\pi_i} \]

- this gives an alternative algorithm for PCA
PCA by SVD

- computation of PCA by SVD
- given $X$ with one example per column
  - 1) create the centered data-matrix
    $$X_c^T = \left( I - \frac{1}{n} 11^T \right) X^T$$
  - 2) compute its SVD
    $$X_c^T = M\Pi N^T$$
  - 3) principal components are columns of $N$, eigenvalues are
    $$\lambda_i = n\sqrt{\pi_i}$$
Principal component analysis

- Principal components are usually quite informative about the structure of the data.

- Example
  - The principal components for the space of images of faces.
  - The figure only show the first 16 eigenvectors.
  - Note lighting, structure, etc.
Principal components analysis

- PCA has been applied to virtually all learning problems
- e.g. eigenshapes for face morphing
Principal component analysis

- sound

average sound images

Eigenobjects corresponding to the three highest eigenvalues
Principal component analysis

- turbulence
- flames
- eigenflames
Principal component analysis

> video

![Eigenrings and Reconstruction Images]

KL Spectrum of Dominant Modes (% of Energy VS Number of Vectors)

Successive KL Reconstructions (Frame 0001)
Principal component analysis

**text:** latent semantic indexing

- represent each document by a word histogram
- perform SVD on the document x word matrix

- principal components as the directions of semantic concepts
Latent semantic analysis

► applications:
  • document classification, information

► goal: solve two fundamental problems in language
  • synonymy: different writers use different words to describe the same idea.
  • polysemy, the same word can have multiple meanings

► reasons:
  • original term-document matrix is too large for the computing resources
  • original term-document matrix is noisy: for instance, anecdotal instances of terms are to be eliminated.
  • original term-document matrix overly sparse relative to "true" term-document matrix. E.g. lists only words actually in each document, whereas we might be interested in all words related to each document-- much larger set due to synonymy
Latent semantic analysis

After PCA some dimensions get "merged":

- \{(\text{car}), (\text{truck}), (\text{flower})\} \rightarrow \{(1.3452 \times \text{car} + 0.2828 \times \text{truck}), (\text{flower})\}

This mitigates synonymy,

- merges the dimensions associated with terms that have similar meanings.

And mitigates polysemy,

- components of polysemous words that point in the "right" direction are added to the components of words that share this sense.
- conversely, components that point in other directions tend to either simply cancel out, or, at worst, to be smaller than components in the directions corresponding to the intended sense.
Extensions

next lecture we will talk about kernels

• turns out that any algorithm which depends on the data through dot-products only, i.e. the matrix of elements
\[ x_i^T x_j \]
can be kernelized

• this is usually beneficial, we will see why later

• for now we look at the question of whether PCA can be written in the form above

recall the data matrix is
\[ X = \begin{bmatrix}
  x_1 & \ldots & x_n \\
\end{bmatrix} \]
Extensions

- the centered-data matrix and the covariance

\[
X_c = X \left( I - \frac{1}{n} 11^T \right) \quad \Sigma = \frac{1}{n} X_c X_c^T
\]

- the eigenvector \( \phi_i \) of eigenvalue \( \lambda_i \) is

\[
\phi_i = \frac{1}{n \lambda_i} X_c X_c^T \phi_i = \frac{1}{n \lambda_i} X_c \alpha_i, \quad \alpha_i = X_c^T \phi_i
\]

- hence, the eigenvector matrix is

\[
\Phi = X_c \Gamma, \quad \Gamma = \begin{bmatrix}
\alpha_1 / n \lambda_d & \cdots & \alpha_d / n \lambda_d
\end{bmatrix}
\]
Extensions

we next note that, from the eigenvector decomposition

\[ \Sigma = \Phi \Lambda \Phi^T \iff \Lambda = \Phi^T \Sigma \Phi \]

and

\[
\Lambda = \Gamma^T X_c^T \left( \frac{1}{n} X_c X_c^T \right) X_c \Gamma \\
= \frac{1}{n} \Gamma^T (X_c^T X_c) (X_c^T X_c) \Gamma
\]

i.e.

\[
\frac{1}{n} (X_c^T X_c) (X_c^T X_c) = \Gamma \Lambda \Gamma^T
\]
Extensions

► in summary, we have

\[ \Sigma = \Phi \Lambda \Phi^T \quad \Phi = X_c \Gamma \]

\[ \frac{1}{n} \left( X_c^T X_c \right) \left( X_c^T X_c \right) = \Gamma \Lambda \Gamma^T \]

► this means that we can obtain PCA by

• 1) assembling \( n^{-1}(X_c^T X_c)(X_c^T X_c) \)
• 2) computing its eigen-decomposition \((\Lambda, \Gamma)\)

► PCA

• the principal components are then given by \( X_c \Gamma \)
• the eigenvalues are given by \( \Lambda \)
Extensions

what is interesting here is that we only need the matrix

\[ K_c = X_c^T X_c = \begin{bmatrix}
- & x_1^c & - \\
\vdots & : & \vdots \\
- & x_n^c & - \\
\end{bmatrix}
\begin{bmatrix}
\vdots \\
x_1^c \\
\vdots \\
x_n^c \\
\end{bmatrix} = \begin{bmatrix}
\vdots \\
(\mathbf{x}_n^c)^T \mathbf{x}_n^c \\
\vdots \\
\end{bmatrix}
\]

this is the matrix of dot-products of the centered data-points

notice that you don’t need the points themselves, only their dot-products (similarities)
Extensions

- to compute PCA, we use the fact that

\[
\frac{1}{n} (X_c^T X_c)(X_c^T X_c) = \frac{1}{n} K_c K_c^T
\]

- but if \( K_c \) has eigendecomposition \((\Lambda, \Gamma)\)

\[
\frac{1}{n} K_c K_c^T = \frac{1}{n} \Gamma \Lambda \Gamma^T \Gamma \Lambda \Gamma^T = \frac{1}{n} \Gamma \Lambda^2 \Gamma^T
\]

- then, \( n^{-1}(X_c^T X_c)(X_c^T X_c) \) has eigendecomposition \((\Lambda^2, \Gamma)\)
Extensions

► in summary, to get PCA
  • 1) compute the dot-product matrix $K$
  • 2) compute its eigen-decomposition $(\Lambda, \Gamma)$

► PCA
  • the principal components are then given by $\Phi = X_c \Gamma$
  • the eigenvalues are given by $\Lambda^2$
  • the projection of the data-points on the principal components is given by

$$X_c^T \Phi = X_c^T X_c \Gamma = K \Gamma$$

► this allows the computation of the eigenvalues and PCA coefficients when we only have access to the dot-product matrix $K$
Any questions?