Principal Component Analysis (PCA)

Salvador Dalí, “Galatea of the Spheres”

CSC411/2515: Machine Learning and Data Mining, Winter 2018
Michael Guerzhoy and Lisa Zhang

Some slides from Derek Hoiem and Alysha Efros
Dimensionality Reduction

- Want to represent data in a new coordinate system with fewer dimensions
  - Cannot easily visualize n-D data, but can plot 2D or 3D data
  - Want to extract features from the data
    - Similar to extracting features with a ConvNet – easier to classify extracted features than original data
  - Want to compress the data while preserving most of the information
- Goal: preserve as much information as we can about the data in the new coordinate system
  - Preserve distance between data points
  - Preserve variation between datapoints

<table>
<thead>
<tr>
<th>Original data</th>
<th>Transformed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1.2)</td>
<td>1.15</td>
</tr>
<tr>
<td>(2, 2)</td>
<td>2</td>
</tr>
<tr>
<td>(3, 3.3)</td>
<td>3.1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Principal Component Analysis

- Linear dimensionality reduction
  - The transformed data is a linear transformation of the original data
- Find a subspace that the data lies in and project the data onto that subspace

![PCA Diagram]

- Original data space
- Component space
Principal Component Analysis

• In the 2D example: want to transform the data so that it varies mostly along $u_1$
  • Can just take the $u_1$ coordinate and keep most of the information about the data
  • Keep the diagram in mind for the rest of the lecture

• We will use a dataset of faces as an example
  • Same idea, but higher-dimensional data
Principal Component Analysis

• If all the data lies in a subspace, we can represent the data using the coordinates in that subspace

• Here: a 1D subspace arguably suffices
  • Just keep information about where in the orange cloud the point is – one dimension suffices
The space of all face images

• When viewed as vectors of pixel values, face images are extremely high-dimensional
  – 100x100 image => 10,000 dimensions

• But very few 10,000-dimensional vectors are valid face images

• We want to effectively model the subspace of face images
  • Need a lot fewer dimensions than 10,000 dimensions for that
The space of faces

- Each images is a point in space
Rotating a Cloud to Be Axis-Aligned

- Consider the covariance matrix of all the points in a cloud
- $\Sigma = \sum_i (x^{(i)} - \mu)(x^{(i)} - \mu)^T$
- The Spectral Theorem says we can diagonalize $\Sigma$ (not covered in detail):
  \[ R^T \Sigma R = D = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}, \]
  R’s columns are the Eigenvectors of $\Sigma$

Spectral theorem:
\[ P^{-1} \Sigma P = D \]
Set $P = R, P^{-1} = R^T$

No need to divide the sum by $n$ when computing the covariance in this context
Now:

\[
\sum_{i} R^T (x^{(i)} - \mu) \left( R^T (x^{(i)} - \mu) \right)^T =
\]

\[
R^T \left( \sum_{i} (x^{(i)} - \mu)(x^{(i)} - \mu)^T \right) R
\]

\[
= R^T \Sigma R = D
\]

So if we rotate the \((x^{(i)} - \mu)\) using \(R^T\), the covariance matrix of the transformed data will be diagonal!
Intuition

• If the covariance of the data is diagonal, the data lies in an axis-aligned cloud

• $R$, the matrix of the eigenvector of $\Sigma$, is the rotation matrix that needs to be applied to the data $(x^{(i)} - \mu)$ to make the cloud of datapoints axis aligned
  
  – Because we proved the covariance of the result will be diagonal
Change of Basis

• (On the board)
• Main points
  – Review of change of basis
  – A rotation around 0 can be interpreted as a change of basis
  – If the data is centred at 0, rotating it means changing its basis
  – We can make data be centred at 0 by subtracting the mean from it
Reconstruction

• For a subspace with the orthonormal basis $V_k = \{v_0, v_1, v_2, \ldots v_k\}$, the best (i.e., closest) reconstruction of $x$ in that subspace is:

$$\hat{x}_k = (x \cdot v_0)v_0 + (x \cdot v_1)v_1 + \ldots + (x \cdot v_k)v_k$$

  – If $x$ is in the span of $V_k$, this is an exact reconstruction
  – If not, this is the projection of $x$ on $V$

• Squared reconstruction error: $|\hat{x}_k - x|^2$
Reconstruction cont’d

• \( \hat{x}_k = (x \cdot v_0)v_0 + (x \cdot v_1)v_1 + \cdots + (x \cdot v_k)v_k \)

• Note: in \( (x \cdot v_0)v_0 \),
  – \( (x \cdot v_0) \) is a measure of how similar \( x \) is to \( v_0 \)
  – The more similar \( x \) is to \( v_0 \), the larger the contribution from \( v_0 \) is to the sum
Representation and reconstruction

- Face $\mathbf{x}$ in “face space” coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^T (\mathbf{x} - \mu), \ldots, \mathbf{u}_k^T (\mathbf{x} - \mu)]$$

$$= w_1, \ldots, w_k$$

- Reconstruction:

$$\hat{\mathbf{x}} = \mathbf{\mu} + w_1 \mathbf{u}_1 + w_2 \mathbf{u}_2 + w_3 \mathbf{u}_3 + w_4 \mathbf{u}_4 + \ldots$$
Reconstruction

After computing eigenfaces using 400 face images from ORL face database
Principal Component Analysis (PCA)

• Suppose the columns of a matrix $X_{N \times K}$ are the datapoints (N is the size of each image, K is the size of the dataset), and we would like to obtain an orthonormal basis of size $k$ that produces the smallest sum of squared reconstruction errors for all the columns of $X - \bar{X}$

  $- \bar{X}$ is the average column of $X$

• Answer: the basis we are looking for is the $k$ eigenvectors of $(X - \bar{X})(X - \bar{X})^T$ that correspond to the $k$ largest eigenvalues
• If $x$ is the datapoint (obtained after subtracting the mean), and $V$ an orthonormal basis, $V^T x$ is a column of the dot products of $x$ and the elements of $x$

• So the reconstruction for the **centered** $x$ is

$$\hat{x} = V(V^T x)$$

• PCA is the procedure of obtaining the $k$ eigenvectors $V_k$
NOTE: centering

- If the image x is *not centred* (i.e., $\bar{X}$ was not subtracted from all the images), the reconstruction is:

$$\hat{x} = \bar{X} + V(V^T(x - \bar{X}))$$
Proof that PCA produces the best reconstruction
DEAR MATH,
I'M NOT A THERAPIST.
SOLVE YOUR OWN PROBLEMS.
Reminder: Intuition for PCA

• The subspace where the reconstruction that will be the best is the major axis of the cloud
• We’ve shown that we are making the cloud axis-aligned by rotating it
  – So we know one of the basis elements will correspond to the best one-dimensional subspace
• The major axis is the Eigenvector which corresponds to the largest Eigenvalue
  – We haven’t shown that (but we could)
Obtaining the Principal Components

• $XX^T$ can be huge
• There are tricks to still compute the Evs
  – Look up Singular Value Decomposition (SVD) if you’re interested
EigenFace as dimensionality reduction

The set of faces is a “subspace” of the set of images

• Suppose it is K dimensional
• We can find the best subspace using PCA
• This is like fitting a “hyper-plane” to the set of (centred) faces
  – spanned by vectors \( \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K \)
  – any face \( \mathbf{x} \approx \bar{\mathbf{x}} + a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \ldots + a_K \mathbf{v}_K \)
Eigenfaces example

Mean: $\mu$

Top eigenvectors: $u_1, \ldots, u_k$
Another Eigenface set
PCA: Applications

convert $x$ into $v_1, v_2$ coordinates

$$x \rightarrow ((x - \bar{x}) \cdot v_1, (x - \bar{x}) \cdot v_2)$$

What does the $v_2$ coordinate measure?
- Distance to line
- Use it for classification—near 0 for orange pts
- Possibly a good feature!

What does the $v_1$ coordinate measure?
- Position along line
- Use it to specify which orange point it is
- Use $v_1$ if want to compress data

$\bar{x}$ is the mean of the orange points
Two views of PCA

- Want to minimize the squared distance between the original data and the reconstructed data, for a given dimensionality of the subspace $k$
  - Minimize the red-green distance per data point
  - Our approach so far
- Maximize the variance of the projected data, for a given dimensionality of the subspace $k$
  - Maximize the “scatter” (i.e., variance) of the green points
  - In general, maximize sum of the variances along all the components
Two views of PCA

- $R^T \Sigma R = D = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}$
- $D$ is the covariance matrix of the transformed data
- The variance of the projected data along the $j$-th component (coordinate) is $\lambda_j = \frac{1}{N} \sum_i (x_j^{(i)} - \bar{x}_j)^2$
- For a given $k$, when maximizing the variance of the projected data, we are trying to maximize
  - $\lambda_1 + \lambda_2 + \cdots + \lambda_k$
- We can show (but won’t) that the variance maximization formulation and minimum square reconstruction error formulation produce the same $k$-dimensional bases
How to choose k?

- If need to visualize the dataset
  - Use $k=2$ or $k=3$
- If want to use PCA for feature extraction
  - Transform the data to be k-dimensional
  - Use cross-validation to select the best $k$
How to choose k?

- Pick based on percentage of variance captured / lost
  - Variance captured: the variance of the projected data
- Pick smallest k that explains some % of variance
  - \( \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_N} \)
- Look for an “elbow” in Scree plot (plot of explained variance or eigenvalues)
  - In practice the plot is never very clean
Limitations of PCA

- Assumption: variance == information
- Assumption: the data lies in a linear subspace only
Autoencoders

- Find the weights that produce as small a difference as possible between the input and the reconstruction
- Train using Backprop
- The code layer is a low-dimensional summary of the input
Autoencoders

- The **Encoder** can be used
  - To compress data
  - As a feature extractor
    - If you have lots of unlabeled data and some labeled data, train an autoencoder on unlabeled data, then a classifier on the features (semi-supervised learning)
- The **Decoder** can be used to generate images given a new code
Autoencoders and PCA

- PCA can be viewed as a special case of an autoencoder
- We can “tie” the encoder and decoder weights to make sure the architecture works the same way PCA does
  - This is sometimes useful for regularization

![Autoencoder Diagram]

- Input: $x$
- Hidden layer: $W^T x$
- Reconstruction: $WW^T x$
- Minimize $\sum_i |WW^T x - x|^2$ -- the square reconstruction error, as in PCA
- Just one set of weight parameters, transposed