CSC 411: Lecture 14: Principal Components Analysis & Autoencoders

Raquel Urtasun & Rich Zemel

University of Toronto

Nov 4, 2015
Today

- Dimensionality Reduction
- PCA
- Autoencoders
One problem with mixture models: each observation assumed to come from one of \( K \) prototypes.

Constraint that only one active (responsibilities sum to one) limits representational power.

Alternative: Distributed representation, with several latent variables relevant to each observation.

Can be several binary/discrete variables, or continuous.
Example: continuous underlying variables

- What are the intrinsic latent dimensions in these two datasets?

- How can we find these dimensions from the data?
PCA: most popular instance of second main class of unsupervised learning methods, projection methods, aka dimensionality-reduction methods.

Aim: find small number of "directions" in input space that explain variation in input data; re-represent data by projecting along those directions.

Important assumption: variation contains information.

Data is assumed to be continuous: linear relationship between data and learned representation.
PCA: Common tool

- Handles high-dimensional data – if data has thousands of dimensions, can be difficult for classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization
  - Preprocessing
  - Modeling – prior for new data
  - Compression
PCA: Intuition

- Assume start with $N$ data vectors, of dimensionality $D$
- Aim to reduce dimensionality – linearly project (multiply by matrix) to much lower dimensional space, $M << D$
- Search for orthogonal directions in space w/ highest variance – project data onto this subspace
- Structure of data vectors is encoded in sample covariance
Finding principal components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)

- Calculate the empirical covariance matrix:

$$C = \frac{1}{N} \sum_{n=1}^{N} (x^{(n)} - \bar{x})(x^{(n)} - \bar{x})^T$$

  with $\bar{x}$ the mean

- What’s the dimensionality of $x$?

- Find the $M$ eigenvectors with largest eigenvalues of $C$: these are the principal components

- Assemble these eigenvectors into a $D \times M$ matrix $U$

- We can now express $D$-dimensional vectors $x$ by projecting them to $M$-dimensional $z$

$$z = U^T x$$
Standard PCA

- Algorithm: to find M components underlying D-dimensional data
  1. Select the top M eigenvectors of C (data covariance matrix):

\[
C = \frac{1}{N} \sum_{n=1}^{N} (x^{(n)} - \bar{x})(x^{(n)} - \bar{x})^T = U\Sigma U^T \approx U\Sigma_{1:M} U_{1:M}^T
\]

where \( U \): orthogonal, columns = unit-length eigenvectors
\[ U^T U = UU^T = 1 \]

and \( \Sigma \): matrix with eigenvalues in diagonal = variance in direction of eigenvector

2. Project each input vector \( x \) into this subspace, e.g.,

\[
z_j = u_j^T x; \quad z = U_{1:M}^T x
\]
Two Derivations of PCA

- Two views/derivations:
  - Maximize variance (scatter of green points)
  - Minimize error (red-green distance per datapoint)
PCA: Minimizing Reconstruction Error

- We can think of PCA as projecting the data onto a lower-dimensional subspace.

- One derivation is that we want to find the projection such that the best linear reconstruction of the data is as close as possible to the original data.

\[ J = \sum_n ||x^{(n)} - \tilde{x}^{(n)}||^2 \]

where

\[ \tilde{x}^{(n)} = \sum_{j=1}^M z_j^{(n)} u_j + \sum_{j=M+1}^D b_j u_j \]

Add somewhere the derivation of this.

- Objective minimized when first M components are the eigenvalues with the maximal eigenvectors.

\[ z_j^{(n)} = (x^{(n)})^T u_j; \quad b_j = \bar{x}^T u_j \]
Applying PCA to faces

- Run PCA on 2429 19x19 grayscale images (CBCL data)
- Compresses the data: can get good reconstructions with only 3 components

![Example images of PCA projections]

- PCA for pre-processing: can apply classifier to latent representation – PCA w/ 3 components obtains 79% accuracy on face/non-face discrimination in test data vs. 76.8% for m.o.G with 84 states
- Can also be good for visualization
Applying PCA to faces: Learned basis

Show maybe Vetter 3D faces, or other examples
Applying PCA to digits

- Original images of digit 3
- Reconstructed images with different bases:
  - 2 bases
  - 10 bases
  - 100 bases
  - 506 bases
- Mean image
- Principal bases 1, 2, 3
PCA is closely related to a particular form of neural network.

An autoencoder is a neural network whose outputs are its own inputs.

The goal is to minimize reconstruction error.
Autoencoders

- Define
  \[ z = f(Wx); \quad \hat{x} = g(Vz) \]

- Goal:
  \[ \min_{W,V} \frac{1}{2N} \sum_{n=1}^{N} \|x^{(n)} - \hat{x}^{(n)}\|^2 \]

- If \( g \) is linear
  \[ \min_{W,V} \frac{1}{2N} \sum_{n=1}^{N} \|x^{(n)} - VWx^{(n)}\|^2 \]

- In other words, the optimal solution is PCA.
Autoencoders: Nonlinear PCA

- What if \( g() \) is not linear?
- Then we are basically doing nonlinear PCA
- Some subtleties but in general this is an accurate description
Comparing Reconstructions

Real data
30-d deep autoencoder
30-d logistic PCA
30-d PCA