Overview

- Curse of dimensionality
- Different ways to reduce dimensionality
- Principal Components Analysis (PCA)
- Example: Eigen Faces
- PCA for classification
- Witten & Frank section 7.3
  - only the PCA section required

True vs. observed dimensionality

- Get a population, predict some property
  - instances represented as (useful, height) pairs
  - what is the dimensionality of this data?

- Data points over time from different geographic areas over time:
  - $X_1$: # of skiing accidents
  - $X_2$: # of fire hydrant plumes
  - $X_3$: snow-snow expenditures
  - $X_4$: # of school closings
  - $X_5$: # patients with heart stroke
  - Temperature?

Curse of dimensionality (2)

- Machine learning methods are statistical in nature
  - count observations in various regions of some space
  - use counts to construct the predictor (fix)
  - e.g., decision trees: $p(X | rain = strong, T > 28)$
  - text: #documents in "hp" and "3d" and not "s" and ...

- As dimensionality grows: fewer observations per region
  - 3d: 3 regions, 2d: 32 regions, 1000d = hopeless
  - statistics need repetition
    - flip a coin once $\Rightarrow$ head
    - P(head) = 100%?

Dealing with high dimensionality

- Use domain knowledge
  - feature engineering: SIFT, MFCC

- Make assumption about dimensions
  - independence: count along each dimension separately
  - smoothness: propagate class counts to neighboring regions
  - symmetry: e.g., invariance to order of dimensions: $x_1 \rightarrow x_k$

- Reduce the dimensionality of the data
  - create a new set of dimensions (variables)

Dimensionality reduction

- Goal: represent instances with fewer variables
  - try to preserve as much structure in the data as possible
  - discriminative: only structure that affects class separability

- Feature selection
  - pick a subset of the original dimensions $X_t, X_1, \ldots X_d, X_n$
  - discriminatively pick good class "predictors" (e.g., gain)

- Feature extraction
  - construct a new set of dimensions $E_1, E_2, \ldots E_m$
  - (linear) combinations of original $X_1, X_2, X_3, \ldots X_n$

Principal Components Analysis

- Defines a set of principal components
  - 1st: direction of the greatest variability in the data
  - $2^{nd}$, perpendicular to 1st, greatest variability of what's left
  - ... and so on until d (original dimensionality)

- First $m<<d$ components become $m$ new dimensions
  - change coordinates of every data point to these dimensions

Why greatest variability?

- Example: reduce 2-dimensional data to 1-d
  - $(x_1, x_2) \rightarrow e$ (along new axis $e$)
- Pick $e$ to maximize variability
- Reduces cases when two points are close in e-space but very far in (x,y)-space
- Minimizes distances between original points and their projections
Principal components

- “Center” the data at zero: \( \mathbf{x}_n = \mathbf{x}_n - \mu \)
- Subtract mean from each attribute
- Compute covariance matrix \( \mathbf{X} \)
  - Covariance of dimensions \( x_1 \) and \( x_2 \):
    - If \( x_i \) and \( x_j \) tend to increase together:
    - Or does \( x_i \) decrease as \( x_j \) increases?
- Multiply a vector by \( \Sigma \) to get \( \Sigma \mathbf{v} \) again
  - Turns towards direction of variance
- Want vectors \( \mathbf{v} \) which aren’t turned: \( \Sigma \mathbf{v} = \lambda \mathbf{v} \)
  - \( \mathbf{v} \) eigenvectors of \( \Sigma \)
  - \( \lambda \) corresponding eigenvalues
  - Principal components = eigenvectors w. largest eigenvalues

Finding Principal Components

1. Find eigenvalues by solving: \( \det(\Sigma - \lambda I) = 0 \)
   \[
   \begin{vmatrix}
   2.0 - \lambda & 0.8 \\
   0.8 & 0.6 - \lambda
   \end{vmatrix} = 0
   \]
   \[
   (2.0 - \lambda)(0.6 - \lambda) - (0.8)(0.8) = 0
   \]
   \[
   \lambda = 2.64, 0.56
   \]

2. Find ith eigenvector by solving: \( \Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i \)
   \[
   \begin{pmatrix}
   2.0 & 0.8 \\
   0.8 & 0.6
   \end{pmatrix}
   \begin{pmatrix}
   v_{11} \\
   v_{12}
   \end{pmatrix} =
   \begin{pmatrix}
   2.64 & 0.8 \\
   0.8 & 0.56
   \end{pmatrix}
   \begin{pmatrix}
   v_{21} \\
   v_{22}
   \end{pmatrix}
   \]
   \[
   \begin{pmatrix}
   2.0v_{11} + 0.8v_{12} = 2.64v_{21} + 0.8v_{22} \\
   0.8v_{11} + 0.6v_{12} = 0.8v_{21} + 0.56v_{22}
   \end{pmatrix}
   \]
   \[
   v_1 = \begin{pmatrix}
   22 \\
   1
   \end{pmatrix}
   \]

3. 1st PC: \( [0.91, 0.41] \), 2nd PC: \( [-0.41, 0.91] \)

Direction of greatest variability

- Select dimension \( \mathbf{e} \) which maximizes the variance
- Points \( \mathbf{x} \) “projected” onto vector \( \mathbf{e} \)
- Variance of projections: \( \sum (x_i - \bar{x})^2 = \sum \mathbf{v}_i \mathbf{e}_i \)
- Maximize variance
  - Want unit length: \( ||\mathbf{e}|| = 1 \)
  - Add Lagrange multiplier
    \[ \mathbf{e}^T \Sigma \mathbf{e} = \lambda \]
    \[ \lambda \mathbf{e} = \Sigma \mathbf{e} \]
    \[ \mathbf{e} \] is an eigenvector of \( \Sigma \)

Variance along eigenvector

\[ \mathbf{V} = \frac{1}{n} \sum (x_i - \bar{x}) (x_i - \bar{x})^T = \mathbf{E}^T \sigma \mathbf{E} \]

PCA in a nutshell

1. Correlated high-d data
2. Center the points
3. Compute covariance matrix
4. Compute eigenvectors + eigenvalues
5. Select top \( k \) eigenvectors
6. Project data points to those eigenvectors
7. Un-correlated low-d data

PCA example: Eigen Faces

Input: data set of \( N \) face images

Face: \( K \times K \) bitmap of pixels

"Unfold" each bitmap to \( K \)-dimensional vector

Arrange in a matrix:
  \( \text{each face = column} \)

Matlab demo on course webpage

Eigen Faces: Projection

- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?
(Eigen) Face Recognition
- Face similarity
  - in the reduced space
  - insensitive to lighting, expression, orientation
- Projecting new "faces"
  - everything is a face

new face
projected to eigenfaces

Linear Discriminant Analysis
- LDA: pick a new dimension that gives:
  - maximum separation between means of projected classes
  - minimum variance within each projected class
- Solution: eigenvectors based on between-class and within-class covariance matrices

PCA: practical issues
- Covariance extremely sensitive to large values
  - multiply some dimension by 1000
  - becomes a principal component
- normalize each dimension to zero mean and unit variance:
  \[ \mathbf{x}' = (\mathbf{x} - \text{mean}) / \text{std dev} \]
- PCA assumes underlying subspace is linear
  - 1d: straight line
  - 2d: flat sheet
  - transform to handle non-linear spaces (manifolds)

PCA and classification
- PCA is unsupervised
  - maximizes overall variance of the data along a small set of directions
  - does not know anything about class labels
  - can pick direction that makes it hard to separate classes
- Discriminative approach
  - look for a dimension that makes it easy to separate classes

PCA vs. LDA
- LDA not guaranteed to be better for classification
  - assumes classes are unimodal Gaussians
  - fails when discriminatory information is not in the mean, but in the variance of the data
- Example where PCA gives a better projection:

Dimensionality reduction
- Pros
  - reflects our intuitions about the data
  - allows estimating probabilities in high-dimensional data
    - no need to assume independence etc.
  - dramatic reduction in size of data
    - faster processing (as long as reduction is fast), smaller storage
- Cons
  - too expensive for many applications (Twitter, web)
  - disastrous for tasks with fine-grained classes
  - understand assumptions behind the methods (linearity etc.)
  - there may be better ways to deal with sparseness

Summary
- True dimensionality \( \ll \) observed dimensionality
- High dimensionality \( \Rightarrow \) sparse, unstable estimates
- Dealing with high dimensionality:
  - use domain knowledge
  - make an assumption: independence / smoothness / symmetry
  - dimensionality reduction: feature selection / feature extraction
- Principal Components Analysis (PCA)
  - picks dimensions that maximize variability
  - eigenvectors of the covariance matrix
  - examples: Eigen Faces
  - variant for classification: Linear Discriminant Analysis