CS 60050
Machine Learning

Dimensionality Reduction

Some slides taken from course materials of Jure Leskovec

Dimensionality

- Dimensionality = number of features or attributes in the data set
- Data can have really large number of features
  - In a corpus of text documents, each distinct word can be a feature (bag of words model)
  - In an image data set, each of 1024 x 768 pixels can be a feature

Dimensionality reduction

- Goal (informal): reduce the number of features, such that information loss is not much
- Ultimate goal – good performance in clustering, classification, etc.

Why dimensionality reduction?

- Feature space may be very sparsely populated
  - E.g., in case of a text document corpus, each individual word may be contained in a very small subset of the corpus
- ML models do not perform well on such sparse feature space
  - ML models statistical in nature – counts observations in various regions of feature space
  - As dimensionality grow, fewer observations per region
  - Curse of dimensionality - number of training examples required increases exponentially with dimensionality

Why dimensionality reduction?

- Some other reasons:
  - Some features may be irrelevant, or redundant (e.g., highly correlated with other features)
  - We want to visualize high dimensional data

Intuition behind dimensionality reduction

- Dimensionality reduction = changing the feature space in which the points lie (to a lower dimensional space)

- What should be the desirable properties of the reduced feature set?

Ways of dimensionality reduction

- Two broad ways of reducing dimensionality
  - (1) Select a subset of the given features
    - E.g., for spam email classification: time of day when the email comes vs. number of spam-words
    - Commonly known as feature selection
  - (2) Define a new set of features that is smaller than the given feature set
    - E.g., given marks of students in 8 subjects (Physics, Chem, Maths, English, Hindi, History, Geography, Pol. Sc.), maybe most variation can be captured considering three (new) dimensions – Science, Social Science, Arts
    - Commonly known as feature extraction

Ways of dimensionality reduction

- Supervised
  - These methods use both the feature values as well as the class labels of the data points
- Unsupervised
  - These methods use only the feature values, not the class values
- Domain-specific
  - E.g. Text:
    - Remove stop-words (and, a, the, …)
    - Stemming (going → go, Tom's → Tom, …)
    - Select important words based on document frequency

Supervised feature selection

- Usually selects a subset of the original features
- Score each feature based on some suitable mechanism (see next slide)

- Forward/Backward elimination
  - Choose the feature with the highest/lowest score
  - Re-score other features
  - Repeat
- If you have lots of features (like in text)
  - Just select top K scored features

Supervised feature selection: some ways to score features

- Mutual information between feature & class
  - Mutual info: a measure between two (possibly multi-dimensional) random variables, that quantifies the amount of information obtained about one random variable, through the other random variable.
- χ^2 independence between feature & class
  - Test whether the occurrence of a specific feature value and the occurrence of a specific class are independent
- Ablation: How classification accuracy varies if a feature is removed

See references for some pointers

Unsupervised feature selection

- Differs from supervised feature selection in two ways:
  - Instead of choosing subset of original features, create new features (dimensions) defined as functions over all original features
  - Do not consider class labels, just the data points

Unsupervised feature selection

- Idea:
  - Given data points in N-dimensional space,
  - Project into lower dimensional space while preserving as much information as possible
    - E.g., find best planar approximation to 3D data
    - E.g., find best planar approximation to 104D data
  - In particular, choose projection that minimizes the squared error in reconstructing original data – PCA

Principal Component Analysis (PCA)

PCA: overview

- Say we have a N-dimensional feature space
- We wish to reduce to K dimensions, K << N
- Dimensionality reduction implies information loss; PCA preserves as much information as possible by minimizing the reconstruction error:

$$\|x - \hat{x}\|$$

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$$\Downarrow$$

$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

PCA: overview

- PCA transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components
- The first principal component
  - Direction of the greatest variability of the data
  - Accounts for as much of the variability in the data as possible
- The second principal component
  - Perpendicular / orthogonal to the first, captures greatest variability of what is left
- Each succeeding component accounts for as much of the remaining variability as possible

PCA: overview

- Number of principal components = number of original dimensions = N
- We can choose the first K<<N of the principal components as the new dimensions
- Change coordinates of every data point to these new dimensions (project a point to each new dimension)

How PCA finds suitable dimensions?

- The data is first “centered” to zero or the origin, by subtracting the mean from each attribute

- PCA then minimizes the distances between the original points and their projections, across all points
- Equivalent to maximizing the distances between the origin and the projections, across all points

Background concepts

- Covariance
  - An indication of whether two variables/attributes change together, or change in opposite directions

- Positive covariance between two variables => if one increases (decreases), the other tends to increase (decrease) as well

Background concepts

- Given an N x M data matrix D,
  - whose N rows are the data points, and
  - whose M columns are the features/attributes
- The covariance matrix C of D is a M x M matrix which has entries $c_{ij} = covariance(d_{*i}, d_{*j})$
  - c_{ij} is the covariance of the i-th and j-th attributes (columns) of the data matrix
  - c_{ij} measures how strongly the attributes vary together
  - If i=j, then the covariance is the variance of the attribute.

Covariance matrix considering 5 features a, b, c, d, e

$$\begin{bmatrix} V_a & C_{a,b} & C_{a,c} & C_{a,d} & C_{a,e} \\ C_{a,b} & V_b & C_{b,c} & C_{b,d} & C_{b,e} \\ C_{a,c} & C_{b,c} & V_c & C_{c,d} & C_{c,e} \\ C_{a,d} & C_{b,d} & C_{c,d} & V_d & C_{d,e} \\ C_{a,e} & C_{b,e} & C_{c,e} & C_{d,e} & V_e \end{bmatrix}$$

Covariance matrix: another representation (μ_1, μ_2 are the mean of attributes a, b, …)

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}.$$

Background concepts

- Given some data points (vectors in feature space), we computed the covariance matrix C
- A property of C: if any of the vectors is multiplied by C, the vector is rotated towards the direction of greatest variability of the data
- So, what is the direction of greatest variability of data?
- Hint: a vector that is already in the direction of greatest variability will <u>not</u> be rotated when multiplied by C

Background concepts

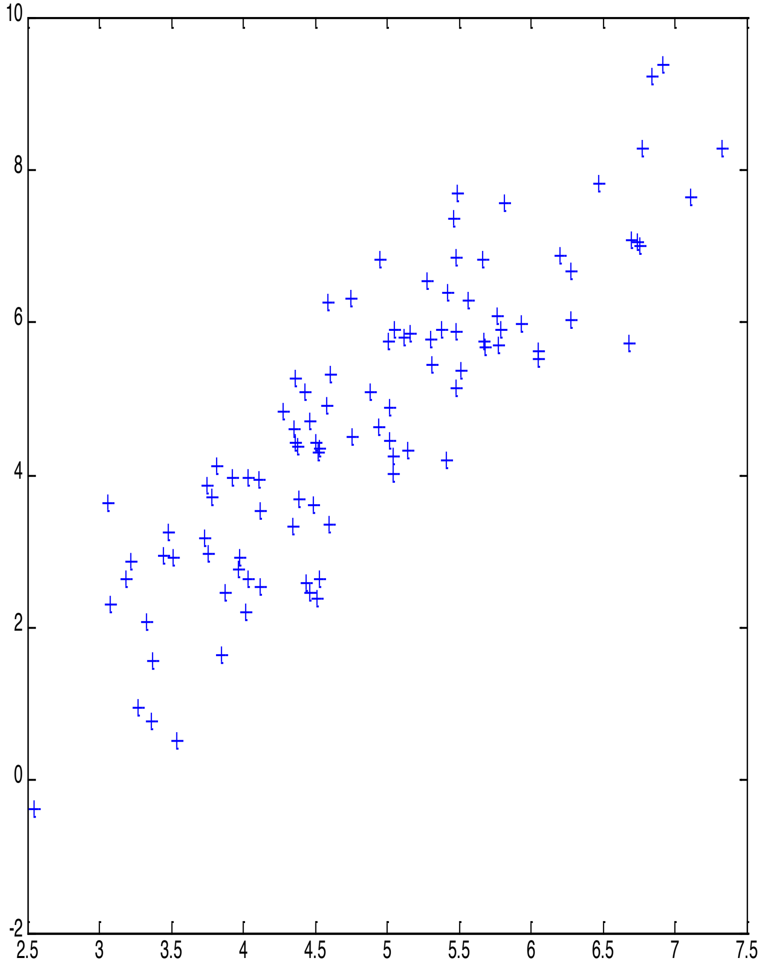
- Given some data points (vectors in feature space), we computed the covariance matrix C
- A property of C: if any of the vectors is multiplied by C, the vector is rotated towards the direction of greatest variability of the data

- Vectors that are not rotated when multiplied by C (magnitude of vector may change, but direction does not change) → eigenvectors of C

Background concepts

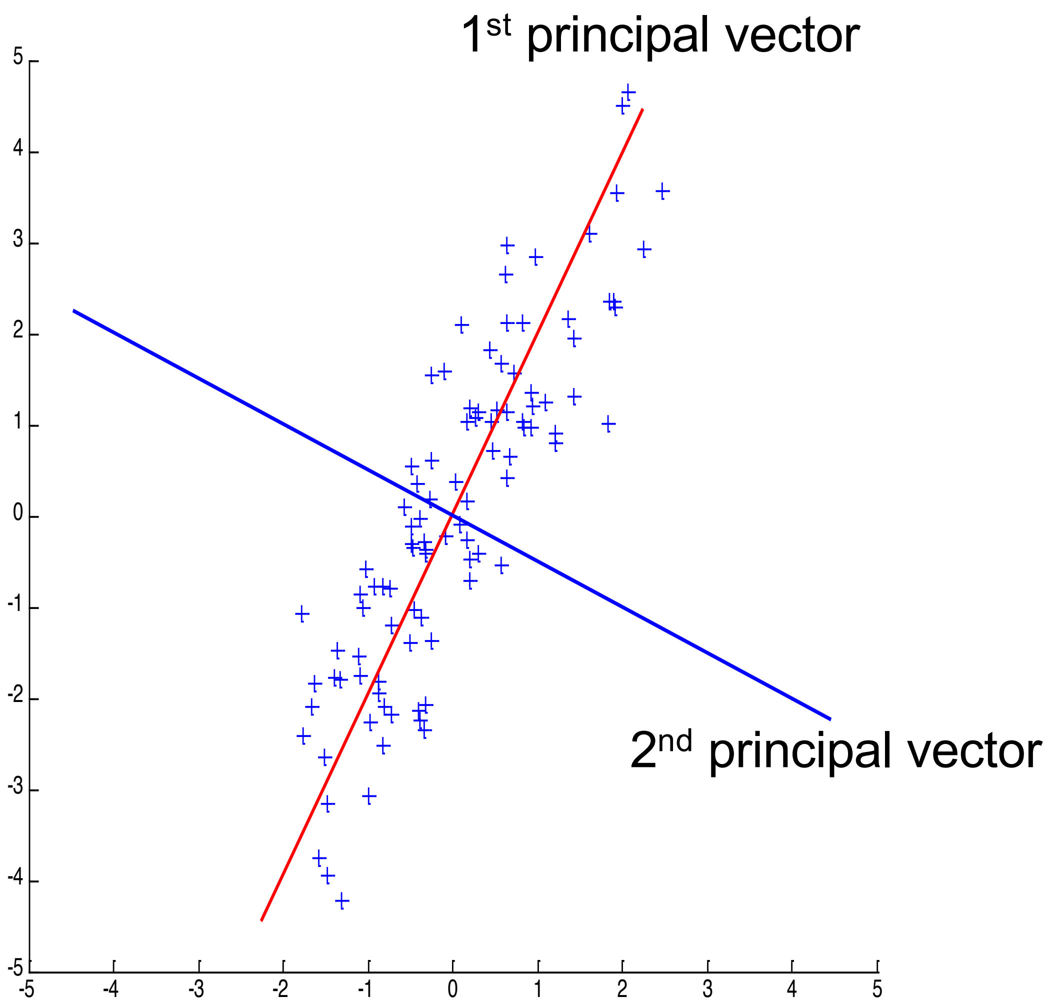
- If the data matrix D is preprocessed so that the mean of each attribute is zero, then $C = D^T D$
- Covariance matrices are examples of positive semidefinite matrices, which have non-negative eigenvalues
  - Eigenvalues of C can be ordered in decreasing order of magnitude
  - Eigenvectors of C can be ordered so that the i-th eigenvector corresponds to i-th largest eigenvalue

Geometric interpretation on 2d data



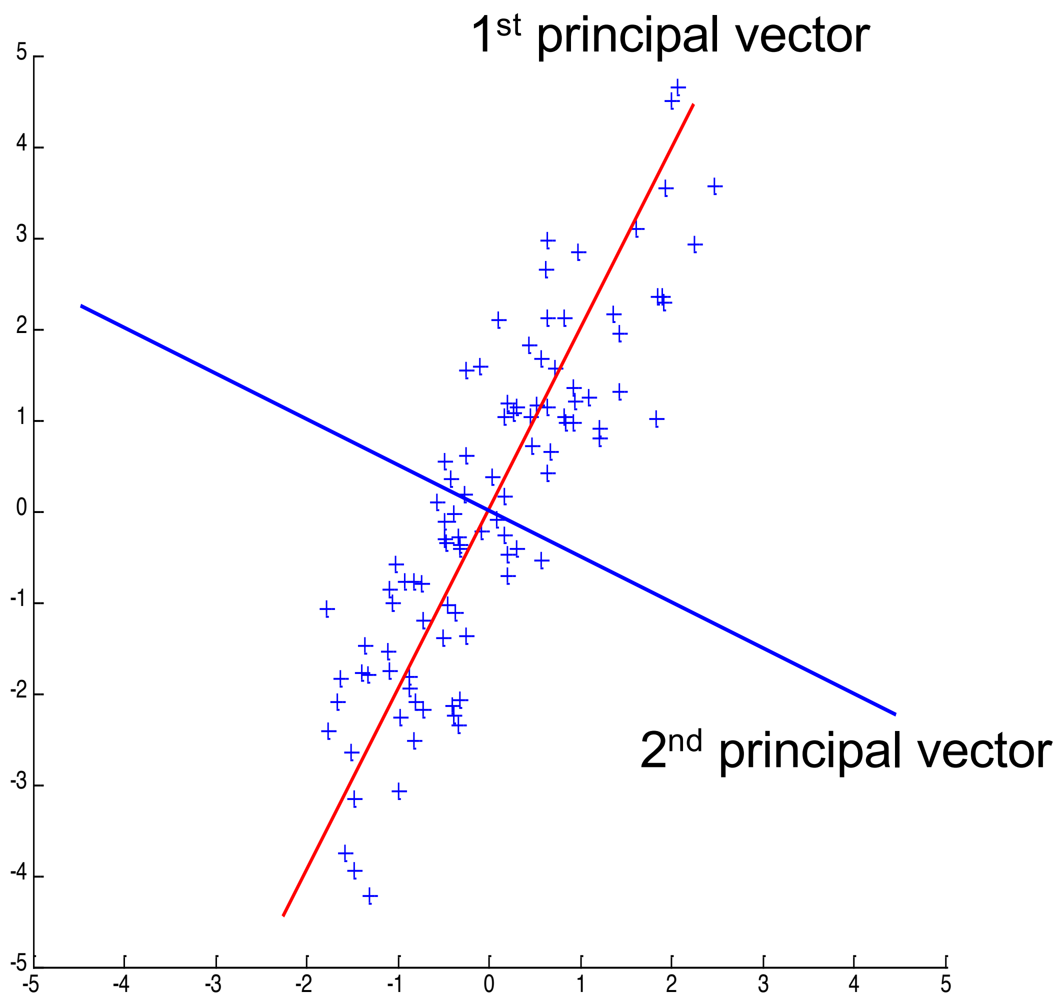
Geometric interpretation on 2d data

- PCA projects the data along the directions where the data varies most
- A rotation of the coordinate system such that the axes show a maximum of variation (covariance) along their directions.
- The directions are orthogonal to each other – these are the new attributes (PCs)



Geometric interpretation on 2d data

- These directions are determined by some of the eigenvectors of the covariance matrix of data
  - Specifically, those eigenvectors that correspond to the largest eigenvalues
  - Magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions
- Each new attribute is a linear combination of the original attributes



PCA - Steps

Suppose $x_1, x_2, \ldots, x_M$ are N x 1 vectors

Step 1: $\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i$

Step 2: subtract the mean: $\Phi_i = x_i - \bar{x}$ (i.e., center at zero)

Step 3: form the matrix $A = [\Phi_1 \ \Phi_2 \cdots \Phi_M]$ (NxM matrix), then compute:

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = \frac{1}{M} A A^T$$

(sample **covariance** matrix, NxN, characterizes the *scatter* of the data)

Step 4: compute the eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$

Step 5: compute the eigenvectors of C: $u_1, u_2, \ldots, u_N$

PCA - Steps

an orthogonal basis

- Since C is symmetric, $u_1, u_2, \ldots, u_N$ form ~~a basis~~, (i.e., any vector x or actually $(x - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$x - \bar{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i \quad \text{where } b_i = \frac{(x - \bar{x}).u_i}{(u_i.u_i)}$$

Step 6: <u>(**dimensionality reduction step**)</u> keep only the terms corresponding to the K largest eigenvalues:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ where } K << N$$

- The representation of $\hat{x} - \bar{x}$ into the basis $u_1, u_2, \ldots, u_K$ is thus

$$\begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_K \end{bmatrix}$$

How to choose K?

- Choose *K* using the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$

- In this case, we say that we “preserve” 90% or 95% of the information (variance) in the data.

- If K=N, then we “preserve” 100% of the information in the data.

Error due to dimensionality reduction

- The original vector *x* can be reconstructed using its principal components:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ or } \hat{x} = \sum_{i=1}^{K} b_i u_i + \bar{x}$$

- PCA minimizes the reconstruction error:

$$e = ||x - \hat{x}||$$

- It can be shown that the reconstruction error is:

$$e = 1/2 \sum_{i=K+1}^{N} \lambda_i$$

Normalization

- The principal components are dependent on the *units* used to measure the original variables as well as on the *range* of values they assume.

- Data should always be normalized prior to using PCA.

- A common normalization method is to transform all the data to have zero mean and unit standard deviation:

$$\frac{x_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \quad (\boldsymbol{\mu} \text{ and } \boldsymbol{\sigma} \text{ are the mean and standard deviation of } x_i\text{'s})$$

Benefits of PCA

- Identify the strongest patterns in the data in an unsupervised way
- Capture most of the variability of the data by a small fraction of the total set of dimensions
- Eliminate much of the noise in the data, making it beneficial for classification and other learning algorithms

Problems and limitations

- What if very large dimensional data?
  - e.g., Images ($d \geq 10^4$)
- Problem:
  - Covariance matrix Σ is size (d^2)
  - $d=10^4 \rightarrow |\Sigma| = 10^8$
- Singular Value Decomposition (SVD)
  - efficient algorithms available
  - some implementations find just top N eigenvectors

References

- Mutual information-based feature selection
  https://thuijskens.github.io/2017/10/07/feature-selection/

- A Gentle Introduction to the Chi-Squared Test for Machine Learning
  https://machinelearningmastery.com/chi-squared-test-for-machine-learning/

- Feature Selection For Machine Learning in Python
  https://machinelearningmastery.com/feature-selection-machine-learning-python/