CS 2750 Machine Learning
Lecture 20

Dimensionality reduction
Feature selection

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Dimensionality reduction. Motivation.

- **Is there a lower dimensional representation of the data that captures well its characteristics?**
- **Assume:**
  - We have an data $\{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N\}$ such that
    $\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$
  - Assume the dimension d of the data point $\boldsymbol{x}$ is very large
  - We want to analyze $\boldsymbol{x}$
- **Methods of analysis are sensitive to the dimensionality *d***
- **Our goal: Find a lower dimensional representation of data**
- **Two learning problems:**
  - **supervised**
  - **unsupervised**

Dimensionality reduction for classification

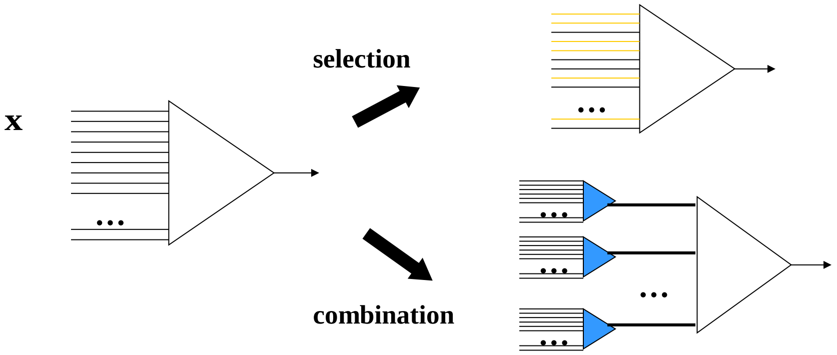
- **Classification problem example:**
  - We have an input data $\{\mathbf{x_1}, \mathbf{x_2}, .., \mathbf{x_N}\}$ such that

    $$\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$$

    and a set of corresponding output labels $\{y_1, y_2, .., y_N\}$
  - Assume the dimension *d* of the data point ***x*** is very large
  - We want to classify ***x***
- **Problems with high dimensional input vectors**
  - **A large number of parameters** to learn, if a dataset is small this can result in:
    - Large variance of estimates and overfit
  - **it becomes hard to explain what features are important in the model (**too many choices some can be substitutable**)**

Dimensionality reduction

- **Solutions**:
  - **Selection of a smaller subset** of inputs (features) from a large set of inputs; train classifier on the reduced input set
  - **Combination of high dimensional inputs** to a smaller set of features $\phi_k(\mathbf{x})$; train classifier on new features



Feature selection

How to find a good subset of inputs/features?

- **We need**:
  - A criterion for ranking good inputs/features
  - Search procedure for finding a good set of features
- **Feature selection process can be**:
  - **Dependent on the learning task**
    - e.g. classification
    - Selection of features affected by what we want to predict
  - **Independent of the learning task**
    - Unsupervised methods
    - may lack the accuracy for classification/regression tasks

Task-dependent feature selection

Assume:

- **Classification problem**:
  - $\mathbf{x}$ – input vector, y - output

Objective: Find a subset of inputs/features that gives/preserves most of the output prediction capabilities

Selection approaches:

- **Filtering approaches**
  - Filter out features with small predictive potential
  - done before classification; typically uses univariate analysis
- **Wrapper approaches**
  - Select features that directly optimize the accuracy of the multivariate classifier
- **Embedded methods**
  - Feature selection and learning closely tied in the method

Feature selection through filtering

- **Assume:**
  - **Classification problem**: **x** – input vector, *y* - output
  - Inputs in x or some fixed feature mappings $\phi_k(\mathbf{x})$
- **How to select the feature:**
  - **Univariate analysis**
    - Pretend that only one variable, x_k, exists
    - See how well it predicts the output *y* alone
  - **Example:**
    - differentially expressed features (or inputs)
    - Good separation in binary (case/control settings)

Differentially expressed features

- **Scores for measuring the differential expression**
  - **T-Test score** (Baldi & Long)
    - Based on the test that two groups come from the same population
  - **Fisher Score** $Fisher(i) = \dfrac{\mu_i^{(+)2} - \mu_i^{(-)2}}{\sigma_i^{(+)2} + \sigma_i^{(-)2}}$
  - **AUROC score:** Area under Receiver Operating Characteristic curve

Problems:

- if many random features, and not many instances we can learn from the features with a good differentially expressed score must arise
- Techniques to reduce **FDR** (False discovery rate) and **FWER** (Family wise error).

Feature filtering

Other univariate scores:

- **Correlation coefficients** $$\rho(\phi_k, y) = \frac{Cov(\phi_k, y)}{\sqrt{Var(\phi_k) Var(y)}}$$
  - Measures **linear dependences**
- **Mutual information**

$$I(\phi_k, y) = \sum_i \sum_j \tilde{P}(\phi_k = j, y = i) \log_2 \frac{\tilde{P}(\phi_k = j, y = i)}{\tilde{P}(\phi_k = j)\tilde{P}(y = i)}$$

- **Univariate assumptions:**
  - Only one feature and its effect on *y* is incorporated in the mutual information score
  - Effects of two features on *y* are independent
- What to do if the combination of features gives the best prediction?

Feature selection: dependent features

Filtering with dependent features

- Let $\boldsymbol{\varphi}$ be a current set of features (starting from complete set)
- We can remove feature $\phi_k(\mathbf{x})$ from it when:
  $\tilde{P}(y \mid \boldsymbol{\varphi} \setminus \phi_k) \approx \tilde{P}(y \mid \boldsymbol{\varphi})$ for all values of ϕ_k, y
- Repeat removals until the probabilities differ.

Problem: how to compute/estimate $\tilde{P}(y \mid \boldsymbol{\varphi} \setminus \phi_k), \tilde{P}(y \mid \boldsymbol{\varphi})$?

Solution: make some simplifying assumption about the underlying probabilistic model

- **Example: use a Naïve Bayes**
- **Advantage:** speed, modularity, applied before classification
- **Disadvantage:** may not be as accurate

Feature selection: wrappers

Wrapper approach:

- The feature selection is driven by the prediction accuracy of the classifier (regressor) we actually want to built

How to find the appropriate feature set?

- **For d binary features there are 2^d different feature subsets**
- **Idea: Greedy search in the space of classifiers**
  - Gradually add features improving most the quality score
  - Gradually remove features that effect the accuracy the least
  - Score should reflect the accuracy of the classifier (error) and also prevent overfit
- **Standard way to measure the quality:**
  - Internal cross-validation (m-fold cross validation)

Internal cross-validation

- **Split train set: to internal train and test sets**
- **Internal train set: train different models** (defined e.g. on different subsets of features)
- **Internal test set/s**: **estimate the generalization error** and select the best model among possible models
- **Internal cross-validation (*m*-fold):**
  - Divide the train data into *m* equal partitions (of size *N/m*)
  - Hold out one partition for validation, train the classifiers on the rest of data
  - Repeat such that every partition is held out once
  - The estimate of the generalization error of the learner is the mean of errors of on all partitions

Feature selection: wrappers

- **Greedy (forward) search:**
  - logistic regression model with features

Start with $p(y=1|\mathbf{x},\mathbf{w}) = g(w_o)$

Choose feature x_i with the best error (in the internal step)

$$p(y=1|\mathbf{x},\mathbf{w}) = g(w_o + w_i x_i)$$

Choose feature x_j with the best error (in the internal step)

$$p(y=1|\mathbf{x},\mathbf{w}) = g(w_o + w_i x_i + w_j x_j)$$

Etc.

When to stop ?

Goal: Stop adding features when the error on the data stops descreasing

Embedded methods

- **Feature selection + classification model learning** done together
- **Embedded models:**
  - **Regularized models**
    - Models of higher complexity are explicitly penalized leading to 'virtual' removal of inputs from the model
    - Regularized logistic/linear regression
    - **Support vector machines**
      - Optimization of margins penalizes nonzero weights
  - **CART/Decision trees**

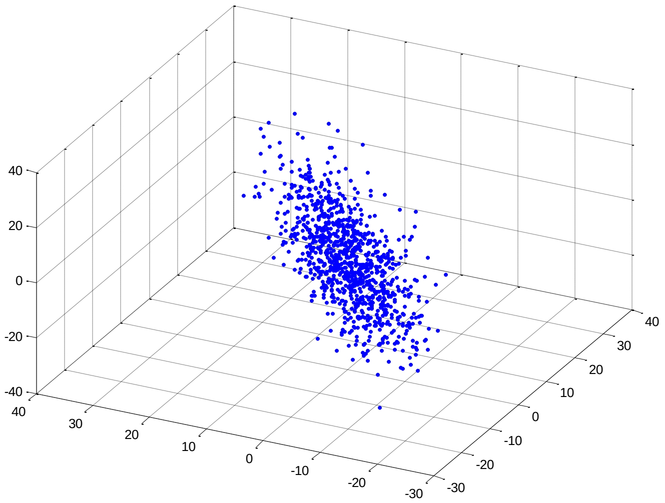
Dimensionality reduction

- **Is there a lower dimensional representation of the data that captures well its characteristics?**
- **Assume:**
  - We have an data $\{\mathbf{x_1}, \mathbf{x_2}, .., \mathbf{x_N}\}$ such that
    $\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$
  - Assume the dimension *d* of the data point ***x*** is very large
  - We want to analyze ***x***
- **Methods of analysis are sensitive to the dimensionality *d***
- **Our goal:**
  - **Find a lower dimensional representation of data of dimension *d'* < *d***

Principal component analysis (PCA)

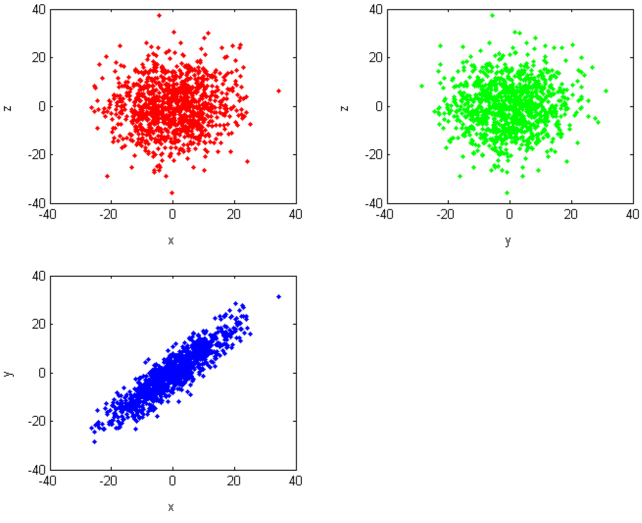
- **Objective:** We want to replace a high dimensional input with a small set of features (obtained by combining inputs)
  - Different from the feature subset selection !!!
- **PCA:**
  - A linear transformation of *d* dimensional input *x* to M dimensional feature vector *z* such that $M < d$ under which the retained variance is maximal.
  - Equivalently it is the linear projection for which the sum of squares reconstruction cost is minimized.

PCA



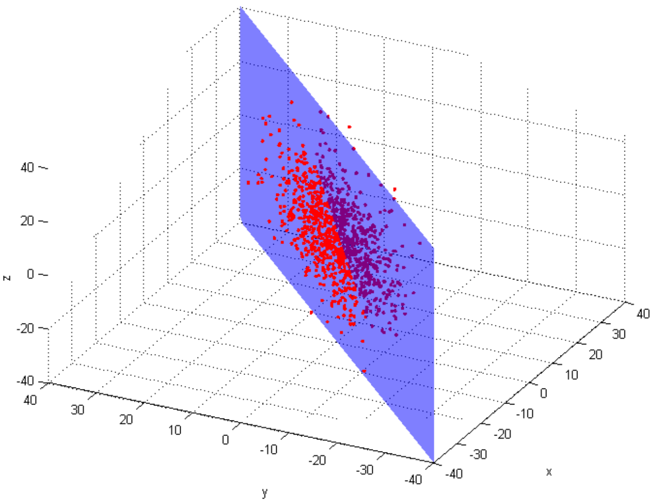
40
20
0
-20
-40
40
30
20
10
0
-10
-20
-30
40
30
20
10
0
-10
-20
-30

PCA

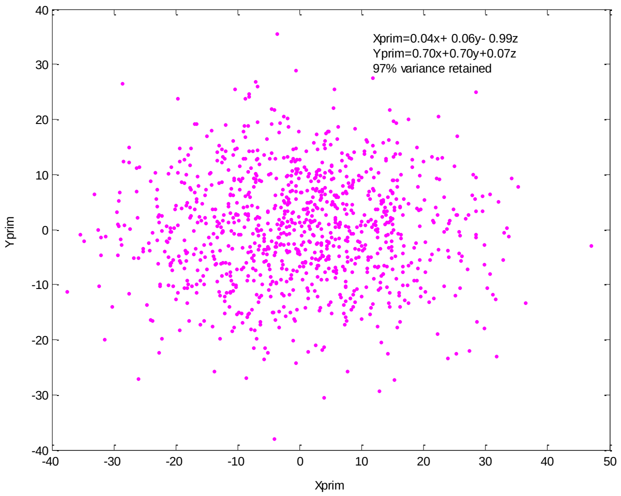


40
20
0
-20
-40
z
-40
-20
0
20
40
x
40
20
0
-20
-40
z
-40
-20
0
20
40
y
40
20
0
-20
-40
y
-40
-20
0
20
40
x

PCA



PCA


Xprim=0.04x+ 0.06y- 0.99z
Yprim=0.70x+0.70y+0.07z
97% variance retained

Principal component analysis (PCA)

- **PCA:**
  - linear transformation of a *d* dimensional input **x** to M dimensional vector **z** such that $M < d$ under which the retained variance is maximal.
  - Task independent
- **Fact:**
  - A vector **x** can be represented using a set of orthonormal vectors **u** $$\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i$$
  - Leads to transformation of coordinates (from **x** to **z** using **u**'s) $$z_i = \mathbf{u}_i^{T} \mathbf{x}$$

PCA

- **Idea:** replace *d* coordinates with *M* of z_i coordinates to represent *x*. We want to find the subset *M* of basis vectors.

$$\tilde{\mathbf{x}} = \sum_{i=1}^{M} z_i \mathbf{u}_i + \sum_{i=M+1}^{d} b_i \mathbf{u}_i$$

b_i - constant and fixed

- **How to choose the best set of basis vectors?**
  - We want the subset that gives the best approximation of data *x* in the dataset on average (we use least squares fit)

Error for data entry $\mathbf{x}^n$ $$\mathbf{x}^n - \tilde{\mathbf{x}}^n = \sum_{i=M+1}^{d} (z_i^n - b_i)\mathbf{u}_i$$

Reconstruction error

$$E_M = \frac{1}{2}\sum_{n=1}^{N} \left\| \mathbf{x}^n - \tilde{\mathbf{x}}^n \right\| = \frac{1}{2}\sum_{n=1}^{N} \sum_{i=M+1}^{d} (z_i^n - b_i)^2$$

PCA

- **Differentiate the error function** with regard to all b_i and set equal to 0 we get:

$$b_i = \frac{1}{N}\sum_{n=1}^{N} z_i^n = \mathbf{u}_i{}^T \bar{\mathbf{x}} \qquad \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^n$$

- Then we can rewrite:

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d} \mathbf{u}_i{}^T \boldsymbol{\Sigma} \mathbf{u}_i \qquad \boldsymbol{\Sigma} = \sum_{n=1}^{N} (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T$$

- The error function is optimized when basis vectors satisfy:

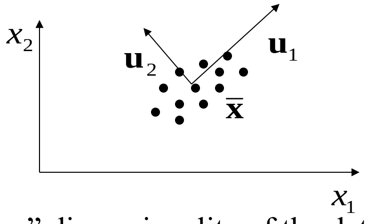
$$\boldsymbol{\Sigma}\mathbf{u}_i = \lambda_i \mathbf{u}_i \qquad E_M = \frac{1}{2}\sum_{i=M+1}^{d} \lambda_i$$

The best *M* basis vectors: discard vectors with *d*-*M* smallest eigenvalues (or keep vectors with M largest eigenvalues)

Eigenvector $\mathbf{u}_i$ – is called a **principal component**

PCA

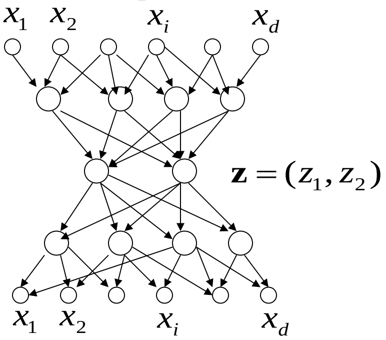
- Once eigenvectors $\mathbf{u}_i$ with largest eigenvalues are identified, they are used to transform the original *d*-dimensional data to *M* dimensions



- To find the “true” dimensionality of the data *d’* we can just look at eigenvalues that contribute the most (small eigenvalues are disregarded)
- **Problem:** PCA is a linear method. The “true” dimensionality can be overestimated. There can be non-linear correlations.
- **Modifications for nonlinearities:** kernel PCA

Dimensionality reduction with neural nets

- **PCA** is limited to linear dimensionality reduction
- To do non-linear reductions we can use neural nets
- **Auto-associative (or auto-encoder) network:** a neural network with the same inputs and outputs (***x***)



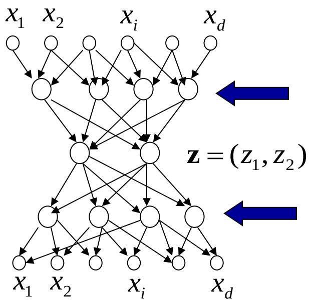
- The middle layer corresponds to the reduced dimensions

Dimensionality reduction with neural nets

- **Error criterion:**

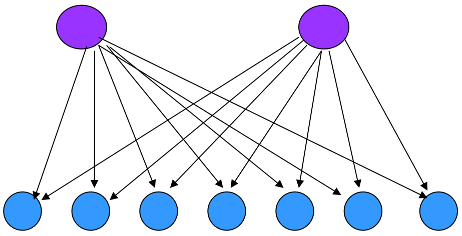
$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{d}\left(y_i(x^n) - x^n\right)^2$$

- Error measure tries to recover the original data through limited number of dimensions in the middle layer
- **Non-linearities** modeled through intermediate layers between the middle layer and input/output
- If no intermediate layers are used the model replicates PCA optimization through learning



Latent variable models

Latent variables (s): Dimensionality k



Observed variables x: real valued vars
Dimensionality d

Cooperative vector quantizer

Model:

Latent var s_i:

~ Bernoulli distribution
parameter: π_i

$$P(s_i \mid \pi_i) = \pi_i^{s_i}(1-\pi_i)^{1-s_i}$$

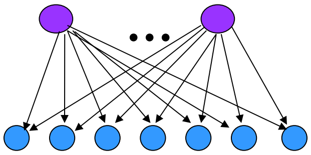
Observable variables x:

~ Normal distribution
parameters: **W, Σ**

$$P(\mathbf{x} \mid \mathbf{s}) = N(\mathbf{Ws}, \Sigma)$$

We assume $\Sigma = \sigma I$

s: k binary vars



x: d real valued vars

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & .. & w_{1k} \\ w_{21} & & & \\ & .. & & \\ w_{d1} & .. & .. & w_{dk} \end{pmatrix}$$

Joint for one instance of x and s:

$$P(\mathbf{x},\mathbf{s} \mid \Theta) = (2\pi)^{-d/2}\sigma^{-d/2}\exp\left\{-\frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{Ws})^T(\mathbf{x}-\mathbf{Ws})\right\}\prod_{i=1}^{k}\pi_i^{s_i}(1-\pi_i)^{(1-s_i)}$$

Other unsupervised methods

- **Factor analysis (a latent variable model)**
- Decompose signal into multiple Gaussian sources

$\mathbf{x} = \mathbf{A}\mathbf{s}$ X is a linear combination of values for sources

$$\mathbf{s} = \mathbf{W}\mathbf{x} = \mathbf{A}^{-1}\mathbf{x}$$

- **Independent component analysis:**
  - Identify independent components/signals/sources in the original data
  - Non-Gaussian signals

$$\mathbf{x} = \mathbf{A}\mathbf{s}$$

Multidimensional scaling

- Find a lower dimensional space projection such that the distances among data points are preserved

- Used in visualization – d-diminensional data transformed to 3D or 2D

- **Dissimilarities before projection** $\delta_{i,j} = \left\| x_i - x_j \right\|$

- **Objective:** Optimize points and their coordinates by fitting the dissimilarities afterwards

$$\min_{\{x_1, x_2, \cdots x_n\}} \sum_{i<j} (\left\| x_i' - x_j' \right\| - \delta_{ij})^2$$