CS 2750 Machine Learning
Lecture 17

Dimensionality reduction
Feature selection

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

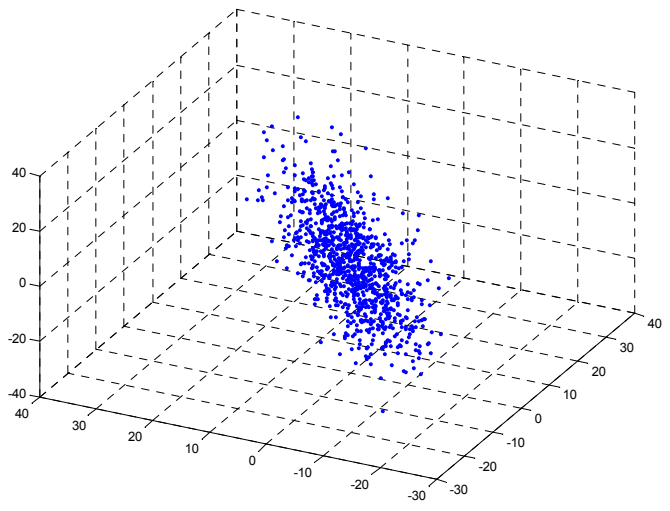
Dimensionality reduction. Motivation.

- **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**

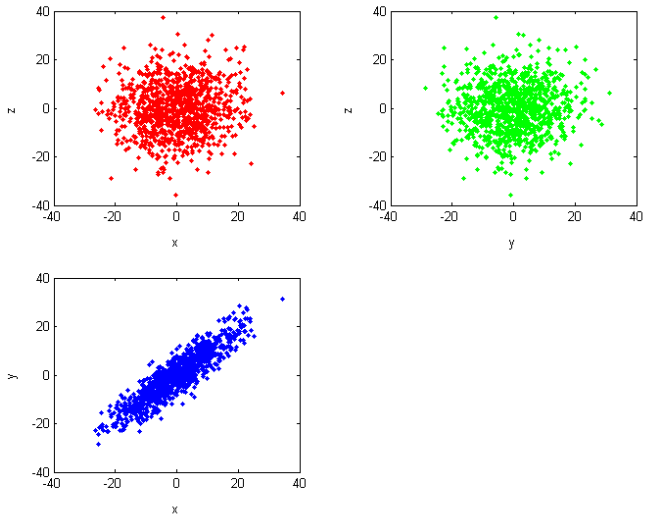
Principal component analysis (PCA)

- **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.

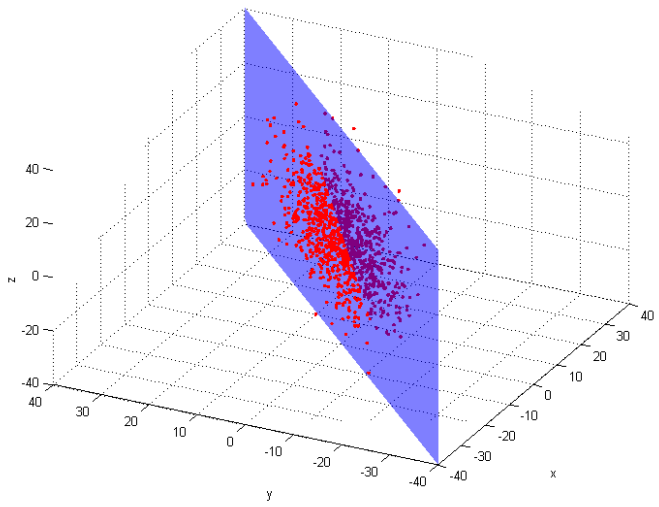
PCA



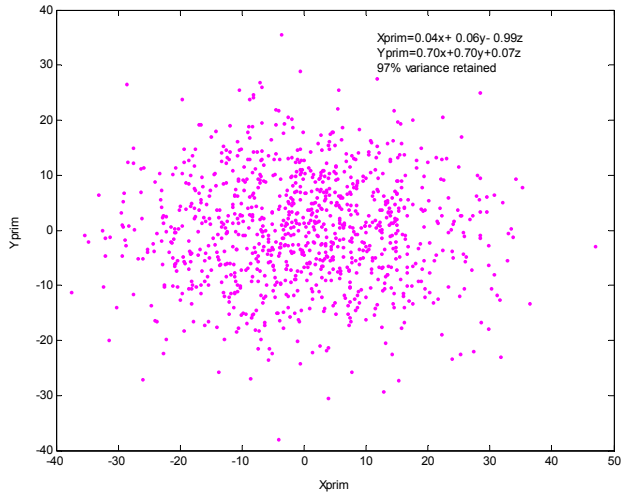
PCA



PCA

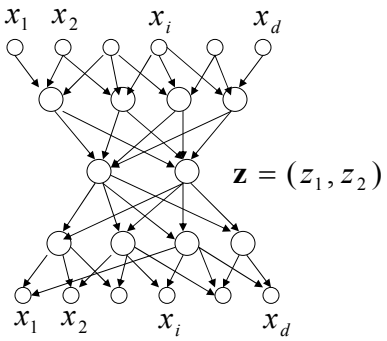


PCA



Dimensionality reduction with neural nets

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



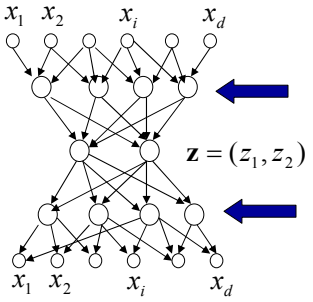
- The middle layer corresponds to the reduced dimensions

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



Other (unsupervised) methods

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

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

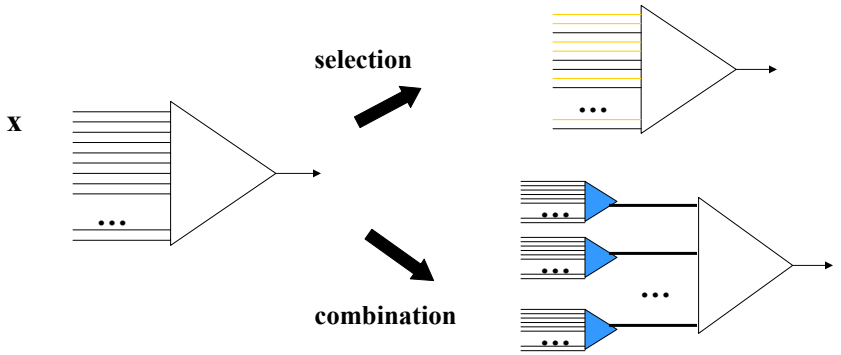
$\mathbf{s} = \mathbf{Wx} = \mathbf{A}^{-1}\mathbf{x}$

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**
    - inputs are reduced without looking at the output
      - PCA, independent component analysis, clustering of inputs
    - may lack the accuracy for classification/regression tasks

Task-dependent feature selection

Assume:

- **Classification problem**: $\mathbf{x}$ – input vector, y - output
- Feature mappings $\boldsymbol{\varphi} = \{\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots \phi_k(\mathbf{x}), \ldots\}$

Objective: Find a subset of 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 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 $Fisher\ (i) = \dfrac{\mu_i^{(+)^2} - \mu_i^{(-)^2}}{\sigma_i^{(+)^2} + \sigma_i^{(-)^2}}$
  - Area under Receiver Operating Characteristic (AUC) score

Problems:

- if many random features, 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**

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

- **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:
  $\widetilde{P}(y \mid \boldsymbol{\varphi} \setminus \phi_k) \approx \widetilde{P}(y \mid \boldsymbol{\varphi})$ for all values of ϕ_k, y
- Repeat removals until the probabilities differ.

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

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) actually built

How to find the appropriate feature set?

- **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)

Feature selection: wrappers

- **Example of a greedy (forward) search:**
  - logistic regression model with features

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

Choose the feature $\phi_i(\mathbf{x})$ with the best score

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

Choose the feature $\phi_j(\mathbf{x})$ with the best score

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

Etc.

When to stop ?

Internal cross-validation

- **Goal:** Stop the learning when smallest generalization error (performance on the population from which data were drawn)
- **Test set** can be used to estimate generalization error
  - Data different from the training set
- **Internal validation set** = test set used to stop the learning process
  - E.g. feature selection process
- **Cross-validation (*m*-fold):**
  - Divide the data into m equal partitions (of size N/m)
  - Hold out one partition for validation, train the classifier 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 all classifiers

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**