CS 1675 Introduction to Machine Learning
Lecture 21

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 data $D = \{\mathbf{x_1}, \mathbf{x_2}, .., \mathbf{x_N}\}$ 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

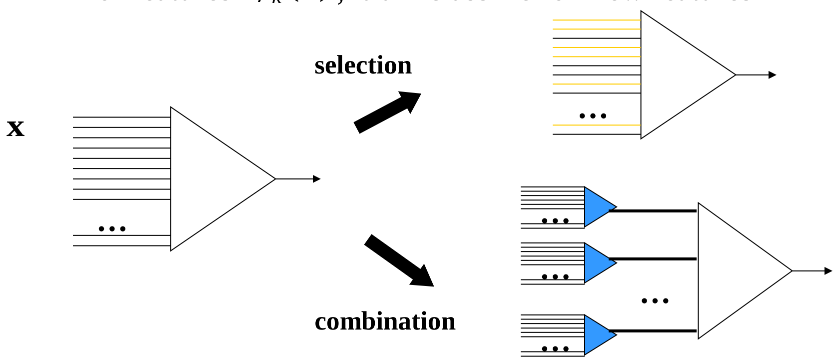
- **Classification problem example:**
  - We have an input data $\{\mathbf{x_1}, \mathbf{x_2}, .., \mathbf{x_N}\}$ such that

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

    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
  - Regularization methods, decision tree methods

Feature selection through filtering

Assume:

Classification problem:

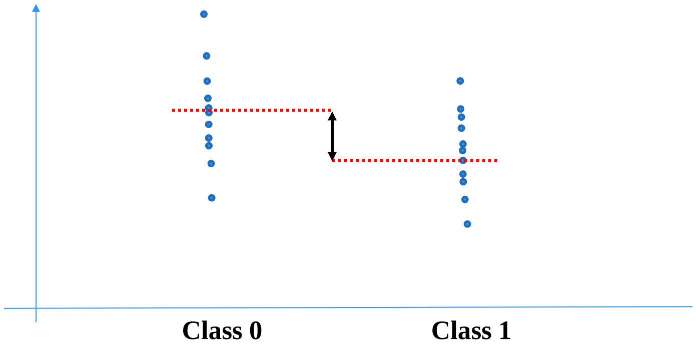
- **x** – input vector,
- *y* - output

- **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 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
    - Null hypothesis: **is mean of class 0 = mean of class 1**

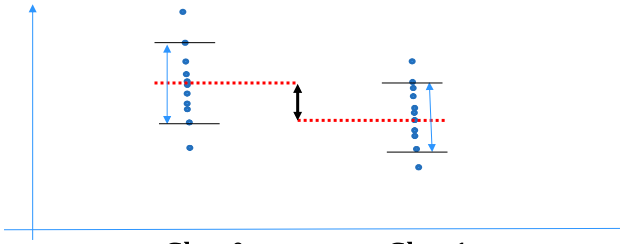


Differentially expressed features

Scores for measuring the differential expression

- **Fisher Score**

$$Fisher(i) = \frac{(\mu_i^{(+)} - \mu_i^{(-)})^2}{\sigma_i^{(+)2} + \sigma_i^{(-)2}}$$



- **AUROC score:** Area under Receiver Operating Characteristic curve

Feature filtering

- **Correlation coefficients**
  - Measures linear dependences

$$\rho(x_k, y) = \frac{Cov(x_k, y)}{\sqrt{Var(x_k)Var(y)}}$$

- **Mutual information**
  - Measures dependences
  - Needs discretized input values

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

Differentially expressed features

Problems:

- **Univariate score assumptions:**
  - Only one input and its effect on *y* is incorporated in the score
  - Effects of two features on *y* are considered to be independent

Partial solution:

- **Correlation based feature selection**
- **Idea:** good feature subsets contain features that are highly correlated with the class but independent of each other

$$M = \frac{k\bar{r}_{cx}}{\sqrt{k + k(k+1)\bar{r}_{xx}}}$$

- Average correlation between x and class $\bar{r}_{cx}$
- Average correlation between pairs of xs $\bar{r}_{xx}$
-

Differentially expressed features

Problems:

- **Many inputs and low sample size**
  - 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 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**