

CS 2750 Machine Learning
Lecture 21

Dimensionality reduction
Feature selection

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

CS 2750 Machine Learning

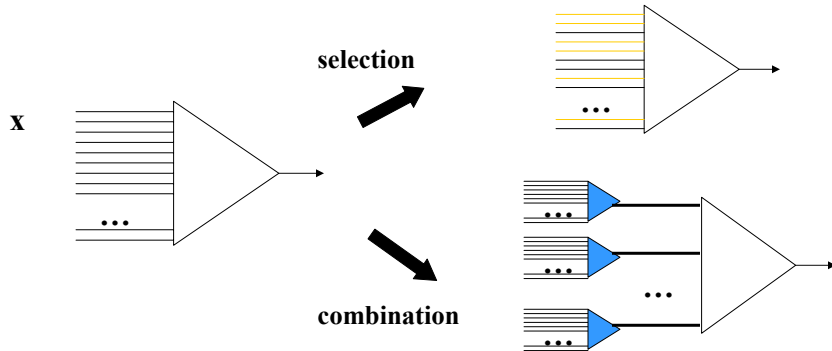
Dimensionality reduction. Motivation.

- **Classification problem example:**
 - We have an input data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ such that
$$\mathbf{x}_i = (x_i^1, x_i^2, \dots, x_i^d)$$
and a set of corresponding output labels $\{y_1, y_2, \dots, y_N\}$
 - Assume the dimension d of the data point \mathbf{x} is very large
 - We want to classify \mathbf{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)

CS 2750 Machine Learning

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



CS 2750 Machine Learning

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

CS 2750 Machine Learning

Task-dependent feature selection

Assume:

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

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

CS 2750 Machine Learning

Feature selection through filtering

Assume:

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

CS 2750 Machine Learning

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

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

- **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 Φ be a current set of features (starting from complete set)
- We can remove feature $\phi_k(\mathbf{x})$ from it when:
$$\tilde{P}(y | \Phi \setminus \phi_k) \approx \tilde{P}(y | \Phi) \quad \text{for all values of } \phi_k, y$$
- Repeat removals until the probabilities differ.

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

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

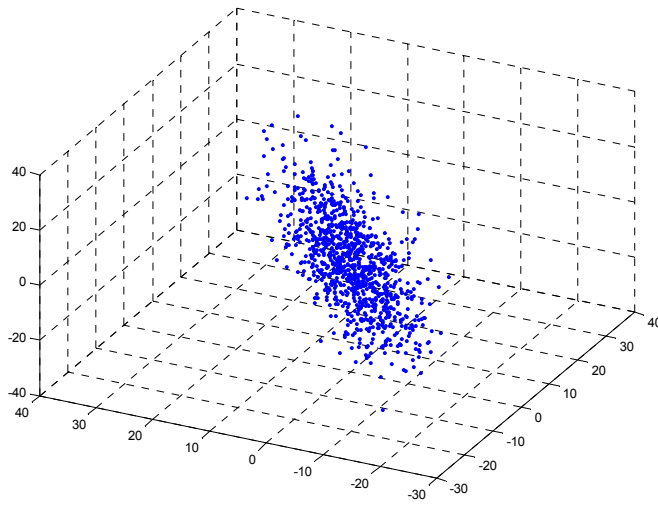
Embedded methods

- **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**

Principal component analysis (PCA)

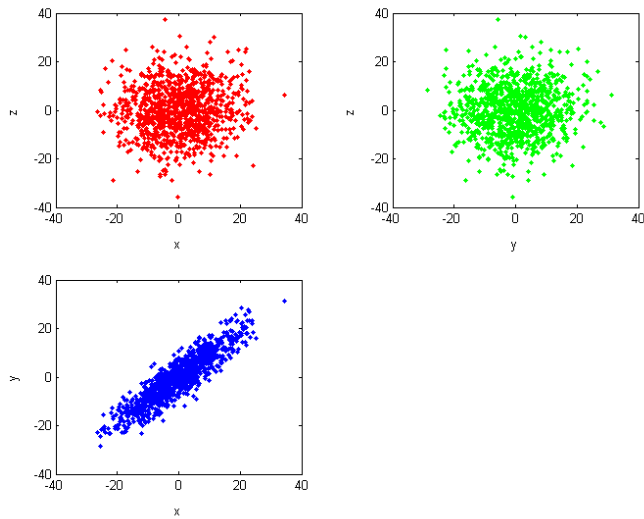
- **Objective:** We want to replace a high dimensional input with a small set of features (obtained by combining inputs)
 - Different from the feature subset selection !!!
- **PCA:**
 - A linear transformation of d dimensional input x to M dimensional feature vector z such that $M < d$ under which the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

PCA



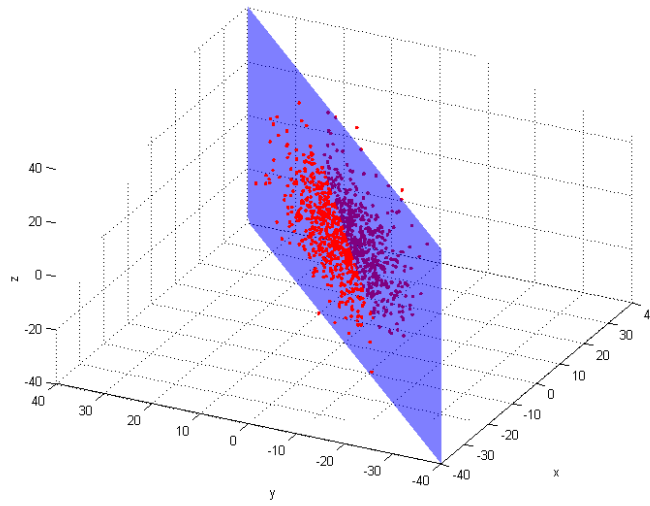
CS 2750 Machine Learning

PCA



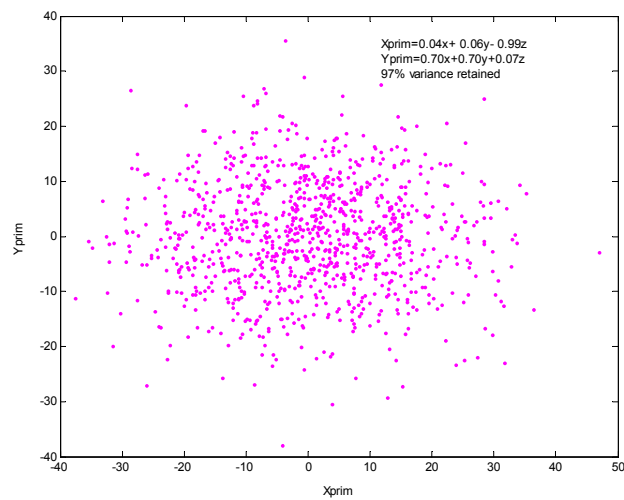
CS 2750 Machine Learning

PCA



CS 2750 Machine Learning

PCA



CS 2750 Machine Learning

Principal component analysis (PCA)

- **PCA:**

- linear transformation of d dimensional input x to M dimensional feature 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 $\mathbf{x}^n - \tilde{\mathbf{x}}^n = \sum_{i=M+1}^d (z_i^n - b_i) \mathbf{u}_i$

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

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:

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

- The error function is optimized when basis vectors satisfy:

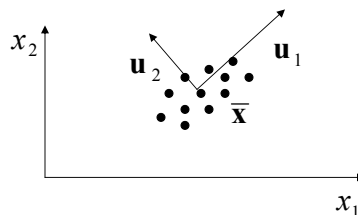
$$\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i \qquad E_M = \frac{1}{2} \sum_{i=M+1}^d \lambda_i$$

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

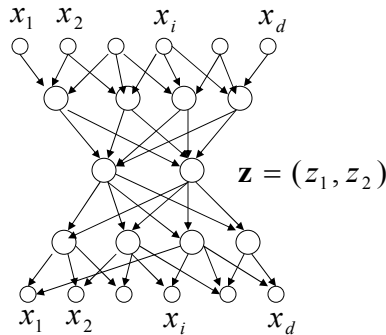
- Once eigenvectors \mathbf{u}_i with largest eigenvalues are identified, they are used to transform the original d -dimensional data to M dimensions



- 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.

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 (\mathbf{x})



- The middle layer corresponds to the reduced dimensions

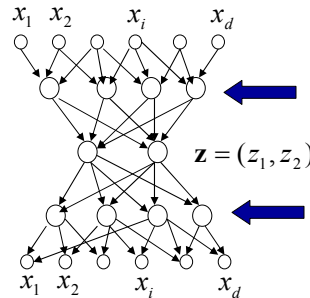
CS 2750 Machine Learning

Dimensionality reduction with neural nets

- **Error criterion:**

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

- 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



CS 2750 Machine Learning

Dimensionality reduction through clustering

- **Clustering algorithms**
 - group together “similar” instances in the data sample
- **Dimensionality reduction based on clustering:**
 - Replace a high dimensional data entry with a cluster label
- **Problem:**
 - Deterministic clustering gives only one label per input
 - May not be enough to represent the data for prediction
- **Solutions:**
 - Clustering over subsets of input data
 - Soft clustering (probability of a cluster is used directly)

CS 2750 Machine Learning

Dimensionality reduction through clustering

- **Soft clustering** (e.g. mixture of Gaussians) attempts to cover all instances in the data sample with a small number of groups
 - Each group is more or less responsible for a data entry (responsibility – a posterior of a group given the data entry)

Mixture of G. responsibility

$$h_{il} = \frac{\pi_i p(x_l | y_l = i)}{\sum_{u=1}^k \pi_u p(x_l | y_l = u)}$$

- **Dimensionality reduction based on soft clustering**
 - Replace a high dimensional data with the set of group posteriors
 - Feed all posteriors to the learner e.g. linear regressor, classifier

CS 2750 Machine Learning

Dimensionality reduction through clustering

- We can use the idea of soft clustering before applying regression/classification learning
- **Two stage algorithms**
 - Learn the clustering
 - Learn the classification
- Input clustering: \mathbf{x} (high dimensional)
- Output clustering (Input classifier): $p(c = i | \mathbf{x})$
- Output classifier: y
- **Example: Networks with Radial Basis Functions (RBFs)**
- **Problem:**
 - Clustering learns based on $p(\mathbf{x})$ (disregards the target)
 - Prediction based on $p(y | x)$

CS 2750 Machine Learning

Networks with radial basis functions

- An alternative to **multilayer NN for non-linearities**
- Radial basis functions:
$$f(x) = w_0 + \sum_{j=1}^k w_j \phi_j(\mathbf{x})$$
 - Based on interpolations of prototype points (**means**)
 - Affected by the distance between the \mathbf{x} and the **mean**
 - Fit the outputs of basis functions through the linear model
- Choice of basis functions:
 - Gaussian
$$\phi_j(x) = \exp\left\{-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}\right\}$$
- **Learning:**
 - In practice seem to work OK for up to 10 dimensions
 - For higher dimensions (ridge functions – logistic) combining multiple learners seem to do better job

CS 2750 Machine Learning