Introduction to Machine Learning NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University in Prague,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Lecture 11 – Feature selection

- **Curse of dimensionality**
- **Feature selection heuristics**
- **Bayes error**

Curse of dimensionality

According to the Wikipedia:

The **curse of dimensionality refers to various phenomena** that arise when analyzing and organizing data **in high-dimensional spaces** (often with hundreds or thousands of dimensions) **that do not occur in low-dimensional settings**.

The common theme of these problems is that **when the dimensionality increases, the volume of the space increases so fast that the available data become sparse**. This sparsity is problematic for any method that requires statistical significance. In order to obtain a statistically sound and reliable result, the **amount of data needed** to support the result often **grows exponentially with the dimensionality**.

Also organizing and searching data often relies on detecting areas where objects form groups with similar properties; **in high dimensional data** however **all objects appear to be sparse and dissimilar** in many ways which prevents common data organization strategies from being efficient.

Curse of dimensionality

High dimensional data is difficult to work because there are not enough observations to get good/reliable statistical estimates

Consider a simple example. Random vector of binary variables with the same binomial distributions. $(X_1, X_2, \ldots, X_n)$.

- Observe the frequency of different vector values if e.g. $\Pr(X_i = 1) = 1/2$ or $\Pr(X_i = 1) = 1/10$.
- If $\Pr(X_i = 1) = 1/10$, then $\Pr(1, 1, \ldots, 1) = 1/10^n$ (!)
  Thus, the need for data grows exponentially with the number of features!

Curse of dimensionality

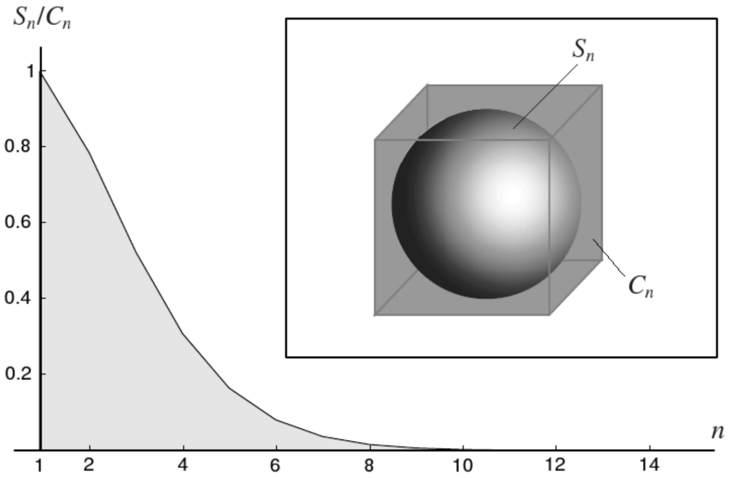
High-dimensional data is difficult to work not only because there are not enough observations to get good estimates... but also because **data distributed in a high dimensional space necesarily tends to be very sparse!**

This fact implies long distances between randomly distributed points

Consider a simple example. Uniformly distributed random points in an n-dimensional hypercube. – What will be their average/expected distance from the origin?

Curse of dimensionality – a geometrical illustration

Source: “The curse of dimensionality” by Mario Köppen



Ratio of the volumes of unit hypersphere and embedding hypercube

Curse of dimensionality

. . . also, in high-dimensional spaces there are long distances between randomly distributed points . . .

Another example with uniformly distributed random points in an n-dimensional hypercube:

- What will be the mutual distance between two randomly selected points?

Curse of dimensionality

. . . also, in high-dimensional spaces there are long distances between randomly distributed points . . .

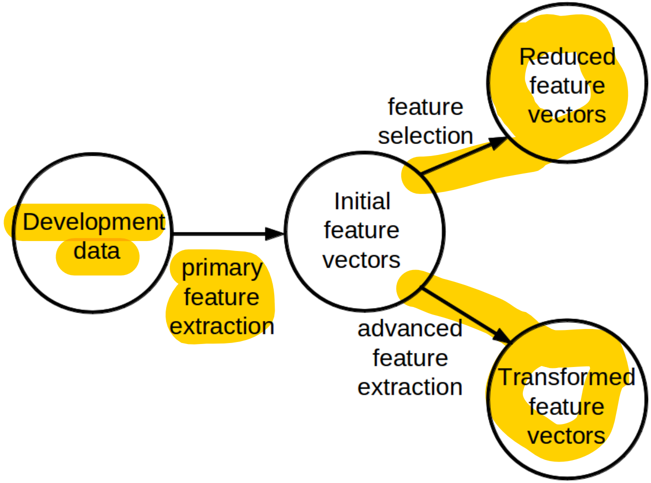
Another example with uniformly distributed random points in an n-dimensional hypercube:

- What will be the mutual distance between two randomly selected points?

"Near neighbours" often do not exist!
– Instead, typically you have only many "far neighbours". . .
. . . and you cannot recognize the "similar ones"

Feature extraction and feature selection

Processes and terminology related to feature extraction/selection


Development data
primary feature extraction
Initial feature vectors
feature selection
Reduced feature vectors
advanced feature extraction
Transformed feature vectors

Introduction to practical feature selection

Goal of the feature selection process = find a minimum set of variables that contain all the substantial information about predicting the target value

Introduction to practical feature selection

Goal of the feature selection process = find a minimum set of variables that contain all the substantial information about predicting the target value

- reduced feature space dimension in the dataset
- enhanced generalization and improved prediction performance by reducing overfitting (irrelevant input features may lead to overfitting)
- better chance to analyse the impact/importance of the features
- removing highly dependent features (some learning methods do not work well with them)
- lower model complexity and improved model interpretability
- feasible/shorter training times

Feature selection methods

Practical feature selection methods are heuristic

Feature selection methods

Practical feature selection methods are heuristic

Feature selection methods can be basically divided into

- **filters** – select feature subsets as a pre-processing step, independently of the learning method
- **wrappers** – use a machine learning algorithm in conjunction with internal cross validation procedure to score feature subsets by measuring their predictive power
- **embedded methods** – perform feature selection during the process of training

Simple methods in R: the FSelector package

```
> packageDescription('FSelector')
```

Description
This package provides functions for selecting attributes from a given dataset. Attribute subset selection is the process of identifying and removing as much of the irrevelant and redundant information as possible.

Feature ranking
$\sim$ aka variable importance metrics/measures

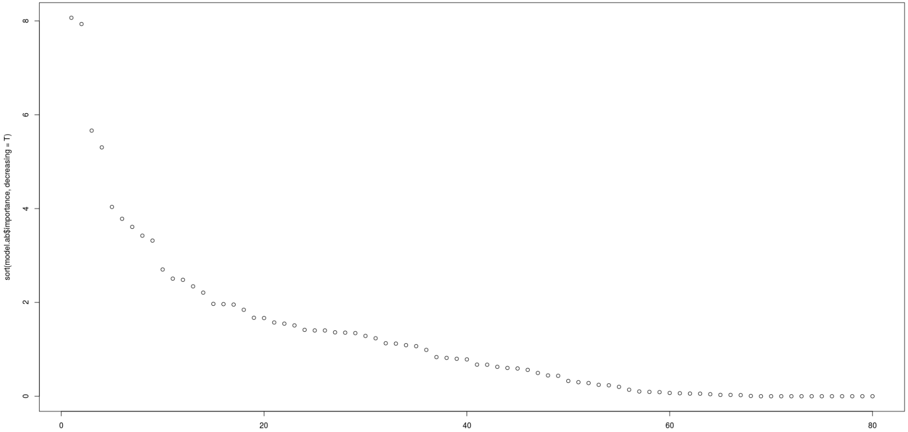
- We need a (real) function to evaluate how useful a feature is
- Frequently/mostly used:
  Information Gain, Gini Index, Chi-square, correlation coefficient, etc.
  - see Wikipedia: "Feature Selection"
  - see the FSelector package in R
- Disadvantages: such methods consider only one variable's contribution without other variables' influences
- However, using them you can easily recognize
  - really useful ones
  - completely unuseful ones
  - higly dependent/correlated ones

Practical methods for feature selection
Selected examples

- **Filters and wrappers**
    - greedy forward selection
    - greedy backward elimination
- **Variable importance produced by ensembles**
- **Feature selection by Lasso**
- **SVM-RFE – Recursive Feature Elimination**

Variable importance (AdaBoost) – *cry*

Example of the variable importance distribution



SVM-RFE feature selection algorithm

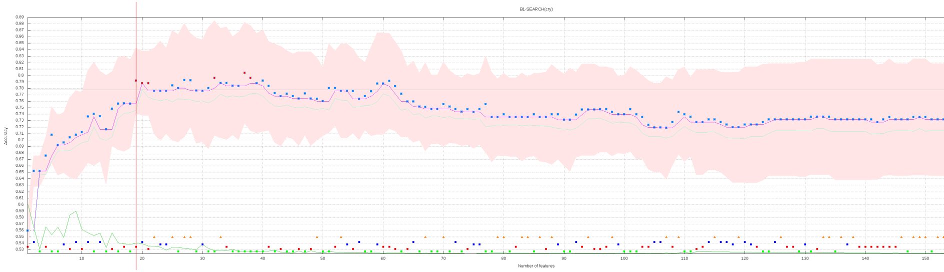
Example of succesfully combined heuristics

Algorithm 2 Recursive feature elimination using the SVM learner with cross-validated optimization of the SVM parameter *cost* in each iteration step.

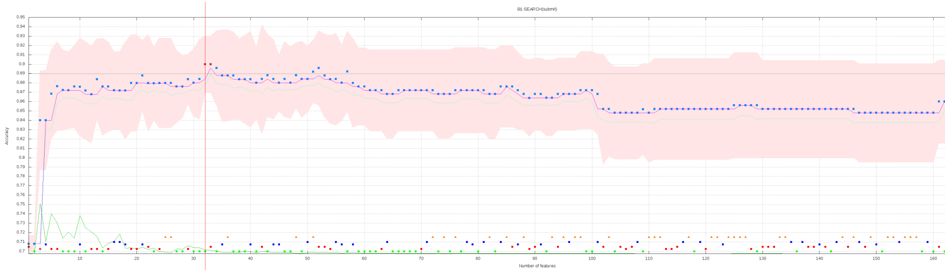
Input: Training data set and the initial feature set
Output: The best SVM classifier $\mathrm{M}_{\mathrm{max}}$ and the corresponding feature subset $\mathrm{S}_{\mathrm{max}}$

```
 1: $K \leftarrow$ the initial feature set size
 2: $S_K \leftarrow$ the initial feature set
 3: for $k \leftarrow K$ downto $1$ do
 4:     learn a linear SVM model using the feature set $S_k$ and tune its parameter cost
 5:     $\mathrm{M}_k \leftarrow$ the best tuned linear SVM model using the feature set $S_k$
 6:     $f_{\mathrm{worst}} \leftarrow$ the least useful feature in the model $\mathrm{M}_k$
 7:     $S_{k-1} \leftarrow S_k \setminus \{f_{\mathrm{worst}}\}$
 8: end for
 9: $\mathrm{M}_{\mathrm{max}} \leftarrow$ choose the best model from $\{\mathrm{M}_i\}_{i=1}^K$
10: $\mathrm{S}_{\mathrm{max}} \leftarrow$ the best feature subset corresponding to the best model $\mathrm{M}_{\mathrm{max}}$
```

SVM-RFE – *cry*

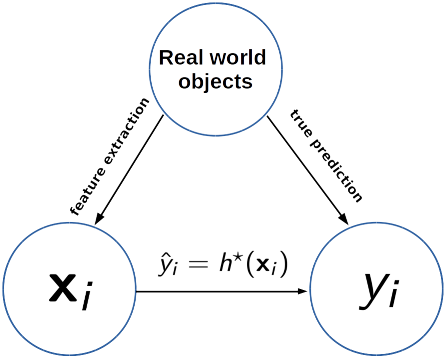


SVM-RFE – *submit*



Bayes classifier and Bayes error

**Imagine that you are able to develop a really optimal classifer.
Is the zero test error always feasible?**



Bayes classifier and Bayes error

Imagine that you are able to develop a really optimal classifer.
Is the zero test error always feasible?

The **Bayes classifier** minimises the probability of misclassification

Thus, by definition, error produced by the Bayes classifier is irreducible and is called *Bayes error*.

What is the lowest possible error rate

Bayes classifier assigns each example to the most likely class, given its feature values

$$\hat{y} = max_y \Pr(y \mid \mathbf{x})$$

The Bayes classifier produces the lowest possible test error rate, so called **Bayes error rate**

$$1 - \mathrm{E}\left(max_y \Pr(y \mid \mathbf{x})\right)$$

What is the lowest possible error rate

Practical view on your development data

Are there identical feature vectors in your data set?

- Get the same feature vectors
- How many of them have the same target value?