# 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,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

## Lecture 9 – Outline

- **Chi-square tests**
- **Curse of dimensionality**
- **Feature selection heuristics**
- **Bayes error**

## Curse of dimensionality

Source: Wikipedia

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

**Data sparsity**
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*.

**Dissimilarity of data points**
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 – example in high dimension

**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 Bernoulli 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!**

$\longrightarrow$ See the curse demo, Part I.

## Curse of dimensionality – data sparsity

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**

**Example**
Consider a simple example. Uniformly distributed random points in a unit n-dimensional hypercube.

– What will be their average/expected distance from the origin?

$\longrightarrow$ See the curse demo, Part II.

# Randomly distributed points in a hypercube

**Unit hypercube**

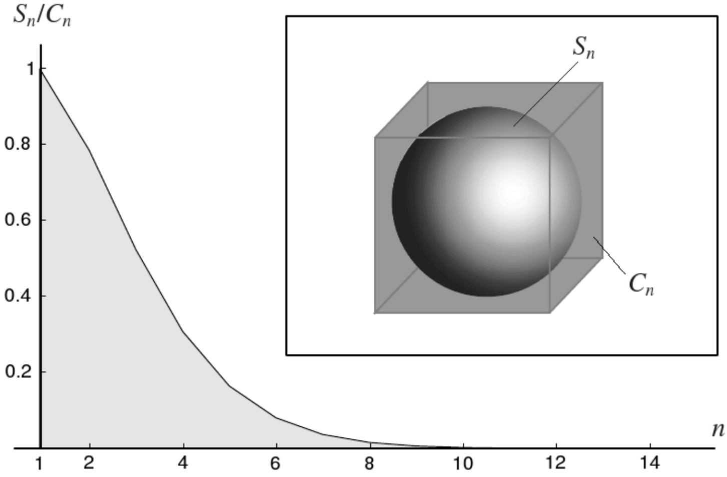
- The corners of the n-dimensional hypercube with sidelength 1 are all those points with coordinates being either 0 or 1.
- Volume of a unit hypercube is 1
- Length of the diagonal of the n-dimensional unit hypercube is $\sqrt{n}$

**What is the proportion of points with the distance from the origin $\leq 1$?**

| | |
|---|---|
| • two dimensions | $\sim \pi r^2/4 = \pi/4$ |
| • three dimensions | $\sim \frac{4}{3}\pi r^3/8 = \pi/6$ |
| • *n* dimensions | $\sim$ ? . . . goes to zero! |

## Curse of dimensionality – a geometric illustration

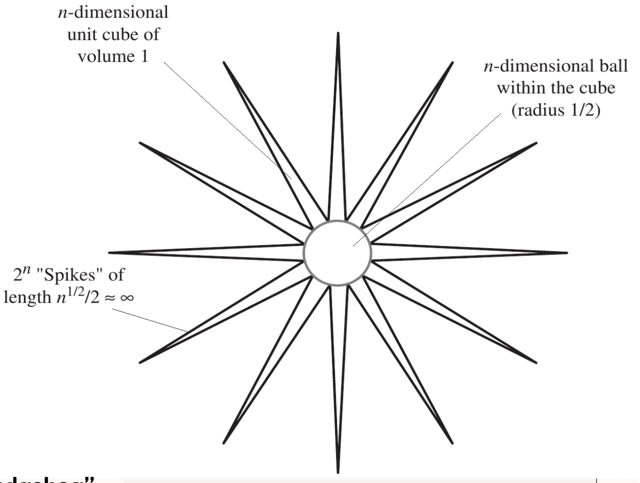
Source: "The curse of dimensionality" by Mario Köppen



**Ratio of the volumes of unit hypersphere and embedding hypercube**

## Curse of dimensionality – a hyperball in a unit cube

Source: "The curse of dimensionality" by Mario Köppen


$n$-dimensional unit cube of volume 1
$n$-dimensional ball within the cube (radius 1/2)
$2^n$ "Spikes" of length $n^{1/2}/2 \approx \infty$

**"Spherical hedgehog"**

While volume of the $n$-dimensional hypercube is 1, the length of its diagonal ($\sqrt{n}$) goes to infinity for increasing $n$, and volume of the embedded hypersphere goes to 0.

## Curse of dimensionality

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

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

- What will be the mutual distance between two randomly selected points?
  $\longrightarrow$ See the curse demo, Part III.

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

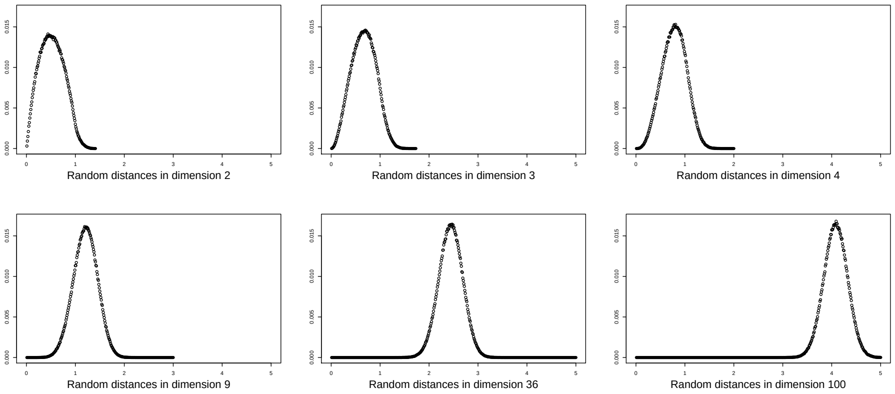
## Curse of dimensionality – demo code

```
# to generate a vector of N random distances in a hypercube of dim dimensions
distances.cube = function(N, dim) {
    distances = numeric(N)
    for(i in 1:N) {
      x = runif(dim); y = runif(dim)        # two random points in the cube
      distances[i] = sqrt(sum((x-y)^2))     # Euclidean distance
    }
    return(distances)
}


# example plot with empirical density in 3 dimensions
plot(((1:500)*5/500)[1:173],
     table(cut(distances.cube(10^6, 3),  breaks = (0:500)*5/500))[1:173]/10^6,
     xlim = c(0,5), ylim = c(0,0.017),
     yaxt="n", xlab="Random distances in dimension 3", ylab="")
     axis(2, at=c(0,0.005,0.01,0.015))
```

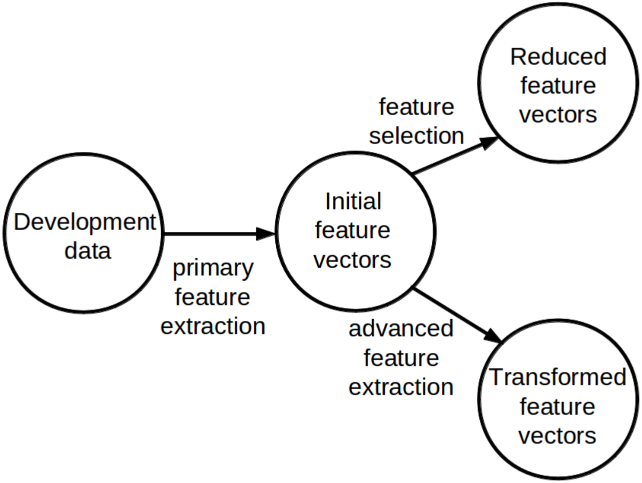
## Demo – distances of random points in a hypercube

Empirical density of distances between random points in a unit hypercube



## 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

## Why we need feature selection?

**Features without useful information make noise in the data!**

**Goal of the feature selection process**
= to efficiently find a minimum set of features that contain all the substantial information needed for predicting the target value

**More compact feature set can lead to**

- improved model interpretability,
- shorter training times,
- enhanced generalisation by reducing overfitting.

## Benefits of succesfull feature selection

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

## Introduction to practical feature selection

**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

## Filters, wrappers, and embedded methods

- **Filters** select features based on criteria independent of any supervised learner. Therefore, the performance of filters may not be optimum for a chosen learner.
- **Wrappers** use a learner as a black box to evaluate the relative usefulness of a feature subset. Wrappers search the best feature subset for a given supervised learner, however, wrappers tend to be computationally expensive.
- Instead of treating a learner as a black box, **embedded methods** select features using the information obtained from training a learner.

**Example**

A well-known example is SVM-RFE (support vector machine based on recursive feature elimination). At each iteration, SVM-RFE eliminates the feature with the smallest weight obtained from a trained SVM.

## 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
  - highly dependent/correlated ones

## 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.

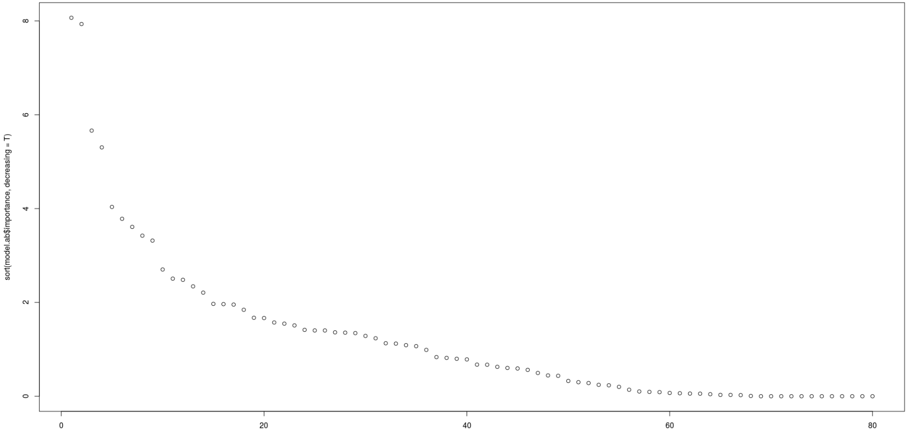
## Practical methods for feature selection

### Selected examples

- **Filters and wrappers**
  - greedy forward selection
  - greedy backward elimination
- **Variable importance produced by ensembles**
  - by Random Forests
  - by Adaboost
- **SVM-RFE – Recursive Feature Elimination**
- **Feature selection by Lasso**
  - – will be explained/discussed later in the lecture on Regularization

# Variable importance (AdaBoost) – *cry*

## Example of the variable importance distribution



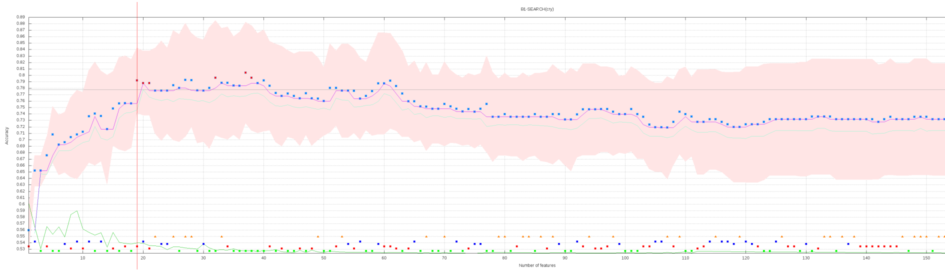
## SVM-RFE feature selection algorithm

**Example of succesfully combined heuristics**

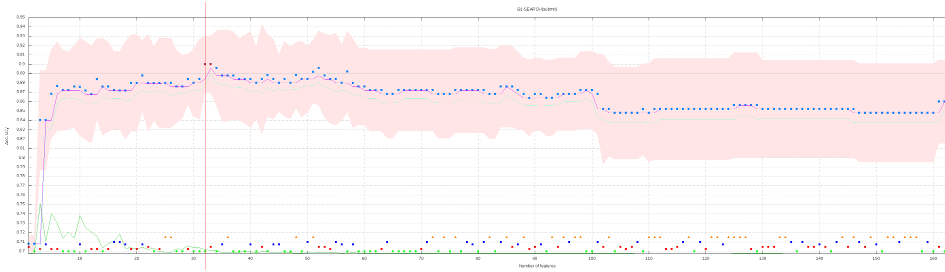
**Algorithm 2** Recursive feature elimination using the SVM learner with cross-validated optimization of the SVM parameter *cost* in each iteration step.

```
Input: Training data set and the initial feature set
Output: The best SVM classifier $\mathrm{M}_{\max}$ and the corresponding feature subset $\mathrm{S}_{\max}$
 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:     $f_{\text{worst}} \leftarrow$ the least useful feature in the model $\mathrm{M}_k$
 7:     $S_{k-1} \leftarrow S_k \setminus \{f_{\text{worst}}\}$
 8: end for
 9: $\mathrm{M}_{\max} \leftarrow$ choose the best model from $\{\mathrm{M}_i\}_{i=1}^{K}$
10: $\mathrm{S}_{\max} \leftarrow$ the best feature subset corresponding to the best model $\mathrm{M}_{\max}$
```

# SVM-RFE – *cry*

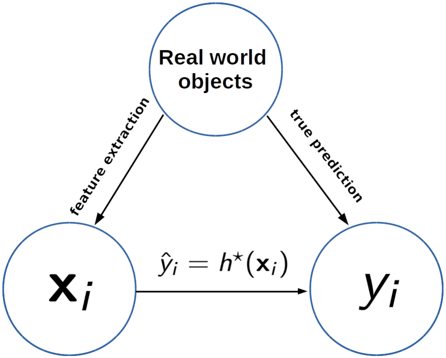


## SVM-RFE – *submit*



## 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?