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

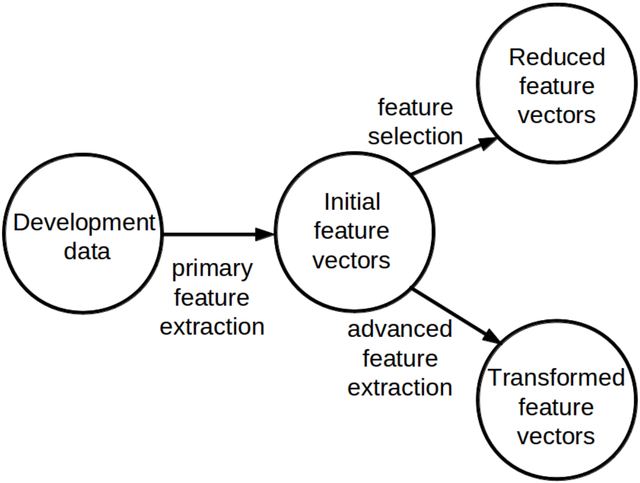
Feature analysis, importance, and selection

Outline

- **Why we need feature selection**
  - Curse of dimensionality
  - Benefits of succesfull feature selection
- **Feature selection heuristics**
  - Feature filtering
  - Feature ranking + greedy selection/elimination
  - Feature importances generated by Random Forests and AdaBoost
  - SVM-RFE – illustration
  - FSelector package
- **Bayes error**
- **Chi-square tests**
  - Independence test
  - Goodness-of-fit test

Feature extraction and feature selection

Processes and terminology related to feature extraction/selection



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.

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. $(X_1, X_2, \dots, X_n)$.

- 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 $\Pr(1, 1, \dots, 1) = 1/10^n$ (!)
  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

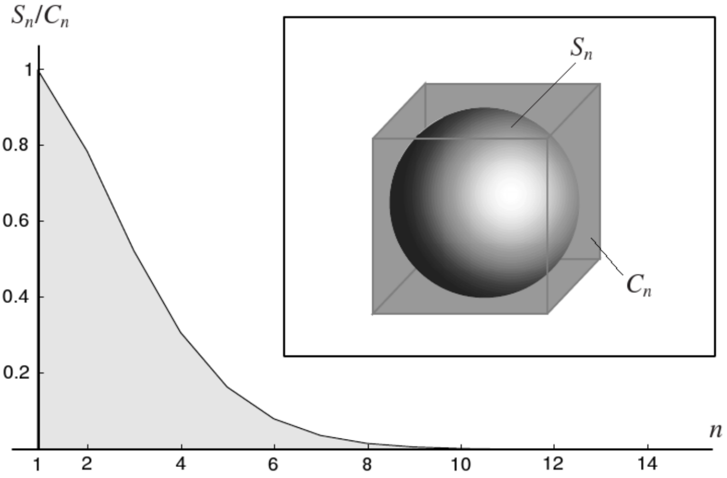
- The corners of the n-dimensional hypercube with sidelength 1 are all those points with coordinates being either 0 or 1.
- Volume of a unit hypercube is 1
- Length of the diagonal of the n-dimensional unit hypercube is $\sqrt{n}$

What is the proportion of points with the distance from the origin ≤ 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

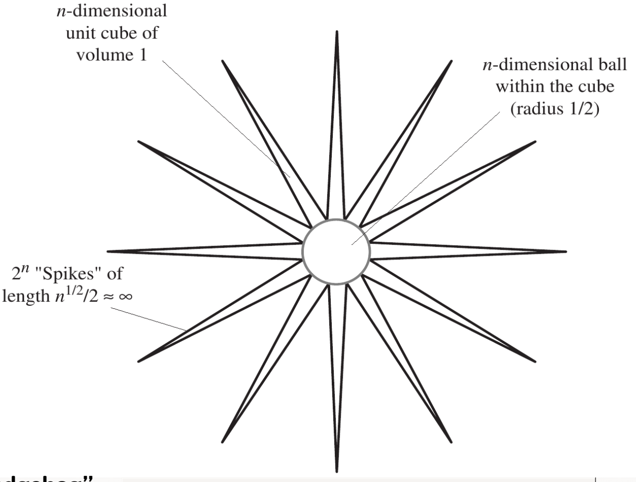
Source: "The curse of dimensionality" by Mario Köppen



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



"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.

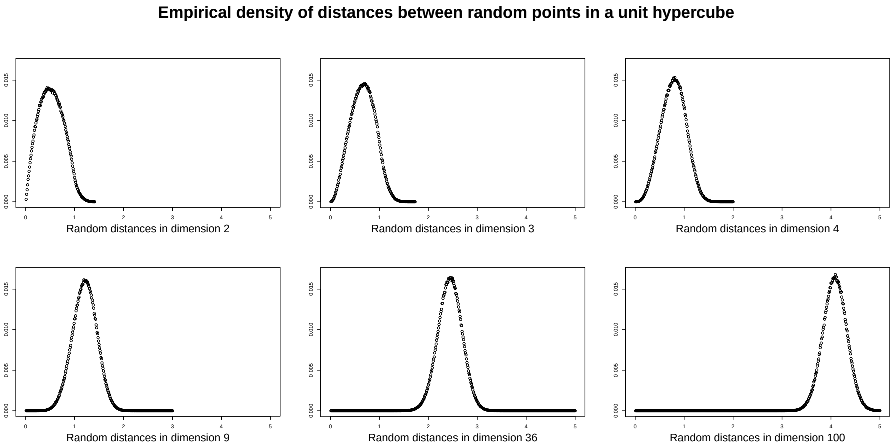
"Near neighbours" often do not exist!
– Instead, typically you have only many "far neighbours". . .
. . . and you cannot recognize the "similar ones"

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



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

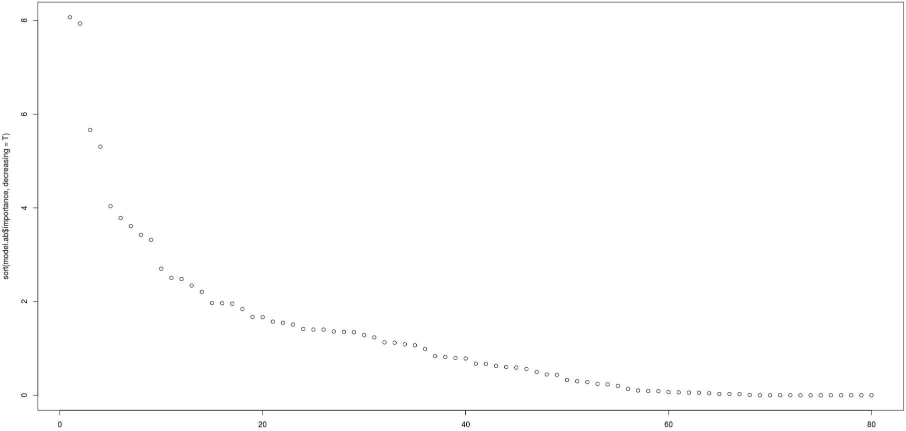
This package provides functions for selecting attributes from a given dataset. Attribute subset selection is the process of identifying and removing as much of the irrevelant and redundant information as possible.

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

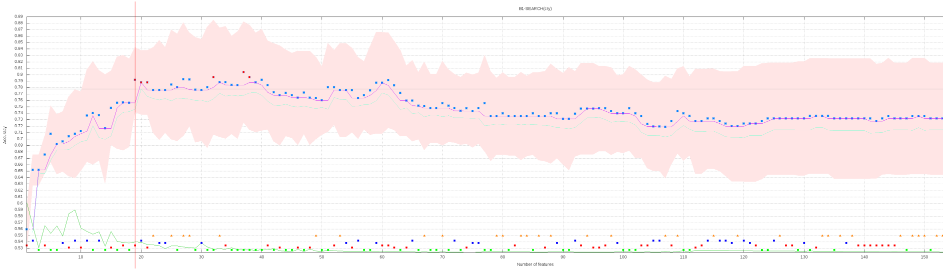
Algorithm 2 Recursive feature elimination using the SVM learner with cross-validated optimization of the SVM parameter *cost* in each iteration step.

Input: Training data set and the initial feature set

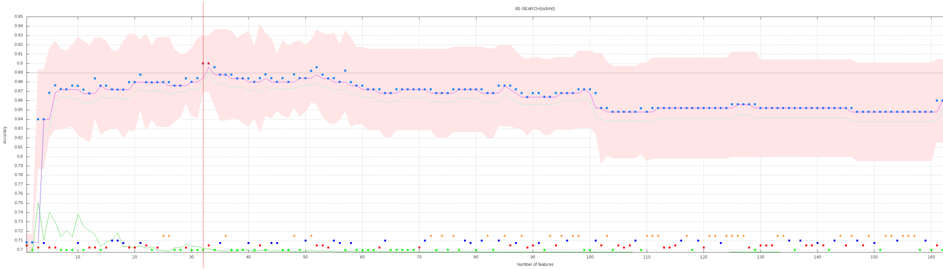
Output: The best SVM classifier $\mathrm{M}_{\mathrm{max}}$ and the corresponding feature subset $\mathrm{S}_{\mathrm{max}}$

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

SVM-RFE – *cry*

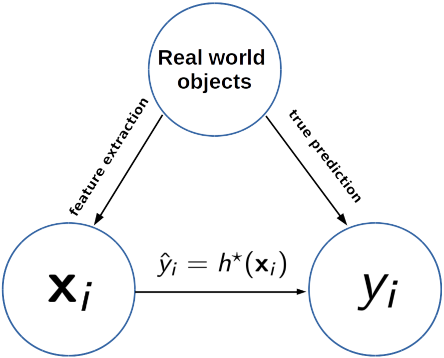


SVM-RFE – *submit*



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?

Pearson's χ^2 tests [chi-squared]

- **Test of independence**
  Are two variables, expressed in a contingency table, independent of each other?

- **Goodness-of-fit test**
  Does an observed frequency distribution differ from a hypothesized theoretical probability distribution?

- **Test of homogeneity**
  Does two observed frequency distributions of the same categorical variable come from populations with different probability distributions?

Sum of k independent standard normal variables

Let $Z_i \sim \mathrm{N}(0, 1)$ be independent variables with standard normal distribution.

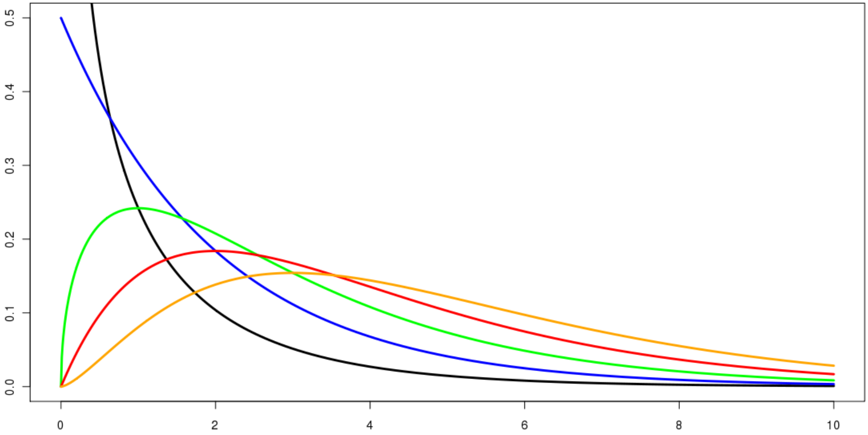
Then what is the distribution of $\sum_{i=1}^{k} Z_i^2$?

```
show.sum.Z.square <- function(k) {
  # shows the empirical distribution of the sum of
  # k independent standard normal variables
  # mean = k, variance = 2k

  sum.Z2 = 0
  for(i in 1:k){ sum.Z2 = sum.Z2 + rnorm(10^6)^2 }

  cat("Sample statistics:\n")
  print(summary(sum.Z2))
  cat("\nSample variance: ", var(sum.Z2), "\n")
  plot(cut(sum.Z2, 200))
}
```

χ^2 distribution – density



Chi-Squared test of independence

A test of independence assesses whether observations on two variables, expressed in a contingency table, are independent of each other.

χ^2 independence test

We observe two categorical variables. $O_{i,j}$ are the observed frequencies arranged in an contingency table. Expectations $E_{i,j}$ can be computed using estimated marginal probabilities. Pearson's χ^2 test is based on the following formula for Pearson's cumulative test statistic

$$X^2 = \sum_{i,j} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

Pearson's cumulative test statistic X^2 has approximately χ^2_{df} distribution, where the degrees of freedom is

$$df = (Rows - 1) \times \ (Cols - 1)$$

χ^2 independence test

Then we compare the test statistic with
χ^2 critical value $\chi^2_k(\alpha)$, which is defined by

$$\Pr\left\{X^2 > \chi^2_k(\alpha)\right\} = \alpha$$

Practical note

χ^2 critical value can be computed as a quantile.

```
> qchisq( (1-alpha), df=k )
```

TODO: Get familiar with functions `{p|d|q}chisq()` available in R.

Chi-Squared Goodness of Fit Test

The Chi-Squared Goodness of Fit Test is a test for comparing a theoretical distribution with the observed data from a sample.

χ^2 Goodness-of-fit test

Example 1
Rolling a die – after 600 rolls you got the following distribution

1	2	3	4	5	6
95	108	101	85	110	101

Question: Is the die fair? = Does it have the uniform distribution?

Example 2
Our hypothesis is that our classifier accuracy is 78 %. However, a test on 100 randomly chosen instances gives the following result

correct	error
81	19

Question: Should we reject the hypothesis?

χ^2 Goodness-of-fit test

Pearson's χ^2 goodness-of-fit test is based on the following formula for Pearson's cumulative test statistic

$$X^2 = \sum_{i=1}^{m} \frac{(O_i - E_i)^2}{E_i}$$

If the observed variables O_i have multinomial distribution, then Pearson's cumulative test statistic X^2 has approximately χ^2_{m-1} distribution.

χ^2 Goodness-of-fit test — example

Example based on real data

SENSES	estimated probabilities	test set observations
cord	9.2%	37
division	8.9%	51
formation	8.1%	52
phone	10.6%	44
product	53.5%	268
text	9.8%	48

```
> x = c(37, 51, 52, 44, 268, 48)
> p = c(9.2, 8.9, 8.1, 10.6, 53.5, 9.8)/100
```

Examination requirements

- Curse of dimensionality – what is the issue
- Feature selection – principles and heuristics
  - Feature importances generated by Random Forests and AdaBoost
- Bayes classifier and Bayes error – definition and meaning
- Chi-square tests – theory and practical use
  - Independence test
  - Goodness-of-fit test