LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Introduction to Machine Learning

Curse of Dimensionality

Department of Statistics – LMU Munich

CURSE OF DIMENSIONALITY: EXAMPLE

Assume that we are given 20 emails, 10 of them are spam and 10 are not.

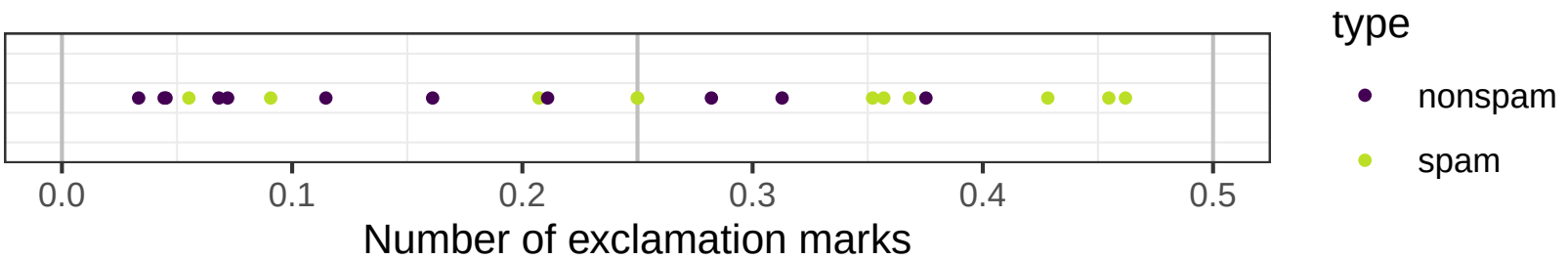
Our goal is to predict if a new incoming mail is spam or not.

For each email, we extract the following features:

- frequency of exclamation marks (in %)
- the length of the longest sequence of capital letters
- the frequency of certain words, e.g., “free” (in %)
- ...

... and we could extract many more features!

CURSE OF DIMENSIONALITY: EXAMPLE

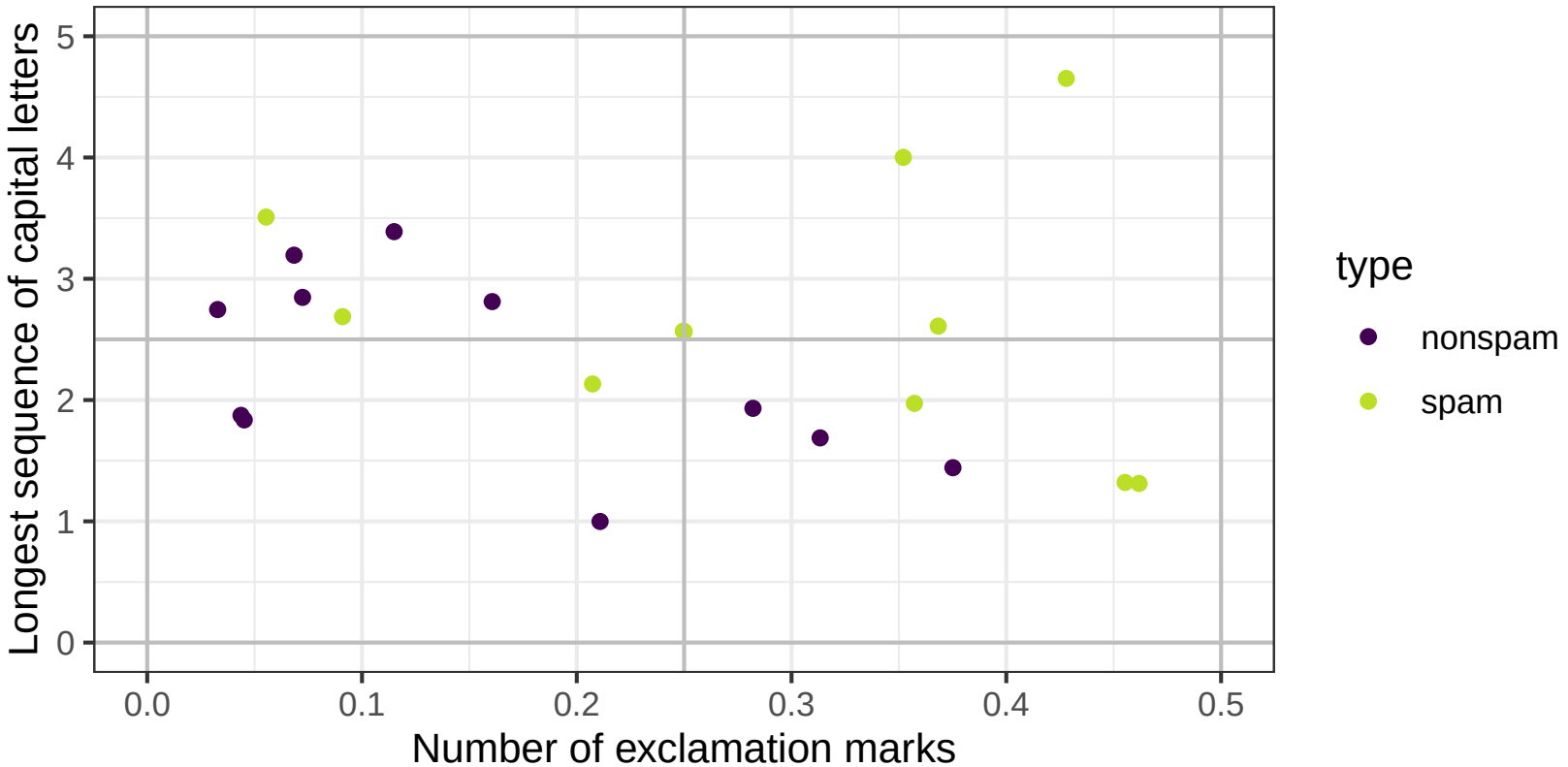


Based on the number of exclamation marks, we train a very simple classifier (a decision stump with split point $\mathbf{x} = 0.25$):

- We divide the input space in 2 equally sized regions
- In the second region $[0.25, 0.5]$, 7 out of 10 are spam
- Given that at least 0.25% of all letters are exclamation marks, an email is spam with a probability of $\frac{7}{10} = 0.7$

CURSE OF DIMENSIONALITY: EXAMPLE

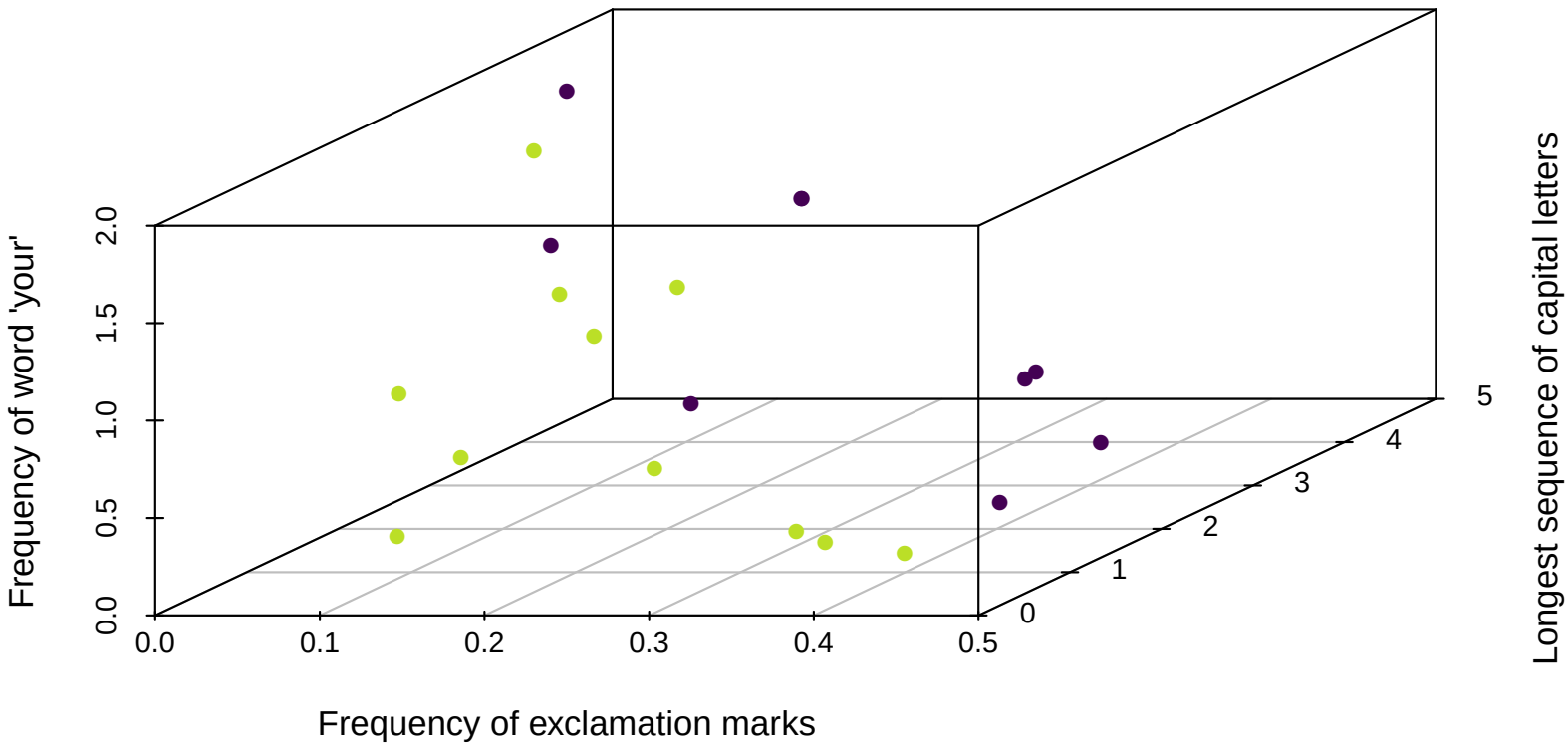
Let us feed more information into our classifier. We include a feature that contains the length of the longest sequence of capital letters:



- In the 1D case we had 20 observations across 2 regions
- Now, the same number of observations is spread across 4 regions

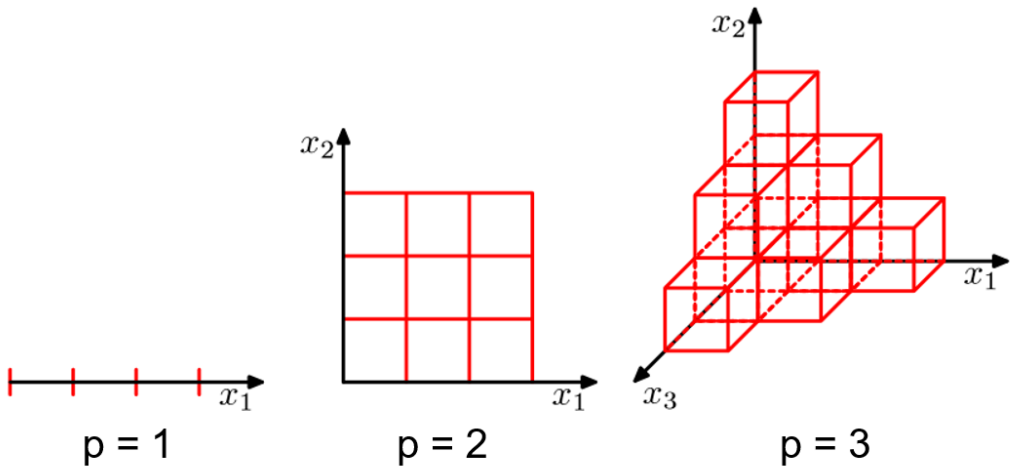
CURSE OF DIMENSIONALITY: EXAMPLE

Let us further increase the dimensionality to 3 by using the frequency of the word “your” in an email.



CURSE OF DIMENSIONALITY: EXAMPLE

- When adding a third dimension, the same number of observations is spread across 8 regions
- In 4 dimensions the data points are spread across 16 cells, in 5 dimensions across 32 cells and so on ...
- As dimensionality increases the data become **sparse**, some of the cells become empty
- In each of the blocks, there might be too few data to understand the distribution of the data and to model it



Bishop, Pattern Recognition and Machine Learning, 2006

CURSE OF DIMENSIONALITY

- The phenomenon of data becoming sparse in high-dimensional spaces is one effect of the **curse of dimensionality**
- The **curse of dimensionality** refers to various phenomena that arise when analyzing data in high-dimensional spaces that do not occur in low-dimensional spaces
- Our intuition about the geometry of a space is formed in two and three dimensions.
- This intuition is often misleading in high-dimensional spaces!

COD and learning algorithms

EXAMPLE: K-NN

Let's look at the performance of algorithms for increasing dimensionality. First, we consider the k-NN algorithm:

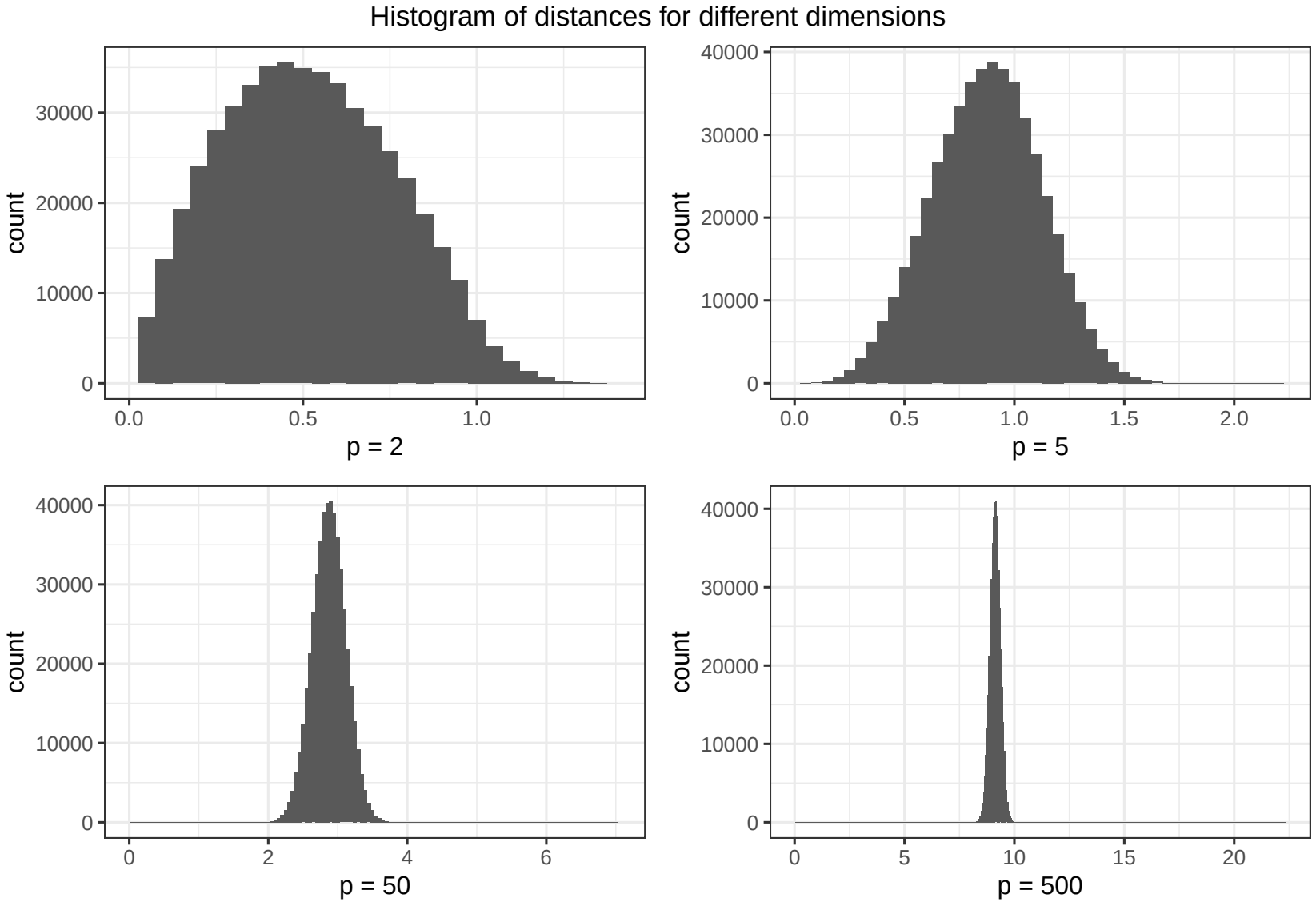
- In a high dimensional space, data points are spread across a huge space
- The distance to the **next neighbor** becomes extremely large
- The distance might even get that large that all points are **equally far** away - we cannot really determine the nearest neighbor anymore

EXAMPLE: K-NN

Minimal, mean and maximal (NN)-distances of 10^{4p} points uniformly distributed in the hypercube $[0, 1]^p$:

p	$\min(d(x,y))$	$\overline{d(x,y)}$	$\max(d(x,y))$	$\overline{d_{NN1}(x,x')}$	$\max(d_{NN1}(x,x'))$
1	0.000000012	0.33	1	0.00005	0.00042
2	0.00011	0.52	1.4	0.0051	0.02
3	0.0021	0.66	1.7	0.026	0.073
5	0.016	0.88	2	0.11	0.23
10	0.15	1.3	2.5	0.39	0.63
20	0.55	1.8	3	0.9	1.2
50	1.5	2.9	4.1	2	2.4
100	2.7	4.1	5.4	3.2	3.5
500	7.8	9.1	10	8.2	8.6

EXAMPLE: K-NN


Histogram of distances for different dimensions
count
p = 2
p = 5
p = 50
p = 500

EXAMPLE: K-NN

The consequences for the k-nearest neighbors approach can be summarized as follows:

- At constant sample size n and growing p, the distance between the observations increases exponentially
  $\rightarrow$ coverage of the p-dimensional space decreases
  $\rightarrow$ every point becomes isolated / far way from all other points
- The size of the neighborhood $N_k(x)$ also "increases"
  (at constant k)
  $\rightarrow$ it's no longer a "local" method.
- Reducing k dramatically does not help much either, since the fewer observations we average, the higher the variance of our fit.

$\rightarrow$ k-NN estimates get more inaccurate with increasing dimensionality of the data.

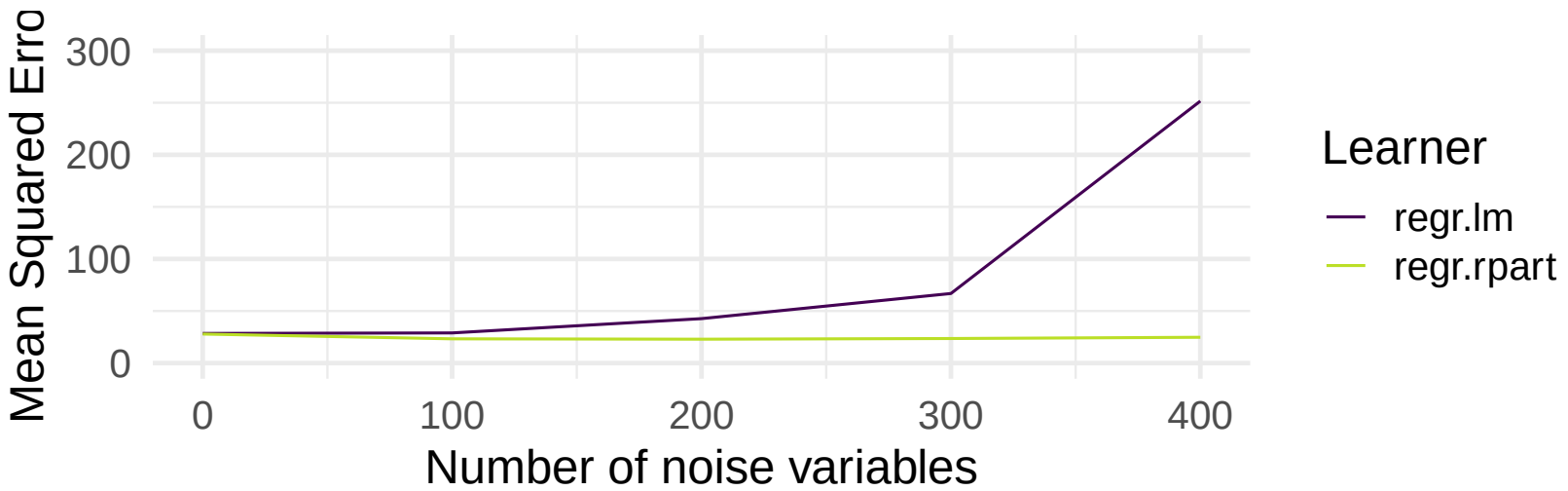
EXAMPLE: LINEAR MODEL

We also investigate how the linear model behaves in high dimensional spaces.

- We take the Boston Housing data set, where the value of houses in the area around Boston is predicted based on 14 features describing the region (e.g. crime rate, status of the population, etc.).
- We train a linear model on the data.
- We artificially create a high-dimensional dataset by adding $100, 200, 300, \ldots$ noise variables (containing no information at all) and look at the performance of a linear model trained on this modified data (10 times repeated 10-fold CV).

EXAMPLE: LINEAR MODEL

We compare the performance of a linear model to that of a regression tree.



$\rightarrow$ The LM struggles with the added noise features, while our tree seems to nicely filter them out.

Note: Trees automatically perform feature selection as only one feature at a time is considered for splitting (the smaller the depth of the tree, the less features are selected). Thus, they often perform well in high-dimensional settings.

COD: WAYS OUT

Many methods besides the LM struggle with the curse of dimensionality. A large part of ML is concerned with dealing with this problem and finding ways around it.

Possible approaches are:

- Increasing the space coverage by gathering more observations (not always viable in practice!)
- Reducing the number of dimensions before training (e. g. by using domain knowledge, PCA or feature selection)
- Regularization