Stochastic Machine Learning
Chapter 01 - Introduction, part II

Thorsten Schmidt

Abteilung für Mathematische Stochastik

www.stochastik.uni-freiburg.de
thorsten.schmidt@stochastik.uni-freiburg.de

SS 2024

Short recap

In the last lecture we have learnt:

- What is artificial intelligence
- What is machine learning (and where is the difference)
- What is a neural network (shallow / deep)
- Examples of machine learning (we had two explicit ones)

1. Introduction → Types of ML

We start with a small classification. **1. Supervised learning**. The data consists of datapoints and associated labels, i.e. we start from the dataset

$$(x_i, y_i)_{i \in I}.$$

We give some examples:

- **Image recognition** (face recognition) where the images come with labels, i.e. cats / dogs or the person to which the image is associated to.
- **Spam filter** the training set contains emails together with the label spam / no spam.
- **Speech recognition** here sample speech files comes together with the content of the sentences. It is clear, that some sort of grammar understanding helps to break up the sentences into smaller pieces, i.e. words.
- **Ratings** here, to a creditor we assign the credit quality (AAA, ...) A typical finance application.
- **Language Models** here the idea is to predict the next letter and train it on a large dataset.

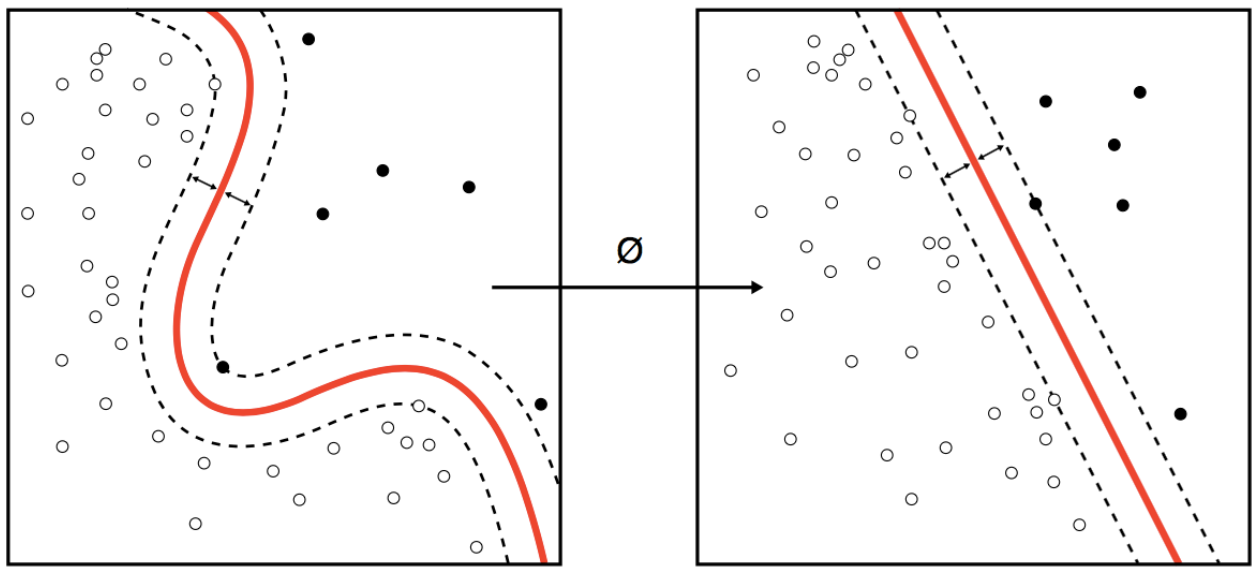
2. Unsupervised learning

In this case the data just comes at it is, i.e.

$$(x_i)_{i \in I}$$

and one goal would be to identify a certain structure from the data itself.

- One goal would be to identify a certain structure from the data itself. In this sense the machine learning algorithm shall itself find a characteristics which divides the data into suitable subsets.
- Clustering: You have a number of images (or objects) and want to cluster them into certain types. The clusters themselves have to be found.
- Outlier detection: closely related. You have a dataset and want to identify extraordinary data points. Examples: fraud detection, changepoint analysis (climate) etc.
- Feature learning and dimension reduction

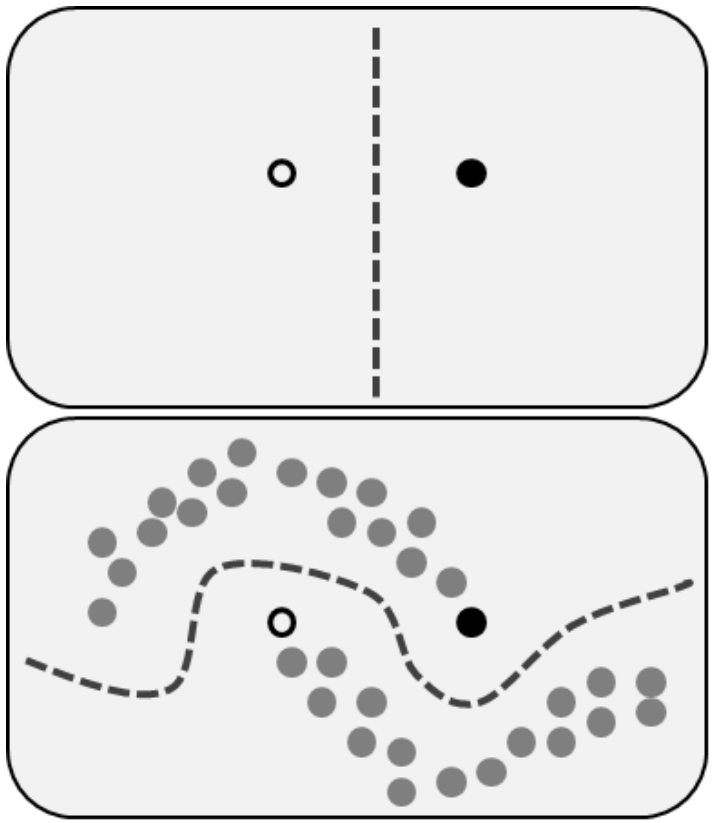


Picture by: Alisneaky, svg version by User:Zirguezi - Own work,
CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=47868867

Semi-supervised learning

In semi-supervised learning only a few data are labelled and many are unlabelled.

- Labelling typically is quite expensive and the additional use of unlabelled data might improve the performance. However, some assumptions need to be made, such that this procedure works through.



Picture by: Techerin - Own work,
CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=19514958

Questions

- What is (semi-/un-/-supervised) learning ?
- Give examples
- The examples you have been chosing in the first part, please classify them

Dynamic contexts

- It is apparent, that the above questions have been static
- Many applications are dynamic !
- To drive a car
- To manage a portfolio
- To predict future evolutions from a time-series

This will require different methods which we will meet in the course. We will use Markov processes when we study **Reinforcement learning** in greater detail.

Statistical Learning

This new area of statistics is quite related to machine learning and we will study a number of relevant problems[1].

- Formally, we have an observation given by pairs (x_i, y_i), $i \in I$ and randomness is modelled with an (unknown) probability distribution
- The task is to predict y based on x.
- From all functions f in some set $\mathscr{H}$ we want to choose f so that the *expected risk*

$$E[L(f(X), Y)]$$

  is minimal. Here, L is some chosen loss function.
- because the probability is unknown, one estimates the expected risk with the *empricial risk*

$$\frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i).$$

Popular and well-known examples are

- **Regression** in the simple least-squares regression, $f(x) = m + nx$ and $L(\cdot) = \cdot^2$
- **Classification** also falls into this framework: here Y takes only finitely many values, like $\{A, B, C, ...\}$ and possibly a step-function is chosen as loss function.

[1] There is a lot of interesting literature in this area: e.g. T. Hastie, R. Tibshirani, and J. Friedman (2009). **The Elements of Statistical Learning**. Springer Series in Statistics. Springer New York Inc. URL: https://statweb.stanford.edu/~tibs/ElemStatLearn/, Vladimir Vapnik (2013). **The nature of statistical learning theory**. Springer science & business media.

Definition

A computer program learns from experience E with respect to tasks T, if its performance P improves with experience E.

This quite vague definition allows us to develop some intuition about the situation.

- **Experience** is given by an increasing sequence of observations, for example $X_1, X_2, \ldots, X_t$ could represent the information at time t. This is typically decoded in a **filtration**: a filtration is an increasing sequence of sub-σ-fields $(\mathscr{F}_t)_{t \in \mathcal{T}}$.
- The performance is often measured in terms of an **utility function** or a **loss function**. For example the utility at time t could be given by $U(X_t)$ with an function U. U could of course depend on more variables.
- A typical task is prediction: We have a statistic $S = S(X_1, \ldots, X_t)$ and want to predict X_{t+1}. The performance is measured via the prediction error

$$\| S(X_1, \ldots, X_t) - X_{t+1} \|^2 .$$

- The minimizer of the square distance is given by the conditional expectation

$$S = E[X_{t+1} | X_1, \ldots, X_t].$$

Definition

A computer program learns from experience E with respect to tasks T, if its performance P improves with experience E.

This quite vague definition allows us to develo

- **Experience** is given by an increasing sequ
  $X_1, X_2, \ldots, X_t$ could represent the info
  filtration: a filtration is an increasing seq
- The performance is often measured in te
  example the utility at time t could be giv
  course depend on more variables.
- A typical task is prediction: We have a st
  X_{t+1}. The performance is measured via

$$\| S(X_1, \ldots, X_t) - X_{t+1} \| \quad .$$

- The minimizer of the square distance is given by the conditional expectation

$$S = E[X_{t+1} | X_1, \ldots, X_t].$$

A σ-algebra (field) satisfies:

1. $\Omega \in \mathscr{F}$
2. $A \in \mathscr{F} \Rightarrow \bar{A} = \Omega \backslash A \in \mathscr{F}$
3. $A_i \in \mathscr{F}, i \geq 1 \Rightarrow \cup_{i \geq 1} A_i \in \mathscr{F}$.

What is a probability measure ?
What is a probability space ?
What is the easiest example ?
What is a random variable ?
What is the normal distribution (density) ?

One very simple learning algorithm is linear regression, a classical statistical concept. Here it arises as an example of **supervised learning**.

Example (Linear Regression)

Suppose we oberseve pairs $(x_i, y_i)_{i=1,\dots,n}$ and want to predict y on basis of x. **Linear** regression requires

$$\hat{y}(x) = \beta x$$

with some weight $\beta \in \mathbb{R}$. We specify a loss function[2]

$$\mathrm{RSS}(\beta) := \sum_{i=1}^{n} (y_i - \hat{y}(x_i))^2$$

and minimize over β.

One could choose $-$MSE as utility function. So how does the system **learn**?

[2] Given by the Residual Sum of Squares here.

The system learns by maximizing the utility, i.e. minimizing the MSE for each n. And additional data will lead to a better prediciton. We will later see that this is in a certain sense indeed optimal.

The system learns by maximizing the utility, i.e. minimizing the MSE for each n. And additional data will lead to a better prediciton. We will later see that this is in a certain sense indeed optimal.

We use the **first-oder condition** to derive the solution letting $\boldsymbol{x} = (x_1, \ldots, x_n)$ and similar for $\boldsymbol{y}$. Assume the dimension is 1, then

$$\begin{aligned} & 0 = \partial_\beta (\boldsymbol{y} - \beta \boldsymbol{x})^2 = \partial_\beta (\boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{y}^\top \beta \boldsymbol{x} + \beta^2 \boldsymbol{x}^\top \boldsymbol{x}) \\ \Leftrightarrow \quad & 0 = -2\boldsymbol{x}^\top \boldsymbol{y} + 2\beta \boldsymbol{x}^\top \boldsymbol{x} \end{aligned}$$

such that we obtain

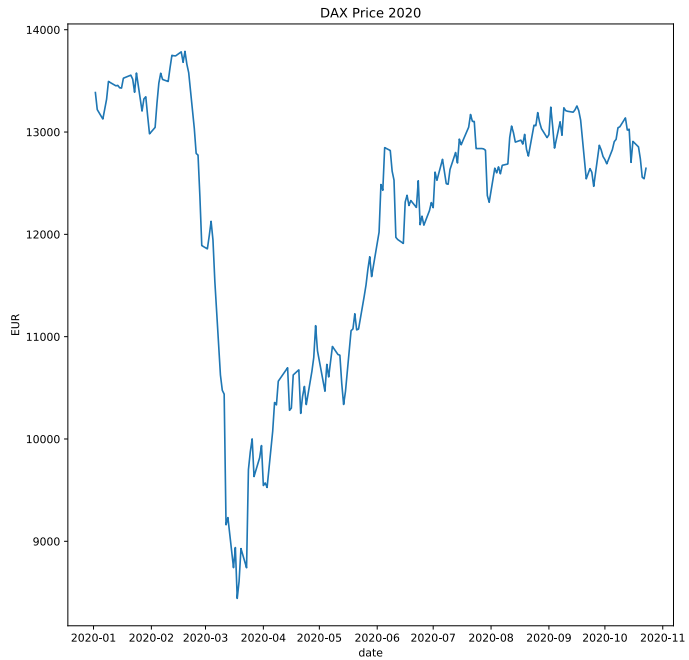
$$\hat{\beta} = (\boldsymbol{x}^\top \boldsymbol{x})^{-1} \boldsymbol{x}^\top \boldsymbol{y}$$

as one possible solution.
Note that typically one considers affine functions of x without mentioning, i.e. one looks at functions $y = \alpha + \beta x$. This can simply be achieved with the linear approach by augmenting $\boldsymbol{x}$ by an additional entry 1.

- Of course many generalizations are possible:
- To higher dimensions: consider data vectors $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, $i = 1, \ldots, n$,
- To nonlinear functions: include $x_i^1, \ldots, x_i^p$ into the covariates
- and many more.

Let us consider a linear regression in python (see jupyternotebook on github).

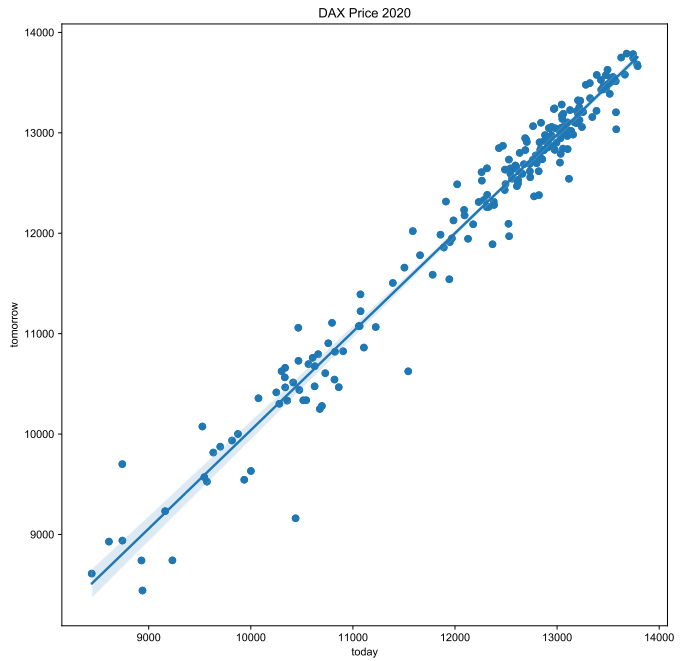
```
import yfinance as yf
import matplotlib.pyplot as plt

DAX = yf.Ticker('%5Egdaxi')
DAX_History = DAX.history(start="2020-01-01", end="2020-10-26")

plt.figure(figsize=(10,10))
plt.plot(DAX_History.index, DAX_History['Close'])

# Linear Regression example: regress tomorrow on today
x = DAX_History['Close'][:-1]    # without the last value
y = DAX_History['Close'][1:]     # without the first value

import numpy as np
from numpy import array
from sklearn.linear_model import Linea

model = LinearRegression()
x = array(x).reshape(-1,1)   # The lir
y = array(y).reshape(-1,1)
model.fit(x, y)  # values in model.int

# Give a very sophisticated plot
import seaborn as sns; sns.set_theme(c
ax = sns.regplot(x=x, y=y)
plt.show()
```



Could we improve this ? Suggestions ?

Let us consider a linear regression in python (see jupyternotebook on github).

```
import yfinance as yf
import matplotlib.pyplot as plt

DAX = yf.Ticker('%5Egdaxi')
DAX_History = DAX.history(start="2020-01-01", end="2020-10-26")

plt.figure(figsize=(10,10))
plt.plot(DAX_History.index, DAX_History['Close'])

# Linear Regression example: regress tomorrow on today
x = DAX_History['Close'][:-1]    # without the last value
y = DAX_History['Close'][1:]     # without the first value

import numpy as np
from numpy import array
from sklearn.linear_model import Linea

model = LinearRegression()
x = array(x).reshape(-1,1)   # The lir
y = array(y).reshape(-1,1)
model.fit(x, y)  # values in model.int

# Give a very sophisticated plot
import seaborn as sns; sns.set_theme(c
ax = sns.regplot(x=x, y=y)
plt.show()
```



Could we improve this ? Suggestions ?

Difference to statistics

- In a statistical approach we start with a **parametric model**:

$$Y_i = \alpha + \beta x_i + \epsilon_i, \quad i = 1, \ldots, n$$

- and assume that $\epsilon_1, \ldots, \epsilon_n$ have a certain structure (for example, i.i.d. and $\mathcal{N}(0, \sigma^2)$).
- Then one can derive (see, e.g. Czado & Schmidt (2011)) **optimal estimators** for α and β. One can also relax the assumptions and gets weaker results.
- So what? What are the advantages of the statistical approach?
- One particular outcome is that we are able to provide **confidence intervals**, **predictive intervals** and **test** hypothesises.

Questions

- What is the definition of Machine Learning?
- Give examples
- Give surprising examples
- Derive the main equation of linear regression
- (do it in 1 dimension first - this goes back to Gauss)
- Write your own python code, providing a linear regression on your favourite stock
- Do this with your least favourite stock
- Can you regress two stocks on each other ?
- Can you predict better the value of the stock tomorrow ?
  (You can also research on this ...)

Generalized Linear Models

We already saw that transforming the input variables suitable might be helpful. This is the idea of a generalized linear model (GLM), see Casella & Berger (2002).

Definition

A GLM consists of three components:

1. Response variables (random) $Y_1, \ldots, Y_n$,
2. a systematic component of the form $\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i$, $i = 1, \ldots, n$,
3. a link function g satisfying

$$\mathbb{E}[Y_i] = g(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i), \quad i = 1, \ldots, n.$$

Regularization of multiple linear regression

- One problem in practice is parsimony of a linear regression: suppose you have many covariates and you want to include only those which are relevant.
- It would be possible to iteratively throw out those parameters which are not significant. This procedure, however is not optimal. Many others have been proposed.
- We concentrate on **continuous** subset selection methods: it is better to introduce a penalty for including two many parameters, which we call regularization. This is moreover a standard procedure for ill-posed problems. We will consider a famous example: the **LASSO** introduced in R. Tibshirani (1996). „Regression Shrinkage and Selection via the Lasso“. In: **Journal of the Royal Statistical Society. Series B (Methodological)** 58.1, pp. 267–288.

LASSO

- The **least absolute shrinkage and selection operator** minimizes the following function

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \frac{1}{n} \parallel \boldsymbol{Y} - \boldsymbol{x}\boldsymbol{\beta} \parallel_2^2 + \lambda \parallel \boldsymbol{\beta} \parallel_1 \right\}.$$

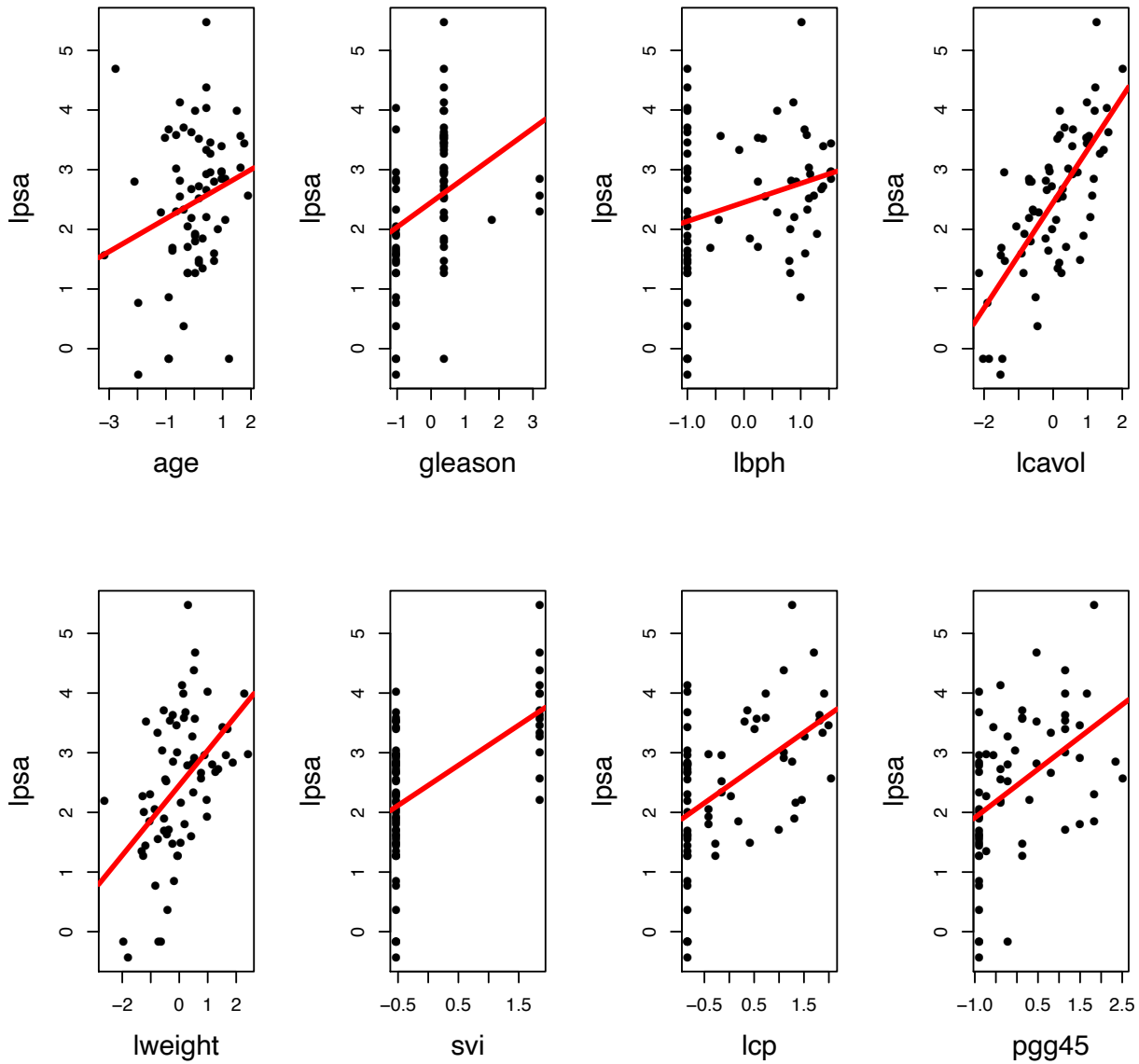
  The parameter λ has to be chosen and allows to vary the level of regularization. Clearly this model prefers to set non-significant parameters to zero.

- Let us illustrate the lasso with an example taken from Chris Franck, http://www.lisa.stat.vt.edu/?q=node/5969. The data stems from Stamey et.al.[3].

- The data describes clinical measures from 97 men about to undergo radical prostatectomy. It is of interest to estimate the relation between the clinical measures and the prostate specific antigen (measures are: lcavol - log (cancer volume), lweight - log(prostate weight volume), age, lbph - log (benign prostatic hyperplasia), svi - seminal vesicle invasion, lcp - log(capsular penetration), Gleason (score), ppg45 - percent Gleason scores 4 or 5, $Y =$lpsa - log(prostate specific antigen))

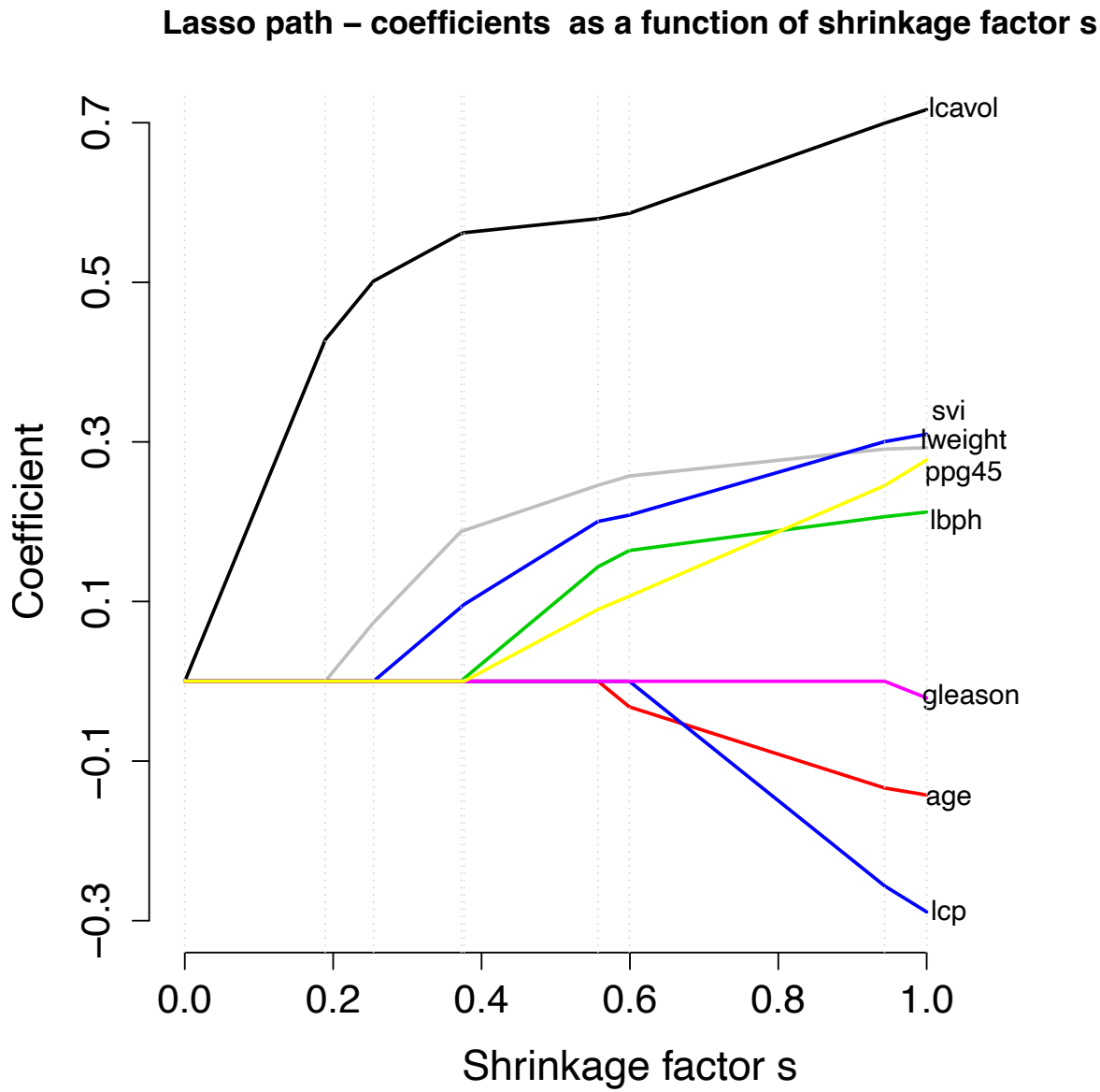
[3] T. A. Stamey et al. (1989). „Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients.“. In: **The Journal of urology** 141.5, pp. 1076–1083.

We start by examining bi-variate regressions.

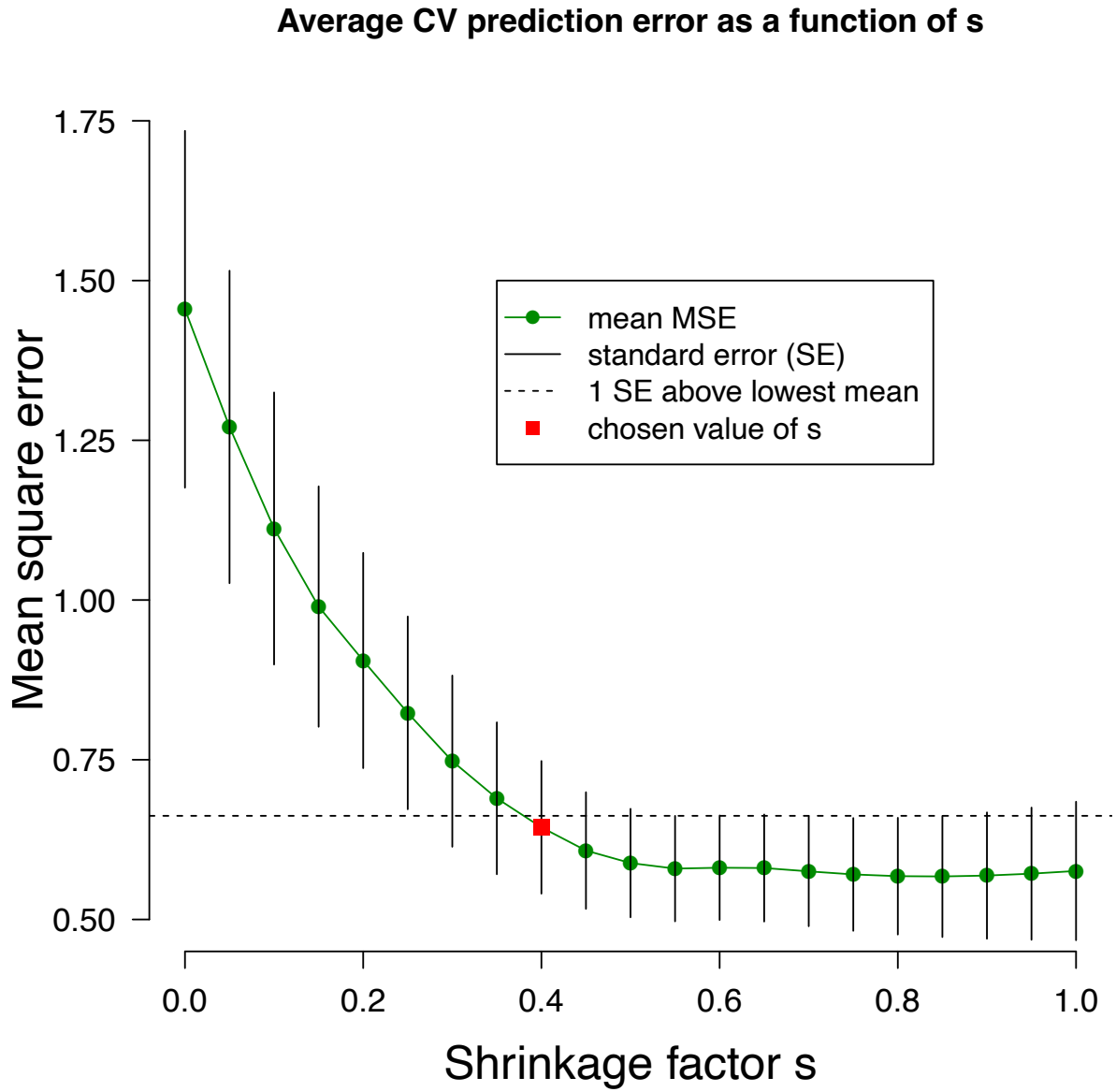


- It is obvious that some variables have fewer impact and some others seem to be more important. The question is how to effectively select those.
- We illustrate how cross-validation may be used in this case. This means we separate the data into a training set and a validation set. The tuning parameter λ is chosen based on the training set and validated on the validation set.
- We use a 10-fold cross validation, ie. the set is split into 10 pieces. Iteratively, each piece is chosen as the validation set while the remaining 9 sets are used to estimate the model.

This is the so-called lasso path. The shrinkage factor is antiproportional to λ.



This is the cross-validation result. A rule of thumb is to select that value of s that is within 1 standard error of the lowest value.



Remarks and Questions

- We see that the optimal choice of λ is far from trivial. Alternative approaches are at hand, compare the recent results by Johannes Lederer and coauthors, J. Lederer and C. Müller (Apr. 2014). „Don't Fall for Tuning Parameters: Tuning-Free Variable Selection in High Dimensions With the TREX". In: **ArXiv e-prints**. eprint: 1404.0541 (stat.ME).

- What is a generalized linear model? Where are the differences to a linear model?
- What is the LASSO ?
- What are the differences to simple least squares ?
- What is an ill-posed problem ? Why do you regulate this ? Why is linear regression an ill-posed problem ?
- What is cross-validation ?

Please note that I encourage you to do research in the internet on words you don't know. Use the references, use google, google scholar, use the katalog at uni freiburg to find online resources for books and literature, use Wikipedia, use the mathematical encyclopdia or discuss with chatGPT ...