Stochastic Machine Learning
01 - Introduction

Thorsten Schmidt

Abteilung für Mathematische Stochastik

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

WS 2020/21

Literature (incomplete, but growing):

- I. Goodfellow, Y. Bengio, and A. Courville (2016). **Deep Learning**. `http://www.deeplearningbook.org`. MIT Press
- D. Barber (2012). **Bayesian Reasoning and Machine Learning**. Cambridge University Press
- R. S. Sutton and A. G. Barto (1998). **Reinforcement Learning : An Introduction**. MIT Press
- G. James et al. (2014). **An Introduction to Statistical Learning: With Applications in R**. Springer Publishing Company, Incorporated. ISBN: 1461471370, 9781461471370
- 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/`
- K. P. Murphy (2012). **Machine Learning: A Probabilistic Perspective**. MIT Press
- CRAN Task View: Machine Learning, available at `https://cran.r-project.org/web/views/MachineLearning.html`
- UCI ML Repository: `http://archive.ics.uci.edu/ml/` (371 datasets)
- Warren B Powell (2011). **Approximate Dynamic Programming: Solving the curses of dimensionality**. Vol. 703. John Wiley & Sons
- A nice resource is `https://github.com/aikorea/awesome-rl`
- There is a lot of examples, code etc. on Josef Teichmanns homepage at ETH Zurich
- Matthew F Dixon, Igor Halperin, and Paul Bilokon (2020). **Machine Learning in Finance**. Springer

Motivation

- **Machine Learning** is nowadays used at many places (Google, Amazon, etc.) with a great variety of applications.
- It is a great job opportunity ! It needs maths and probability !
- Many applications are surprisingly successful (speech / face recognition) and currently people are seeking further applications
- Here we want to learn about the foundations, discuss implications and what can be done by ML and what not.

Organization

The lecture will be available online. We meet every Monday 10.15 on ZOOM and discuss the lectures from last week. I expect active participation from your side, the videos should be switched on.

- The lectures will mix python implementation with theory - so it is now a good time for you to start learning python.
- Every lecture ends with a short set of questions. These questions will be discussed in our session on Monday.
- Homework will be done by projects. You can choose a topic which interests you, and we will provide topics. Groups up to 5 people work on a project, more than one group can also work on the same project.
- We provide a shared bitbucket repository for all projects, such that you share your current work and can profit form the work from others.
- **Lars Niemann** is organizing the projects - please contact him for questions.

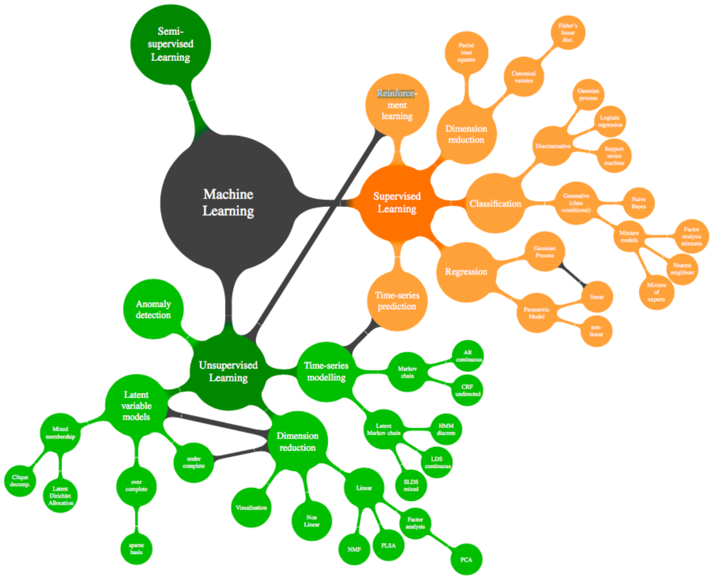
Overview[1]

- Artificial intelligence is the field where computers solve problems.
- It is easy for a computer to solve tasks which can be described formally (Chess, Tic-Tac-Toe). The challenge is to solve a tasks which are hard to describe formally (but are easy for humans: walk, drive a car, speak, recognize people ...)
- The solution is to allow computers to learn from experience and to understand the world by a hierarchy of concepts, each concept defined in terms of its relation to simpler concepts.
- A fixed knowledge-base would be somehow limiting such that we are interested in such attempts where the systems acquire their own knowledge, which we call **Machine Learning**.

[1] This introduction follows closely Goodfellow et.al. (2016).

- First examples of machine learning are **logistic regression** or **naive Bayes** $\rightarrow$ standard statistical procedures (Cesarean delivery / Recognition of Spam, more examples to follow)
- Problems become simpler with a nice representation. Of course it would be nice if the system itself could find such a representation, which we call **representation learning**.
- An example is the so-called **auto-encoder**. This is a combination of an encoder and a decoder. The encoder converts the input to a certain representation and the decoder converts it back again, such that the result has nice properties.
- Speech for example might be influenced by many factors of variation (age, sex, origin, ...) and it needs nearly human understanding to disentangle the variation from the content we are interested in.
- **Deep Learning** solves this problem by introducing hierarchical representations.

- This leads to the following hiearchy:
- AI $\rightarrow$ machine learning $\rightarrow$ representation learning $\rightarrow$ deep learning.



Source: Barber (2012).

Examples of Machine Learning

Some of the most prominent examples:

- LeCun et.al.[2] recognition of handwritten digits. The MNIST Database[3] provides 60.000 samples for testing algorithms. The NIST database is of increased size[4]
- The Viola & Jones face recognition,[5]. This path-breaking work proposed a procedure to combine existing tools with machine-learning algorithms. One key is the use of approx. 5000 learning pictures to train the routine. We will revisit this procedure shortly.
- Imagenet is an image database containing many images classified (cats, cars, etc.)[6]
- Various twitter datasets are available, for example for learning to detect hate speach.
- `Kaggle`[7] is a platfrom where computational competitions are hosted. It also provides many many data examples with it.
- Datasets for machine-learning research on Wikipedia[8].

[2] Y. LeCun et al. (1998). „Gradient-based learning applied to document recognition“. In: **Proceedings of the IEEE** 86.11, pp. 2278–2324.

[3] `http://yann.lecun.com/exdb/mnist/`

[4] `https://www.nist.gov/srd/nist-special-database-19`

[5] P. Viola and M. Jones (2001). „Robust Real-time Object Detection“. In: **International Journal of Computer Vision**. Vol. 4. 34–47.

[6] `http://image-net.org`

[7] www.kaggle.com

[8] `https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research`

- Speech recognition has long been a difficult problem for computers (first works date to the 50's) and only recently been solved with high computer power. It may seem surprising, that mathematical tools are at the core of these solutions. Let us quote Hinton et.al.[9]

  > *Most current speech recognition systems use hidden Markov models (HMMs) to deal with the temporal variability of speech and Gaussian mixture models (GMMs) to deter- mine how well each state of each HMM fits a frame or a short window of frames of coefficients that repre- sents the acoustic input. (...)*
  > *Deep neural networks (DNNs) that have many hidden layers and are trained using new methods have been shown to outperform GMMs on a variety of speech recognition benchmarks, sometimes by a large margin*

  So, one of our tasks will be to develop a little bit of mathematical tools which we will need later. Most notably, some of the mathematical parts can be replaced by deep learning, which will be of high interest to us.

[9] Geoffrey Hinton et al. (2012). „Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: **IEEE Signal Processing Magazine** 29.6, pp. 82–97.

Questions

- Was is artificial intelligence ?
- Was is machine learning ?
- Do you know what a neural network is (look for the history in the internet)?
- What are shallow / deep networks ?
- What are the applications which you find most exciting ?
- What are the applications that you think will have the largest impact on our future?
- Research a bit yourself: look for datasets, look for latest applications etc.

1. Introduction $\rightarrow$ Machine learning basics

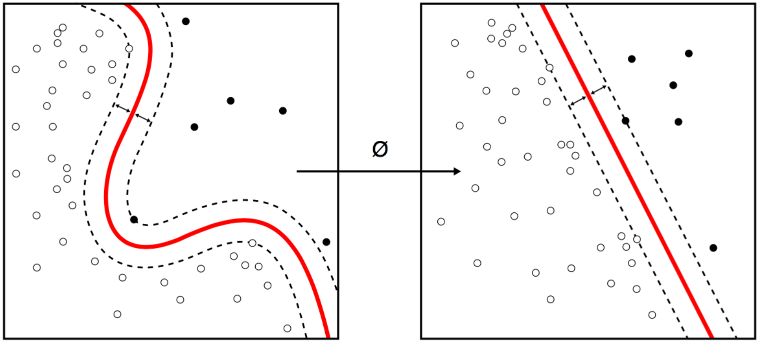
Types of machine learning:

- **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.

- **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. In this sense the machine learning algorithm shall itself find a characteristics which divides the data into suitable subsets.

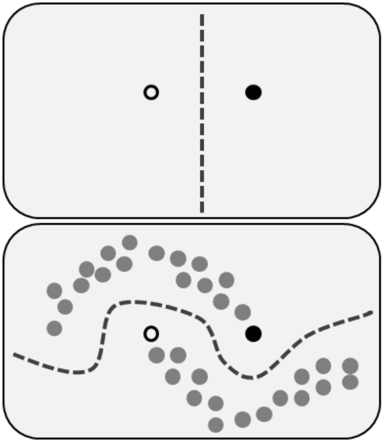


Picture by: Alisneaky, svg version by User:Zirguezi - Own work,
CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=47868867

Some examples

- Analysis of genomic data
- Density estimation
- Clustering
- Principal component analysis

- **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 give one example

Reinforcement learning

is quite different from the above examples.

- First: time matters, the problem depends on time ! Observations accumulate over time.
- There is no supervisor but a reward signal measuring the quality of the decision.
- The approach utilizes a probabilistic framework: **Markov decision processes.**
- Examples are: drive a car, optimally manage a portfolio, program a roboter, playing a game (Pong, breakout for example) ...

- A **stochastic process** is simply a collection of rvs: $(X_t)_{t\in\mathbb{N}}$.
- It is called **Markovian**, if for all Borel sets A, the transition probabilities do not depend on the full history, but only on the last value from the history:
$$P(X_{t+1} \in A | X_s : s \leq t) = P(X_{t+1} \in A | X_t), \quad t \in \mathbb{N}.$$
- A **Markov decision process** consists of a process X and decisions $(d_t)_{t\in\mathbb{N}}$, such that the transition probalities may depend on d. The behaviour of the process is described by the *transition probabilities*
$$P(X_{t+1} \in A | X_t, d_t), \qquad t \in \mathbb{N}$$
and the *policy*
$$P(d_t \in A | X_t), \quad t \in \mathbb{N}.$$
- In the reinforcement learning, the probabilities are **not** known and have to be learned!
- In a nutshell, we proceed iteratively through time.
- At time t, we *observe* X_t, get a reward $U(X_t)$ and are able to make a decision d_t which influences the state at time $t+1$, X_{t+1}.
- A *policy* describes the decision given the state. It can be stochastic or deterministic.
- While initially the environment is unknown, the system gathers information through its interactions with the environment and improves its policy.

A quite related area is **Statistical Learning**. This new area of statistics is quite related to machine learning and we will study a number of relevant problems[10].

- 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.

[10] 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.

Questions

- What is the essential difference of static and dynamic settings?
- What is a stochastic process. What is a Markov process. Do you know a process which is not Markovian?
- What is the difference to a Markov decision process.
- Simulate a Markov process in Python.
- Simulate a Markov decision process in Python.
- Choose a target, a loss function (or a reward) and try to find the optimal decision process.
- What is statistical learning.

Introduction → Machine Learning Basics

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 $X_1, X_2, \dots, X_t$ 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**. 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. One could also look for the accumulated utility

$$\sum_{t=1}^{T} U(X_t).$$

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[11]

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

and minimize over β.

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

[11] 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.

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

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

such that we obtain

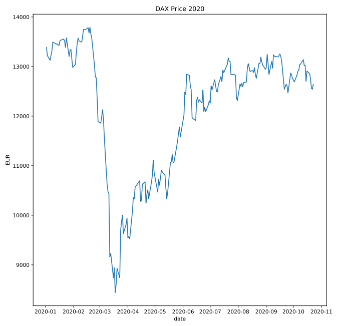
$$\hat{\beta} = (\boldsymbol{x}^\top \boldsymbol{x})^{-1} \boldsymbol{x}^\top \boldsymbol{y}.$$

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.

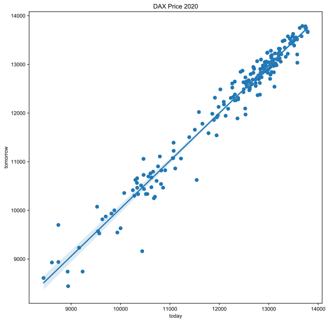
```
import yfinance as yf
import matplotlib.pyplot as plt

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 lin
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.

```
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 lin
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 ?

What is the 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 $\mathcal{N}(0, \sigma^2)$). The 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) $Y_1, \dots, Y_n$,
2. a systematic component of the form $\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i$, $i = 1, \dots, n$,
3. a link function g satisfying

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

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

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

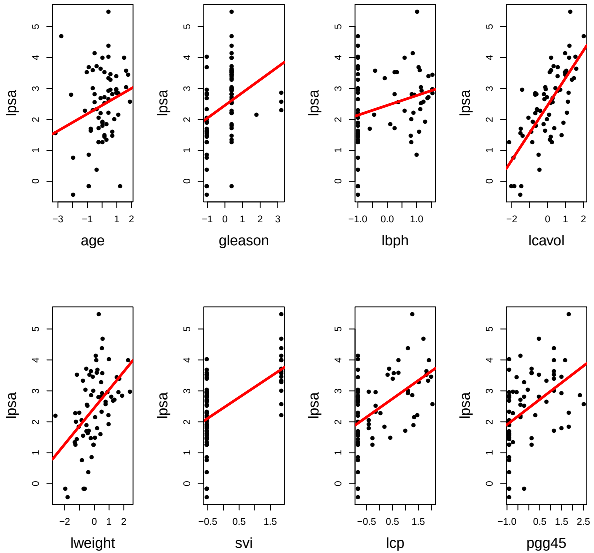
  The parameter λ has to be chosen and allows to vary the level of regularization. Clearly this model prefers to set non-significant parameters to zero.

- 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.[12].

- 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))

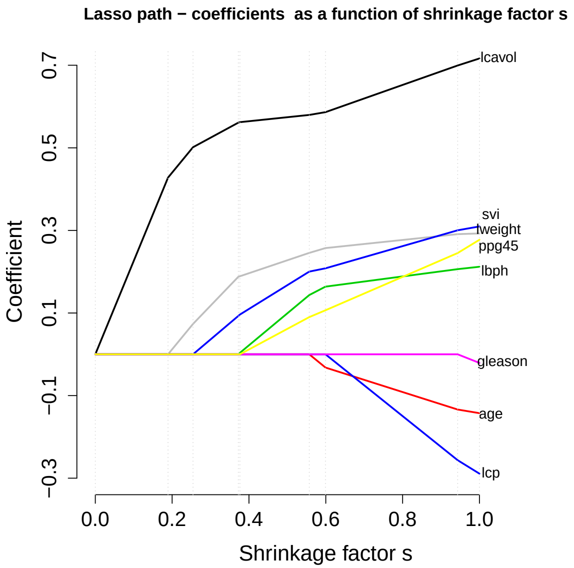
[12] T. A. Stamey et al. (1989). „Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients.“. In: **The Journal of urology** 141.5, pp. 1076–1083.

We start by examining bi-variate regressions.

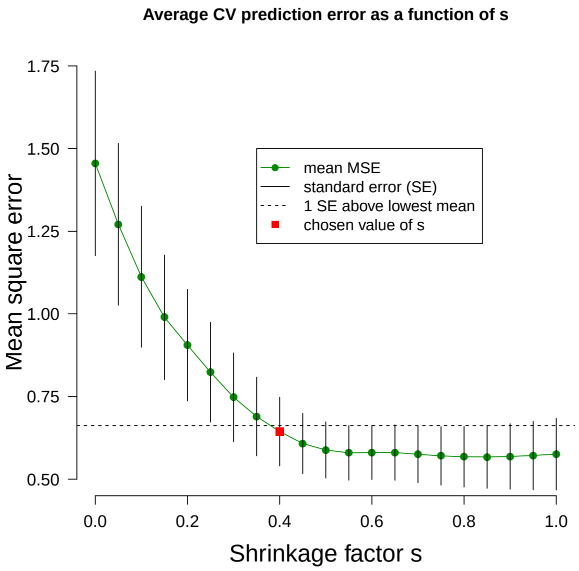


- It is obvious that some variables have fewer impact and some others seem to be more important. The question is how to effectively select those.
- We illustrate how cross-validation may be used in this case. This means we separate the data into a training set and a validation set. The tuning parameter 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 ...

Logistic regression

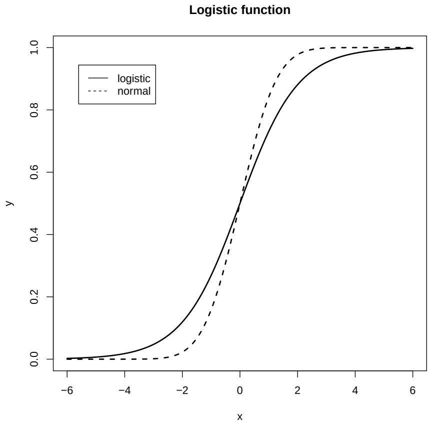
- One important regression approach for **classification** is logistic regression.
- We start by considering **simple** logistic regression, i.e. the classification into **two** classes. In this case, the response is always binary.
- One therefore needs to transform the whole real line to $[0, 1]$ and two approaches are common: first, via the logistic function

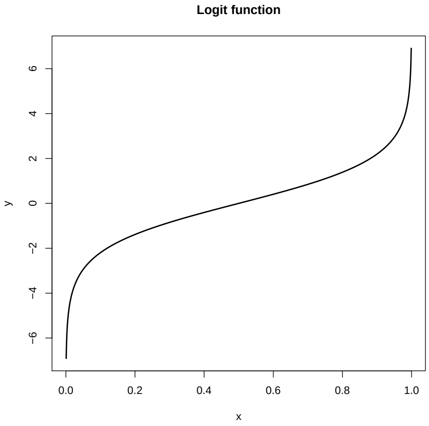
$$\sigma(x) = \frac{e^x}{1 + e^x}.$$

The most common way is to transform y via

$$\sigma^{-1}(p) = \text{logit}(p) = \log \frac{p}{1 - p},$$

the so-called **logit** function.
- Second, by a cumulative distribution function (when this is Φ - standard normal - this approach is called **probit** model.


Logistic function
logistic
normal
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4
6


Logit function
y
x
6
4
2
0
−2
−4
−6
0.0
0.2
0.4
0.6
0.8
1.0

Definition (Logistic regression)

A logistic regression is the generalized linear model where

$$\operatorname{logit}(p_i) = \alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i, \qquad i = 1, \dots, n.$$

Note that this model is equivalent to

$$p_i = \frac{\exp(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i)}{1 + \exp(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i)}.$$

The observations $y_1, \dots, y_n$ are binary, hence take values in $\{0, 1\}$ and are assumed to be i.i.d. Bernoulli with $P(y_i = 1) = p_i = p_i(\boldsymbol{x}_i)$.
A nice source explaining the depth of logistic regression and their various applications is[13].

[13] Ronald Christensen (2006). **Log-linear models and logistic regression**. Springer Science & Business Media.

The most common estimation method used is **maximum-likelihood**. We take a small detour towards this exciting statistical concept going back to Sir Ronald Fisher.

Maximum-likelihood

- A **statistical model** is given by a family of probability measures $(P_\theta)_{\theta\in\Theta}$ on a common measurable space $(\Omega, \mathscr{F})$. It is typically called **parametric**, if Θ is of finite dimension.
- The **likelihood**-function for the observation E is given by

  $$L(\theta) = P_\theta(E)$$

  If $P_\theta(E) = 0$ for all $\theta \in \Theta$ one proceeds via the density: assume $P_\theta \ll P^*$ for all $\theta \in \Theta$ and denote the densities by $f_\theta := {}^{dP_\theta}/_{dP^*}$. Then, for the observation x,

  $$L(\theta) = f_\theta(x).$$
- This looks complicated, but is in most cases quite simple: consider i.i.d. random variables $X_1, \dots, X_n$ with common density f_θ. Then P^* is clearly the Lebesgue-measure. Due to the i.i.d.-property,

  $$L(\theta) = \prod_{i=1}^n f_\theta(x_i).$$

Definition

Any maximizer $\hat{\theta}$ of the likelihood-function is called maximum-likelihood estimator for the model $(P_\theta)_{\theta \in \Theta}$.

In the above example, we need to maximize $\prod_{i=1}^{n} f_\theta(x_i)$, which is typically infeasible. One therefore considers the log-likelihood function

$$\ell(\theta) := \ln L(\theta)$$

which is often much easier to maximize. Typically one can apply first-order conditions or needs to solve numerically.

Example (ML for the normal distribution)

Consider $X_i \sim \mathcal{N}(\mu, 1)$. Then the density is

$$f_\theta(x) = \frac{1}{\sqrt{2\pi}} \exp\Big(- \frac{1}{2}(x-\mu)^2 \Big).$$

We obtain the log-likelihood function

$$l(\theta) = \text{const.} - \frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^2.$$

The first derivative is

$$\partial_\mu l(\theta) = \sum_{i=1}^{n} x_i - n\mu \overset{!}{=} 0$$

and we obtain the maximum-likelihood estimator (second derivative is < 0)

$$\hat{\mu} = \bar{x} = \frac{\sum_{i=1}^{n}}{n} x_i.$$

Exercise: compute the ML estimator for σ ! Read Czado & Schmidt (2011) on ML-estimation and further estimation procedures.

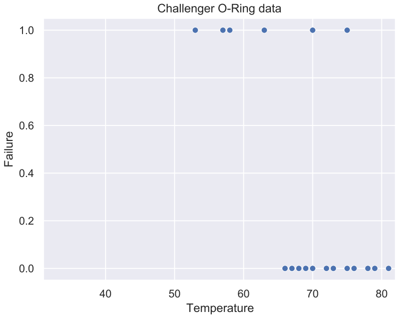
Maximum-Likelihood for the logistic regression

- For the logistic regression, where $y_1, \ldots, y_n$ are Bernoulli, we obtain the likelihood function
$$L(\boldsymbol{y}) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}.$$
- Maximization has to be done numerically, eg. by gradient descent or by weighted least squares.
- Asymptotic distributions are available, such that we can test approximately several hypothesis, like for example $\beta_i = 0$ or $\alpha = 0$.

Back to logistic regression. We look at the by now infamous Challenger[14] O-ring data set (taken from Caslla & Berger (2002))

1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1
53	57	58	63	66	67	67	67	68	69	70	70	70	70	72	73	75	75

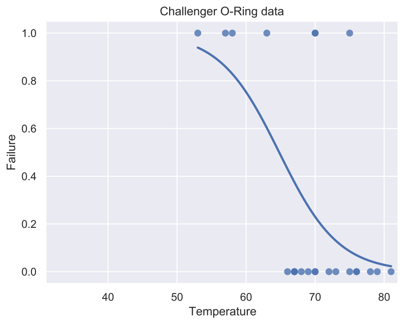
The table reports failures with associated temperature.



[14] See https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

x = np.array([53,57,58,63,66,67,67,67,68,69,70,70,70,70,72,73,
              75,75,76,76,78,79,81]).reshape(-1, 1)
y = np.array([1,  1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
              0, 1, 0, 0, 0, 0, 0]).reshape(-1, 1)

# logistic regression model
logreg = LogisticRegression().fit(x, y)

print(logreg.intercept_, logreg.coef_[0])
#[0.52055518] [-0.02100215]

import seaborn as sns
sns.set_theme(color_codes=True)
sns.regplot(x=x, y=y, logistic=True)
plt.show()
```

The estimated probability
for a failure at 31 degree is 0.9996088.

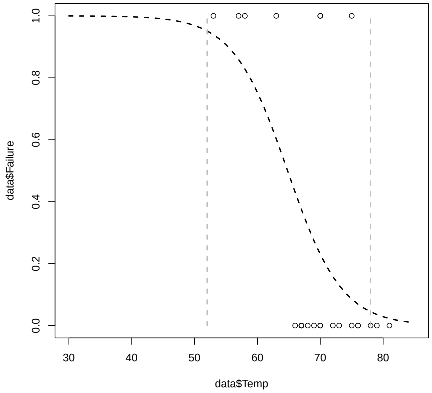


- Logistic regression naturally classifies the data into two fields: the ones with probability above 0.5, where we would optimally decide for outcome one and the ones with probability below 0.5, where we would decide for outcome 0.
- Hence, we obtain a **decision boundary**, given by the hyperplane

$$\alpha + \boldsymbol{\beta x} = 0.$$

- If the decision boundary separates the two groups, then the data is called **linearly separable.** Note that this can not be achieved in the Challenger dataset.
- Note that the logistic regression also provides probabilities of false decisions: at the boundary this is 50/50, but further out the probability of a false decision decrease. **Significant** decisions requires the probability of a false decision to be below a significance level, e.g. $\alpha = 0.05$ or $\alpha = 0.01$.

With significance level $\alpha = 0.05$ obtained decision boundaries.



Load the python example[15] from the homepage and revisit the above steps. Try your own examples.

- The likelihood-function has to be maximized numerically.
- A first-order iterative scheme is the **gradient-descent** algorithm. Look this algorithm up and recall its properties and functionality.

[15] Called 01_05_logistic_regression.py

Questions

- What is the difference between logistic regression and regression?
- What are the logit and probit functions ?
- What is maximum-likelihood?
- Compute the maximum-likelihood estimator for an exponential distribution.
- Look up the challenger catastrophe and watch Richard Feynman's famous speach.