

# Stochastic Machine Learning

## Chapter 01 - Introduction, part III

Thorsten Schmidt

Abteilung für Mathematische Stochastik

[www.stochastik.uni-freiburg.de](http://www.stochastik.uni-freiburg.de)  
thorsten.schmidt@stochastik.uni-freiburg.de

SS 2024

## Short recap

In the last lecture we have learnt:

- ▶ Types of ML
- ▶ What is machine learning precisely (definition!) with an example
- ▶ Linear Regression, example (DAX)
- ▶ Generalized linear models (GLM)
- ▶ Regularization in linear models (LASSO)

## 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

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

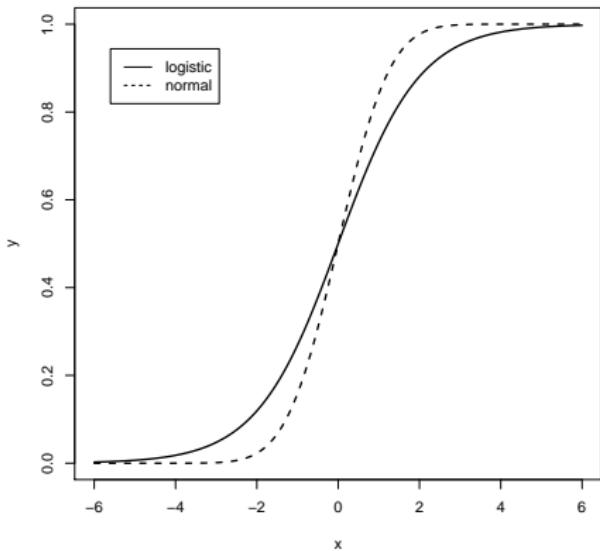
The most common way is to transform  $y$  via

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

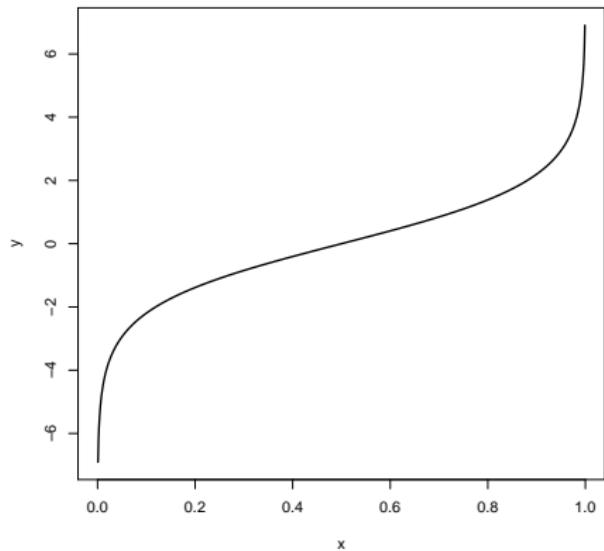
the so-called **logit** function.

- ▶ Second, by a cumulative distribution function (when this is  $\Phi$  - standard normal - this approach is called **probit** model).

**Logistic function**



**Logit function**



## Definition (Logistic regression)

A logistic regression is the generalized linear model where

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

Note that this model is equivalent to

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

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  $P(y_i = 1) = p_i = p_i(\mathbf{x}_i)$ .

A nice source explaining the depth of logistic regression and their various applications is<sup>1</sup>.

---

<sup>1</sup>Christensen (2006).

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  $(\Omega, \mathcal{F})$ . It is typically called **parametric**, if  $\Theta$  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  $f_\theta$ . 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  $X_i \sim \mathcal{N}(\mu, 1)$ , hence  $\theta = \mu$ . Then the density is

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

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

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

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

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

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

- ▶ Observe that in this case, ML is equivalent to least-squares – under the normality assumption this comes out very naturally. Under other assumptions, least-squares will be different from maximum likelihood.

## Maximum-Likelihood for the logistic regression

- ▶ For the logistic regression, where  $y_1, \dots, y_n$  are Bernoulli, we obtain the likelihood function

$$L(\mathbf{y}) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}.$$

- ▶ 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  $\alpha = 0$ .
- ▶ We revisit a number of numerical methods for this approach, since they also build the basis for our learning techniques later.

## Gradient Descent

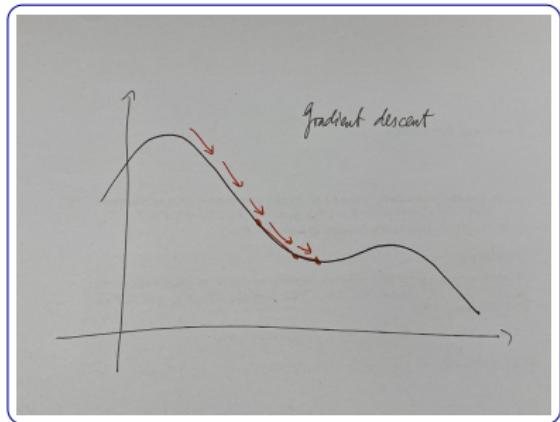
- ▶ Proposed by Augustin - Louis Cauchy in 1847.

---

<sup>2</sup>[https://en.wikipedia.org/wiki/Gradient\\_descent](https://en.wikipedia.org/wiki/Gradient_descent)

# Gradient Descent

- ▶ Proposed by Augustin - Louis Cauchy in 1847.



<sup>2</sup>[https://en.wikipedia.org/wiki/Gradient\\_descent](https://en.wikipedia.org/wiki/Gradient_descent)

# Gradient Descent

- ▶ Proposed by Augustin - Louis Cauchy in 1847.
- ▶ Gradient descent is an iterative procedure which iterates

$$x_{i,k+1} = x_{i,k} - \alpha_k \frac{\partial f}{\partial x_{i,k}}, \quad i = 1, \dots, d$$

through all dimensions  $i$ . Here  $\alpha_k > 0$  is the **learning rate**

- ▶ In vector notation,  
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) =: \mathbf{x}_k + \alpha_k \mathbf{d}_k$$
with **step size**  $\alpha_k$  and **direction**  $\mathbf{d}_k$
- ▶ When the function is convex, gradient descent converges to a global minimum, otherwise we might have also local minima.
- ▶ Wikipedia<sup>2</sup> gives a nice explanation, python examples and further information. See also Grippo and Sciandrone (2023) for a thorough treatment.

---

<sup>2</sup>[https://en.wikipedia.org/wiki/Gradient\\_descent](https://en.wikipedia.org/wiki/Gradient_descent)

## Line search

- ▶ The gradient method has two steps: computing the gradient and choosing the step size.
- ▶ The step size is often chosen via **linear search**: Given a start value  $\mathbf{x}_k$  and a direction  $\mathbf{d}_k$  we minimize  $f$  in the direction of  $\mathbf{d}_k$  (along the affine subspace  $\mathbf{x}_k + \alpha \mathbf{d}_k$ ,  $\alpha \in \mathbb{R}$ ).
- ▶ More precisely, define

$$\phi(\alpha) := f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$

- ▶ For minimization we aim at

$$0 = \phi'(\alpha) = \nabla f(\mathbf{x}_k + \alpha \mathbf{d}_k)^\top \mathbf{d}_k \quad (1)$$

- ▶ In the quadratic case, the optimal step size can be computed: assume  $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top Q \mathbf{x} - \mathbf{c}^\top \mathbf{x}$ . Then,  $\nabla f(\mathbf{x}) = \mathbf{x}^\top Q - \mathbf{c}$  and hence

$$\nabla f(\mathbf{x} + \mathbf{y}) = \mathbf{x}^\top Q - \mathbf{c} + \mathbf{y}^\top Q = \nabla f(\mathbf{x}) + \mathbf{y}^\top Q.$$

For us more important is

$$\nabla f(\mathbf{x}_k + \alpha \mathbf{d}_k)^\top \mathbf{d}_k = \nabla f(\mathbf{x}_k)^\top \mathbf{d}_k + \alpha \mathbf{d}_k^\top Q \mathbf{d}_k.$$

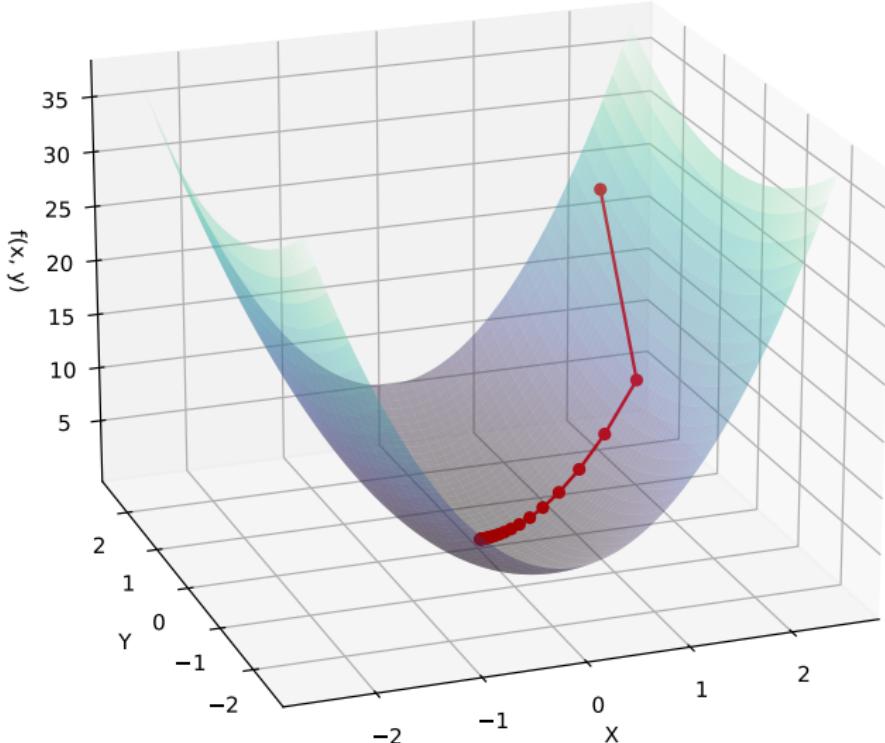
Choosing

$$\alpha_k = -\frac{\nabla f(\mathbf{x}_k)^\top \mathbf{d}_k}{\mathbf{d}_k^\top Q \mathbf{d}_k}$$

therefore does the job and (1) is satisfied. This equation therefore gives the optimal step size in the quadratic case.

- ▶ Of course it should be checked if  $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ , otherwise this point would not be selected. Note that this results in a monotone decreasing sequence. Thus, if  $f$  is continuously differentiable and the set of stationary points (where the derivative vanishes) is compact, we obtain convergence against a stationary point.

## Line search in a gradient descent



# Convergence

A suitable convergence result is the following (see Grippo and Sciandrone (2023), Proposition 11.1). We start from an initial point  $b\mathbf{x}_0$  and denote the level set by

$$\mathcal{L}_0 := \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq f(\mathbf{x}_0)\}.$$

## Proposition

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be continuously differentiable and assume that  $\mathcal{L}_0$  is compact. Assume that the line search satisfies

- (a)  $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$  if  $\nabla f(\mathbf{x}_l) \neq 0$ ,
- (b) if  $\nabla f(\mathbf{x}_k) \neq 0$  for all  $k$ ,

$$\lim_{k \rightarrow \infty} \frac{\nabla f(\mathbf{x}_k)^\top \mathbf{d}_k}{\|\mathbf{d}_k\|} = 0.$$

Then, one of the following claims hold:

- (i) There exists an index  $k^*$  such that  $b\mathbf{x}_{k^*} \in \mathcal{L}^0$  and  $\nabla f(\mathbf{x}_{k^*}) = 0$
- (ii) There exist limit points and every limit points of  $(\mathbf{x}_k)$  is a stationary point.

Since compactness of  $\mathcal{L}_0$  follows from continuity of  $f$ , and the sequence is monotone, hence converging, we obtain the result quite immediate.

## Convergence rate

Again from Grippo and Sciandrone (2023) we obtain a nice convergence rate - but typically under much stronger assumptions. We consider the case with constant step size. The assumption (3) implies that  $f$  is strongly convex.

### Proposition

Assume that  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice continuously differentiable and there exist  $0 < m < M$  such that

$$m \| u \|^2 \leq u^\top \nabla^2 f(z) u \leq M \| u \|^2, \quad z, u \in \mathbb{R}^n \quad (2)$$

Moreover, assume that  $\alpha_k = \eta$  is constant with

$$0 < \eta < \frac{2}{M}$$

and that  $\nabla f(x_k) \neq 0$  for all  $k$ . Then  $(x_k)$  converges to the unique minimum  $x^*$  with at least linear rate and we have

$$\| x_{k+1} - x^* \| \leq q \| x_k - x^* \|$$

where  $q = \max\{|1 - \eta m|, |1 - \eta M|\}$ .

One can also conclude on an optimal step size  $\eta^* = 2/M+m$  which minimizes  $q$ .

## Proposition

Assume that  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is twice continuously differentiable and there exist  $0 < m < M$  such that

$$m \| u \|^2 \leq u^\top \nabla^2 f(z) u \leq M \| u \|^2, \quad z, u \in \mathbb{R}^n \quad (3)$$

Moreover, assume that  $\alpha_k = \eta$  is constant with

$$0 < \eta < \frac{2}{M}$$

and that  $\nabla f(x_k) \neq 0$  for all  $k$ . Then  $(x_k)$  converges to the unique minimum  $x^*$  with at least linear rate and we have

$$\| x_{k+1} - x^* \| \leq q \| x_k - x^* \|$$

where  $q = \max\{|1 - \eta m|, |1 - \eta M|\}$ .

The proof of this result uses basic estimates: the key step is that

$$\nabla f(y) = \nabla f(x) + \int_0^1 \nabla^2 f(x + t(y - x)) (y - x) dt,$$

which allows to estimate the first derivative by a bound on the second derivative and a linear term,

$$\| \nabla f(y) - \nabla f(x) \| \leq M \| y - x \|.$$

Since  $\nabla f(x^*) = 0$  we have  $\nabla f(x_k) = \nabla f(x_k) - \nabla f(x^*) = H_k(x_k - x^*)$  where

$$H_k = \int_0^1 \nabla^2 f(x^* + t(x_k - x^*)) dt$$

Using our iteration we obtain

$$\| x_{k+1} - x^* \| \leq \| I_n - \eta H_k \| \cdot \| x_k - x^* \|,$$

and estimating the first term with the eigenvalues of the matrix gives  $q$  and hence the result.

## Computing the gradient

- ▶ A key point is therefore how to compute the gradient.
- ▶ First, since we often have  $f(\mathbf{x}) = \sum_{j=1}^N f_j(\mathbf{x}) = \sum_{j=1}^n f_j(g_j(\mathbf{x}))$  we can use the chain rule and linearity.
- ▶ Clearly

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^N \frac{\partial f_j}{\partial x_i}$$

and we concentrate on  $f_j$  in the following.

- ▶ Recall the chain rule:  $y = g(x)$ ,  $z = f(g(x))$  such that

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)} \frac{\partial g(x)}{\partial x}.$$

- ▶ Computing of the derivative can be done backward (using **backpropagation**): we start from the last layer and compute  $\nabla g$ . Then we proceed back and compute  $\nabla f$  and aggregate and so on ....

## Examples

With this scheme we can solve:

- ▶ Linear regression
- ▶ Maximum-Likelihood estimation
- ▶ Train neural networks. Here one often splits the data in test and training data, for regularization, as we shortly see.

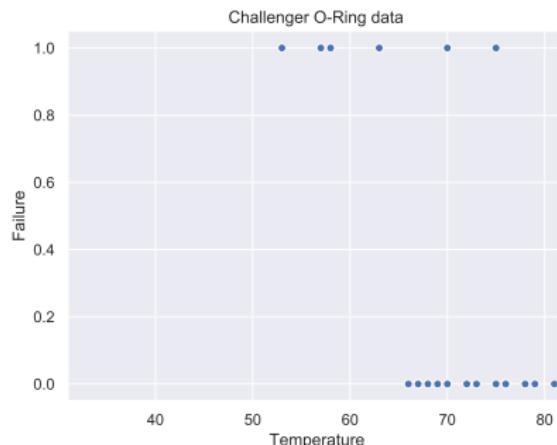
Let us come to an interesting example.

Back to logistic regression. We look at the Challenger<sup>3</sup> O-ring data set (taken from Casella & Berger (2002))

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

The table reports failures with associated temperature.

*The engineers argued that they did not have enough data to determine whether the O-rings would seal at temperatures colder than 53° F (12° C), the coldest launch of the Space Shuttle to date. The air temperature on January 28 was (...) 26° F (-3° C) at the scheduled launch time of 9:38 a.m – wikipedia*



<sup>3</sup>See [https://en.wikipedia.org/wiki/Space\\_Shuttle\\_Challenger\\_disaster](https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster): 1986 the space shuttle Challenger broke apart, shortly after its start.

```

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, 1, 0, 0, 0, 0, 0, 0]).reshape(-1, 1)

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

print(logreg.intercept_, logreg.coef_)
#[14.86182022] [-0.22949881]

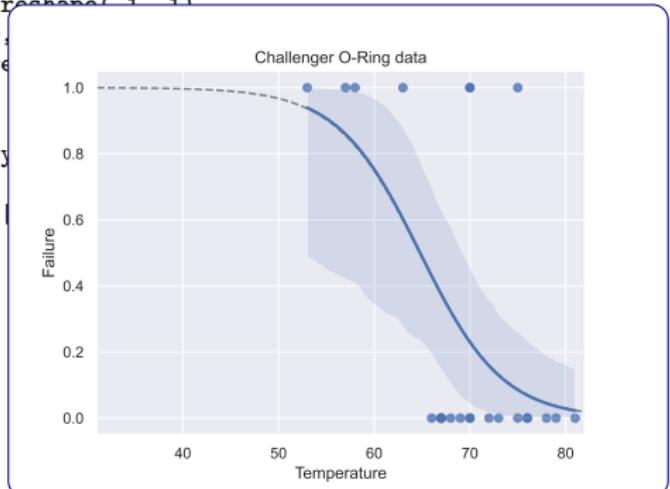
import seaborn as sns
sns.set_theme(color_codes=True)
sns.regplot(x=x, y=y, logistic=True)
plt.show()

```

The estimated probability  
for a failure at 31 degree is

$$\frac{1}{1 + \exp(14.86182022 - 0.22949881 \cdot 31)} \approx 0.999568$$

PS: If you do the implementation with R - you get slightly different values.

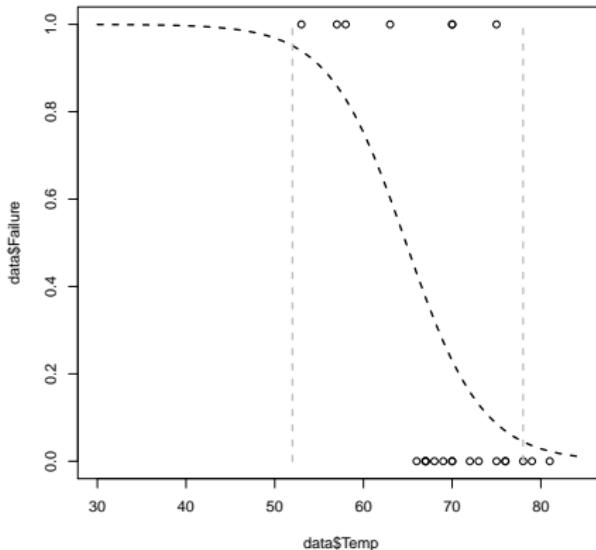


- ▶ Logistic regression naturally classifies the data into two fields: the ones with probability above 0.5, where we would optimally decide for outcome one and the ones with probability below 0.5, where we would decide for outcome 0.
- ▶ Hence, we obtain a **decision boundary**, given by the hyperplane

$$\alpha + \beta x = 0.$$

- ▶ 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<sup>4</sup> from the github page 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.

---

<sup>4</sup>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 speech.

## The imitation game - der Turing Test

- ▶ Ursprünglich 1960 von Turin in komplexerer Form (imitation game) eingeführt, gilt er heute als einer der Tests auf künstliche Intelligenz
- ▶ Der einfache Turing Test ist eine Kommunikation mit zwei Partnern A und B und man muss am Anschluss entscheiden wer die Maschine ist. Typischerweise würde man viele Personen befragen und erwarten dass man nicht in der Lage ist die beiden Personen zu unterscheiden, die Wahrscheinlichkeit die Maschine zu identifizieren also nicht über 50% betragen sollte.
- ▶ Es gibt einige berühmte Beispiele. Starten wir mit
- ▶ ELIZA (1965 - Joseph Weizenbaum) ein Chatbot der im Emacs integriert ist. Probieren wir diesen doch einmal aus.



I am the psychotherapist. Please, describe your problems. Each time you are finished talking, type RET twice.

i am bad at maths

Why do you say you are bad at maths?

my teacher tells me so

Tell me about your teacher.

oh - he is crazy. even if he is a probabilist, he is teaching us machine learning. he thinks computers are intelligent

You have your mind on computers, it seems.

a little bit.

Is it because of your plans that you say a little bit?

yes, my plan is to become a famous mathematician

Maybe your life has something to do with this.

my life ? do you think i am not doing enough maths ? should i do more homework ?

What makes you believe that?

because my teacher tells me i should do all the homework in the class. but it is so much.

Is it because your teacher tells you you should do all the homework in the class but it is so much that you came to me?

because it is so much.

Does it bother you that it is so much?

yes. it is tough to do all the homework on my own

Earlier you said your teacher tells you you should do all the homework in the class but it is so much?

yes - that is right. the teacher gives a lot of homework and it is so much. i want to go out and do skateboard

When did you first know that is right the teacher gives a lot of

## Weitere Beispiele

- ▶ In den 1980s wurde Cleverbot entwickelt, der erste Tests gut überstand
- ▶ 2017 gab es auf der ART Basel ein Projekt in welchem KI-generierte Bilder mit denjenigen von zeitgenössischen Künstlern ausgestellt wurden. Ein großer Teil der Zuschauer:innen bevorzugte die KI-generierten Bilder (durch ein Creative Adversarial Network erzeugt).
- ▶ Schauen Sie auch einmal an meinem Büro vorbei ...



Nº 4

Artificial Art

Telmo  
Zemba

T. Sennett

## Chat GPT

- ▶ Nun können wir natürlich auch mit Chat GPT einen Turing Test machen. Los gehts.

-  Christensen, Ronald (2006). **Log-linear models and logistic regression.** Springer Science & Business Media.
-  Grippo, Luigi and Marco Sciandrone (2023). **Introduction to Methods for Nonlinear Optimization.** Vol. 152. Springer Nature.