FastTrack–Statistical learning

Mathieu Ribatet—Full Professor of Statistics

What is Machine Learning?

- A machine learning algorithm is an algorithm that is able to learn from the data.
- According to Mitchell (1997),

> a program is said to learn from data $\mathscr{D}$ with respect to some class of objectives $\mathscr{O}$ and loss function ℓ, if its performance at objectives in $\mathscr{O}$, as measured by ℓ, improves with experience $\mathscr{D}$.

What is Machine Learning?

- A machine learning algorithm is an algorithm that is able to learn from the data.
- According to Mitchell (1997),

> a program is said to learn from data $\mathscr{D}$ with respect to some class of objectives $\mathscr{O}$ and loss function ℓ, if its performance at objectives in $\mathscr{O}$, as measured by ℓ, improves with experience $\mathscr{D}$.

☞ This "definition" highlights important points : **task**, **measure of performance** and **experience**.

Motivations for Machine Learning

Why do we need computer to learn rather than using dedicated programs for a given task?

- Too complicated to program: speech, image recognition, ...
- large and complex datasets: genome, text analysis, ...
- updating abilities: a written program will never change its behaviour.

How do machine learn to take decision? (statistical way)

- Well since machines are deterministic, it just amounts to define what we shall call a decision/regression/discrimination rule, i.e., a (mesurable) function

$$f : \mathcal{X} \longrightarrow \mathcal{Y}$$
$$x \longmapsto f(x).$$

- Most often, the decision rule is parametrized so that we rather should write $f(x) = f(x; \theta)$ for some parameter value $\theta \in \Theta$.

How do machine learn to take decision? (statistical way)

- Well since machines are deterministic, it just amounts to define what we shall call a decision/regression/discrimination rule, i.e., a (mesurable) function

$$f : \mathcal{X} \longrightarrow \mathcal{Y}$$
$$x \longmapsto f(x).$$

- Most often, the decision rule is parametrized so that we rather should write $f(x) = f(x; \theta)$ for some parameter value $\theta \in \Theta$.
- The question is now

  "How do we find the right value for θ?".

How do machine learn to take decision? (statistical way)

- Well since machines are deterministic, it just amounts to define what we shall call a decision/regression/discrimination rule, i.e., a (mesurable) function

$$f : \mathcal{X} \longrightarrow \mathcal{Y}$$
$$x \longmapsto f(x).$$

- Most often, the decision rule is parametrized so that we rather should write $f(x) = f(x; \theta)$ for some parameter value $\theta \in \Theta$.
- The question is now

  "How do we find the right value for θ?".

☞ So many sets here... $\mathcal{X}$ is often called the covariates/features space, $\mathcal{Y}$ the response space, the product space $\mathcal{X} \times \mathcal{Y}$ is called the observational space and Θ the parameter space.

How do machine learn to take decision? (statistical way)

- Well since machines are deterministic, it just amounts to define what we shall call a decision/regression/discrimination rule, i.e., a (mesurable) function

$$f : \mathcal{X} \longrightarrow \mathcal{Y}$$
$$x \longmapsto f(x).$$

- Most often, the decision rule is parametrized so that we rather should write $f(x) = f(x; \theta)$ for some parameter value $\theta \in \Theta$.
- The question is now

  "How do we find the right value for θ?".

☞ So many sets here... $\mathcal{X}$ is often called the covariates/features space, $\mathcal{Y}$ the response space, the product space $\mathcal{X} \times \mathcal{Y}$ is called the observational space and Θ the parameter space.

- But before talking about finding the optimal θ, let me talk about some common objectives $\mathscr{O}$.

Some common objectives

Classification We are asked to specify which of K categories some input belongs to, i.e., the learning algorithm outputs a function

$$f : \mathcal{X} \longrightarrow \{1, \ldots, K\}.$$

Regression We are asked to predict a numerical value given some inputs, i.e., the learning algorithm outputs a function

$$f : \mathcal{X} \longrightarrow \mathcal{Y}.$$

Imputation We are asked to "complete" some missing values in the input $X \in \mathcal{X}$, e.g., $X = (X_1, X_2, ?, ?, X_5, \ldots, ?, ?, X_p)$.

Risk (Generalization error)

Definition 1. A function ℓ defined on $\mathcal{Y} \times \mathcal{Y}$, $\mathcal{Y} \subseteq \mathbb{R}^d$, $d \geq 1$, is said to be a loss function if for all $y, y' \in \mathcal{Y}$ we have

$$\ell(y, y) = 0, \qquad \ell(y, y') > 0, \quad y \neq y'.$$

Risk (Generalization error)

Definition 1. A function ℓ defined on $\mathcal{Y} \times \mathcal{Y}$, $\mathcal{Y} \subseteq \mathbb{R}^d$, $d \geq 1$, is said to be a loss function if for all $y, y' \in \mathcal{Y}$ we have

$$\ell(y, y) = 0, \qquad \ell(y, y') > 0, \quad y \neq y'.$$

- Given an application driven loss function ℓ, our idealized objective is to minimize w.r.t. θ the theoretical risk or generalization error

$$R_{\text{theo}}(\theta) = \mathbb{E}\{\ell(Y, f(X; \theta))\},$$

  where expectation is w.r.t. the unknown joint distribution of (X, Y).
- Thus our answer to our original question would be

$$\theta_* = \arg\min_{\theta \in \Theta} R_{\text{theo}}(\theta).$$

Risk (Generalization error)

Definition 1. A function ℓ defined on $\mathcal{Y} \times \mathcal{Y}$, $\mathcal{Y} \subseteq \mathbb{R}^d$, $d \geq 1$, is said to be a loss function if for all $y, y' \in \mathcal{Y}$ we have

$$\ell(y, y) = 0, \qquad \ell(y, y') > 0, \quad y \neq y'.$$

- Given an application driven loss function ℓ, our idealized objective is to minimize w.r.t. θ the theoretical risk or generalization error

$$R_{\text{theo}}(\theta) = \mathbb{E}\{\ell(Y, f(X; \theta))\},$$

  where expectation is w.r.t. the unknown joint distribution of (X, Y).
- Thus our answer to our original question would be

$$\theta_* = \arg\min_{\theta \in \Theta} R_{\text{theo}}(\theta).$$

☞ Note the important point that the above risk correspond to the **expected** loss for a **future**, hence still unobserved, observation.

Empirical risk

- Unfortunately we cannot compute $R_{\text{theo}}(\theta)$ since the distribution of (X, Y) is unknown.
- However if we have data $\mathscr{D}_n = \{(X_i, Y_i) \in \mathcal{X} \times \mathcal{Y} \colon i = 1, \ldots, n\}$, supposed to have the same distribution as (X, Y), we can focus on a surrogate optimization problem

$$\hat{\theta} = \arg\min_{\theta \in \Theta} R_{\text{emp}}(\theta), \qquad R_{\text{emp}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i; \theta)).$$

- In statistics, $\hat{\theta}$ is called an estimator of θ_*.
- However note that, given the data $\mathscr{D}_n$, the expected risk related to our decision rule $x \mapsto f(x; \hat{\theta})$ is

$$\mathbb{E}_{(X,Y)} \left\{ \ell(Y; f(X; \hat{\theta})) \right\}.$$

Overfitting and underfitting

- As we said previously $R_{\text{emp}}(\hat{\theta})$ is different from $R_{\text{theo}}(\hat{\theta})$
- Actually the predictive error as measured by $R_{\text{emp}}(\hat{\theta})$ underestimate the generalization error, and thus, often lead to overfitting.
- Hence there is a pressing need to estimate the generalization error.

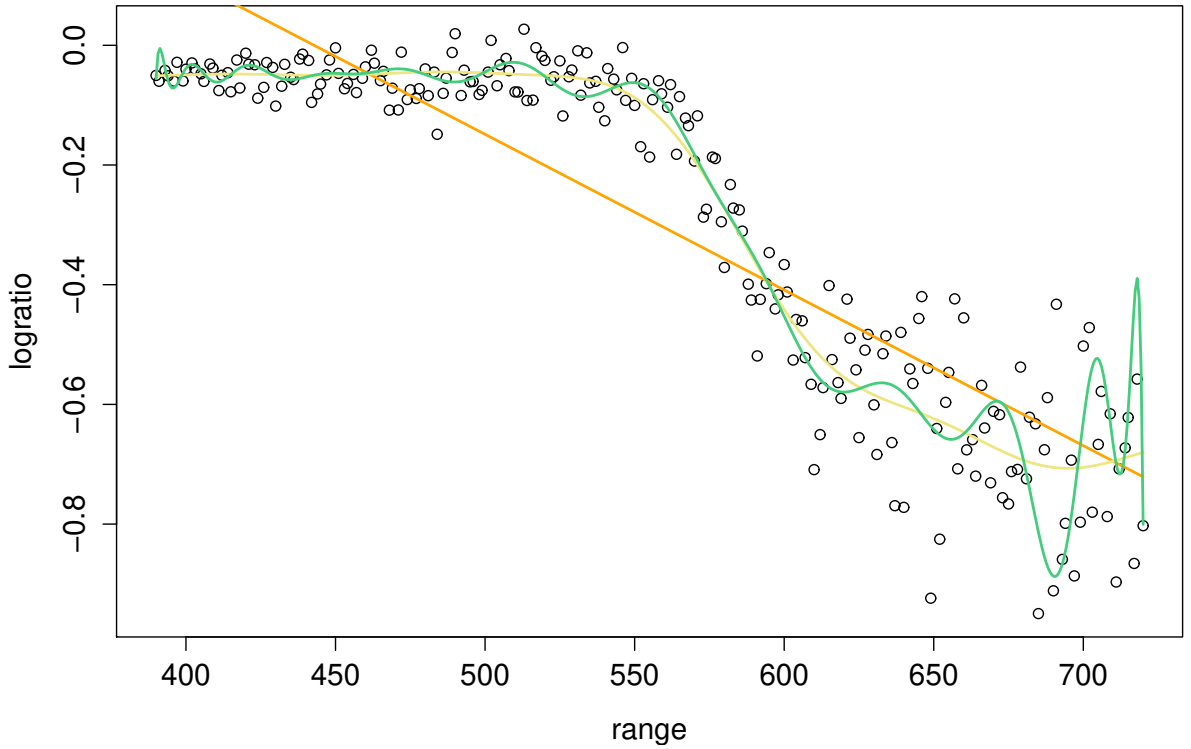


Figure 1: *Illustration of underfitting and overfitting on the* `lidar` *dataset. col2: Underfitting; Kakhi: Right fit; Green: Overfitting.*

Test/Validation risk

- Suppose we have a second, independent of $\mathscr{D}_n$, dataset, say $\tilde{\mathscr{D}}_{n_2}$[1].
- We can then compute the test/validation risk

$$R_{\text{test}}(\hat{\theta}) = \frac{1}{n_2} \sum_{i=1}^{n_2} \ell(Y_i; f(X_i; \hat{\theta})).$$

- We then have

$$\mathbb{E}\left\{ R_{\text{test}}(\hat{\theta}) \mid \mathscr{D}_n \right\} = R_{\text{theo}}(\hat{\theta}).$$

[1] Or that we have divided into two pieces the original dataset $\mathscr{D}_n$...

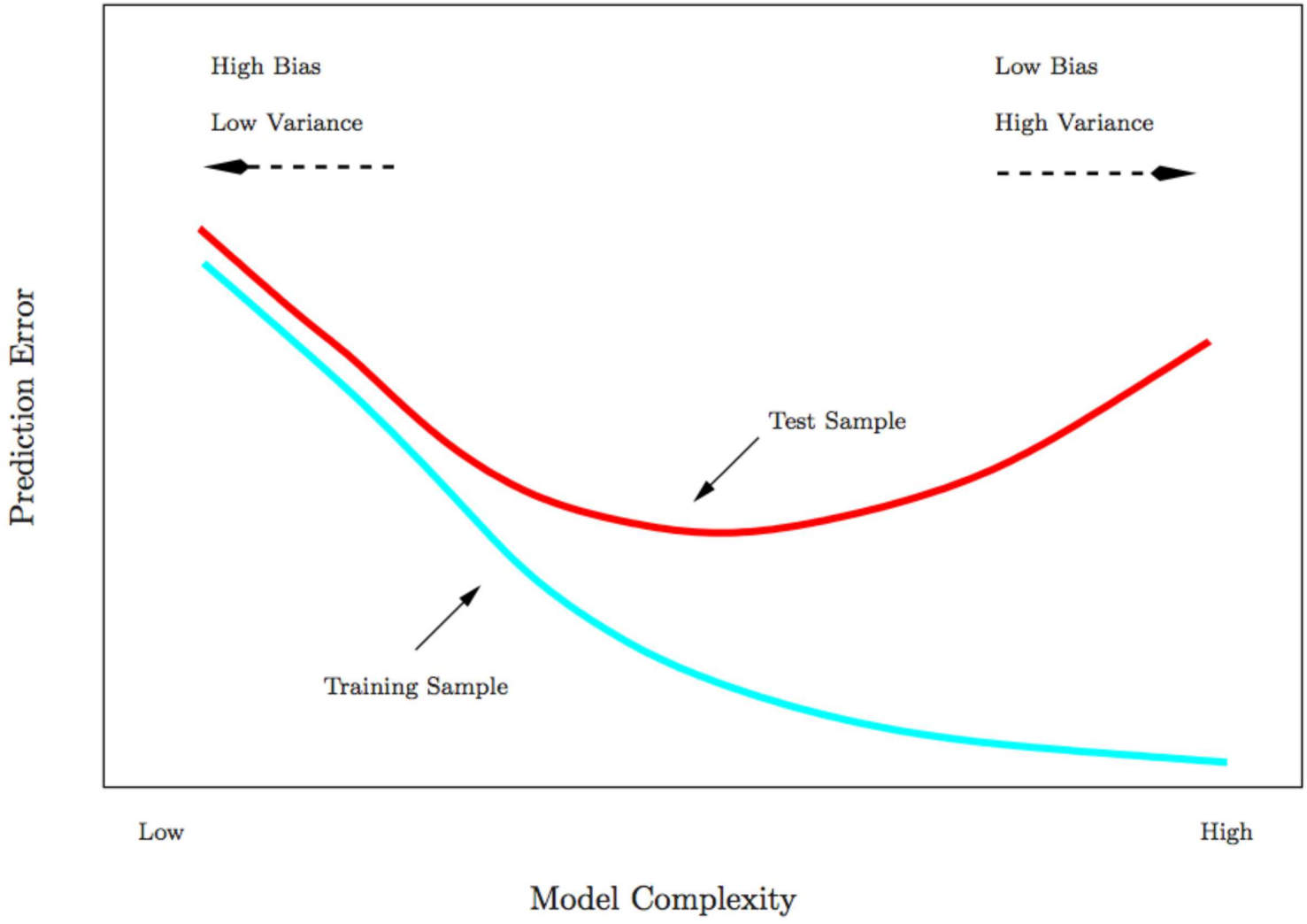


Figure 2: *Typical evolution of the training error and test error as the model complexity increases.*

Validation set

Algorithm 1: "Hold-out validation" algorithm.

input : A dataset $\mathscr{D}_n$, a training set $\mathcal{T} \subset \{1, \ldots, n\}$ and a validation set $\mathcal{V} \subset \{1, \ldots, n\}$ such that $\mathcal{T} \cap \mathcal{V} = \emptyset$ and $\mathcal{T} \cup \mathcal{V} = \{1, \ldots, n\}$.

output: An estimate of the generalization error $R_{\text{theo}}(\hat{\theta})$.

1 Compute the decision rule

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \ell(Y_i; f(X_i; \theta))$$

2 Return an estimate of the generalization error

$$\hat{R}_{\text{theo}}(\hat{\theta}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell(Y_i; f(X_i; \hat{\theta})).$$

Validation set

Algorithm 1: "Hold-out validation" algorithm.

input : A dataset $\mathscr{D}_n$, a training set $\mathcal{T} \subset \{1, \ldots, n\}$ and a validation set $\mathcal{V} \subset \{1, \ldots, n\}$ such that $\mathcal{T} \cap \mathcal{V} = \emptyset$ and $\mathcal{T} \cup \mathcal{V} = \{1, \ldots, n\}$.

output: An estimate of the generalization error $R_{\text{theo}}(\hat{\theta})$.

1 Compute the decision rule

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \ell(Y_i; f(X_i; \theta))$$

2 Return an estimate of the generalization error

$$\hat{R}_{\text{theo}}(\hat{\theta}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell(Y_i; f(X_i; \hat{\theta})).$$

- Such an approach has a main drawback: the generalization error estimate heavily relies on the validation set.
- It would be better to have several validation sets to mitigate this effect.

Cross–validation "leave p out"

Algorithm 2: "Leave p out cross validation" algorithm.

input : A dataset $\mathscr{D}_n$ and an integer $p < n$.
output: An estimate of the generalization error $R_{\text{theo}}(\hat{\theta})$.

1 Compute the $n_p = \binom{n}{p}$ subsets, say $\mathcal{V}_1, \ldots, \mathcal{V}_{n_p}$, of $\{1, \ldots, n\}$ having p elements.
2 **for** $k \leftarrow 1$ **to** s_{n_p} **do**
3 Build the training set $\mathcal{T}_k = \mathscr{D}_n \setminus \mathcal{V}_k$;
4 Compute the decision rule $\hat{\theta}_k = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}_k|} \sum_{i \in \mathcal{T}} \ell(Y_i; f(X_i; \theta))$;
5 Compute an estimate of the generalization error for $\mathcal{V}_k$

$$\hat{R}_{\text{theo},k}(\hat{\theta}_k) = \frac{1}{|\mathcal{V}_k|} \sum_{i \in \mathcal{V}_k} \ell(Y_i; f(X_i; \hat{\theta}_k)).$$

6 Return an estimate of the generalization error $\tilde{R}_{\text{theo}}(\hat{\theta}) = \frac{1}{n_p} \sum_{k=1}^{n_p} \hat{R}_{\text{theo},k}(\hat{\theta}_k)$;

Cross–validation "leave p out"

Algorithm 2: "Leave p out cross validation" algorithm.

input : A dataset $\mathscr{D}_n$ and an integer $p < n$.
output: An estimate of the generalization error $R_{\text{theo}}(\hat{\theta})$.

1 Compute the $n_p = \binom{n}{p}$ subsets, say $\mathcal{V}_1, \ldots, \mathcal{V}_{n_p}$, of $\{1, \ldots, n\}$ having p elements.
2 **for** $k \leftarrow 1$ **to** s_{n_p} **do**
3 Build the training set $\mathcal{T}_k = \mathscr{D}_n \setminus \mathcal{V}_k$;
4 Compute the decision rule $\hat{\theta}_k = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}_k|} \sum_{i \in \mathcal{T}} \ell(Y_i; f(X_i; \theta))$;
5 Compute an estimate of the generalization error for $\mathcal{V}_k$

$$\hat{R}_{\text{theo},k}(\hat{\theta}_k) = \frac{1}{|\mathcal{V}_k|} \sum_{i \in \mathcal{V}_k} \ell(Y_i; f(X_i; \hat{\theta}_k)).$$

6 Return an estimate of the generalization error $\tilde{R}_{\text{theo}}(\hat{\theta}) = \frac{1}{n_p} \sum_{k=1}^{n_p} \hat{R}_{\text{theo},k}(\hat{\theta}_k)$;

- The drawback of such an approach is that it may be very time consuming (unless $p \in \{1, 2\}$).
- The widely used case $p = 1$ is often called leave one out for obvious reasons.

Algorithm 3: "K fold cross validation" algorithm.

input : A dataset $\mathscr{D}_n$ and $K \geq 2$ divisor of n.
output: An estimate of the generalization error $R_{\text{theo}}(\hat{\theta})$.

1 Compute K disjoint subsets, say $\mathcal{V}_1, \ldots, \mathcal{V}_K$, of $\{1, \ldots, n\}$.
2 **for** $k \leftarrow 1$ **to** s_{n_p} **do**
3 Build the training set $\mathcal{T}_k = \mathscr{D}_n \setminus \mathcal{V}_k$;
4 Compute the decision rule $\hat{\theta}_k = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}_k|} \sum_{i \in \mathcal{T}} \ell(Y_i; f(X_i; \theta))$;
5 Compute an estimate of the generalization error for $\mathcal{V}_k$

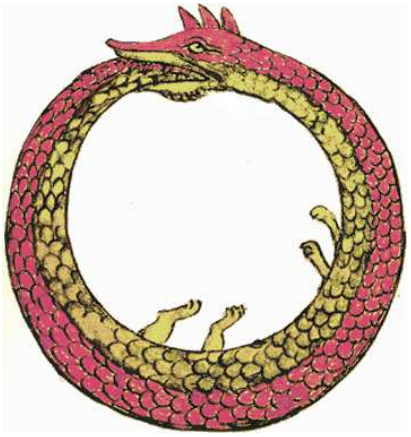
$$\hat{R}_{\text{theo},k}(\hat{\theta}_k) = \frac{K}{n} \sum_{i \in \mathcal{V}_k} \ell(Y_i; f(X_i; \hat{\theta}_k))$$

6 Return an estimate of the generalization error $\tilde{R}_{\text{theo}}(\hat{\theta}) = \frac{1}{K} \sum_{k=1}^{K} \hat{R}_{\text{theo},k}(\hat{\theta}_k)$;

- Typical choices: $K \in \{5, 10\}$;
- When $K = n$ this is the leave one out cross validation.

Train + Validation + Test

- Watch out if when building your decision rule you tuned some hyper-parameters[2], then you must use the train + validation + test.
- Essentially the validation set will be used to tune these hyperparameters while the test set will be used to estimate the generalization error.
- Indeed if you tune and estimate the generalization error on the same dataset, i.e., the test dataset, then you will underestimate the generalization error since you



did set those hyperparameters to minimize the generalization error!!!

[2] Don't worry we will define what it is later. Right now you can just assume that these are paramters that you do not estimate but rather held fix to some value of your choice.

Some common loss functions

Recall our decision rule

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i; \theta)), \qquad \text{for some loss function } \ell.$$

- quadratic loss or ℓ_2 loss: $\ell(y, y') = \|y - y'\|_2^2$
- absolute loss or ℓ_1 loos: $\ell(y, y') = \|y - y'\|_1$
- $0 - 1$ loss: $\ell(y, y') = 1_{\{y \neq y'\}}$
- Cross-entropy: $\ell(y, y') = -y^\top \ln y'$ defined on $\mathbb{S}_{K-1} = \{u \in [0,1]^K : \|u\|_1 = 1\}$.
- Negative log-likelihood (to be defined later if you don't know it)[3]

[3] Technically speaking, it is not a loss function but it plays a major role in statistics so I had to mention it here!

What is a statistical learner?

- It is just a statistician actually...

What is a statistical learner?

- It is just a statistician actually. . . who has gained some computational and optimization skills
- You will be considered as a "good" if you have a comprehensive knowledge of the available choices for the $f(x;\theta)$ function[4]
- You will be considered as an "expert" if
  - given an application, you have a prior knowledge of which function f would perform best;
  - when things goes wrong, you know why and how to fix it.

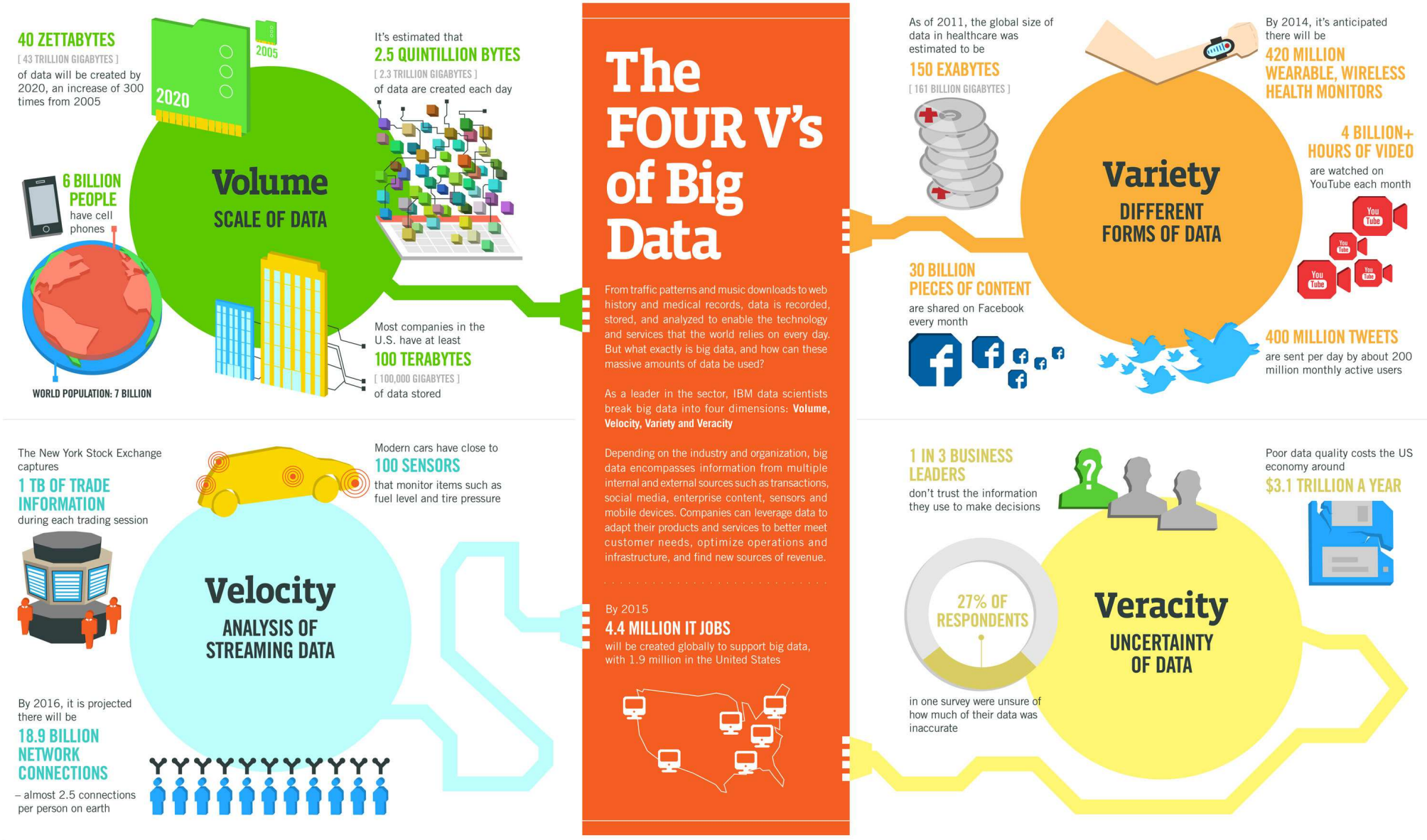
What is a statistical learner?

- It is just a statistician actually. . . who has gained some computational and optimization skills
- You will be considered as a "good" if you have a comprehensive knowledge of the available choices for the $f(x;\theta)$ function[4]
- You will be considered as an "expert" if
  - given an application, you have a prior knowledge of which function f would perform best;
  - when things goes wrong, you know why and how to fix it.

☞ In this course I will introduce just a few of these f (but keep in mind that we are missing soooooo many of them!)

[4] by this I mean not finding θ but the "parametric shape" of f itself!

Big data 4V (5V = 4V + Value?) (Source IBM Infography)



1. Statistical refresher

Types of variables

- There are two main type of variables:

  Quantitative such as height, weights, ...
  Qualitative such colors, lefty/righty, ...

- Often qualitative variables are encoded as integers.
- Possible side effect is that computer may wrongly perform standard algebra on those values!
- Pressing need to encode them as factors
- Note that, if needed, one can convert a quantitative variable to a factor using discretization, e.g., $[0, 5], [5, 10], \ldots$

Definition 2. Having a sample $\mathcal{D}_n = \{X_i : i = 1, \ldots\}$ we can compute the associated ordered statistics, denoted $X_{i:\,n}$, $i = 1, \ldots, n$, by

$$X_{1:\,n} \leq X_{2:\,n} \leq \cdots \leq X_{(n-1):\,n} \leq X_{n:\,n}.$$

Even if the X_i's are independent, the order statistics $X_{i:\,n}$ are not! It is a consequence of the ordering, e.g., $X_{2:\,n}$ has to be larger than $X_{1:\,n}$.

Summary statistics

- Having observed a sample $x_1, \ldots, x_n$, it is common practice to give a brief summary of the data using summary statistics.
- Measures of location refer to the central position of the data, i.e., where a future observation would typically lie.
- Measure of dispersion refer to the spread of the data, i.e., does observation can vary a lot or not?

Location sample mean, sample median, midhinge
Dispersion sample standard deviation, range, inter quartile range, MAD
Shape Skewness, kurtosis

Measures of location

Mean

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

Median

$$\text{Median} = \begin{cases} x_{\frac{n+1}{2}:n}, & n \text{ is odd} \\ 0.5\left(x_{\frac{n}{2}:n} + x_{(\frac{n}{2}+1):n}\right), & n \text{ is even.} \end{cases}$$

Quantile of order p with $0 < p < 1$

$$Q_p = (1-\gamma)x_{j:n} + \gamma x_{j+1:n}, \qquad j = [np + 1 - p], \quad \gamma = np + 1 - p - j$$

Quartiles are special cases with $p = 1/4, 3/4$ and often denoted Q_1 and Q_3.

Measures of dispersion

Standard deviation

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Range

$$\text{Range} = \max x_i - \min x_i$$

Interquartile Range

$$\text{IQR} = Q_3 - Q_1$$

Statistical graphics

- A picture worths a thousand words

Statistical graphics

- A picture worths a thousand words but takes place so need to worth it
- Widely used statistical plots are
  - histograms, barplots
  - boxplots
  - scatterplots
  - quantile–quantile plots

Histograms

- Histograms are used to visualized the distribution of the data.
- They are empirical versions of the probability density function of a quantitative variable
- Each class/modality is depicted by a rectangle whose area is proportional to the corresponding class frequency.
- Statisticians usually used normalized versions so that the total area of the histogram is 1[5].
- More precisely we have

$$h_j = \frac{n_j}{n\ell_j}, \qquad j = 1, \ldots, J, \quad n_j = \# \text{ obs. in class j.}$$

[5] as the probability density function integrates to 1

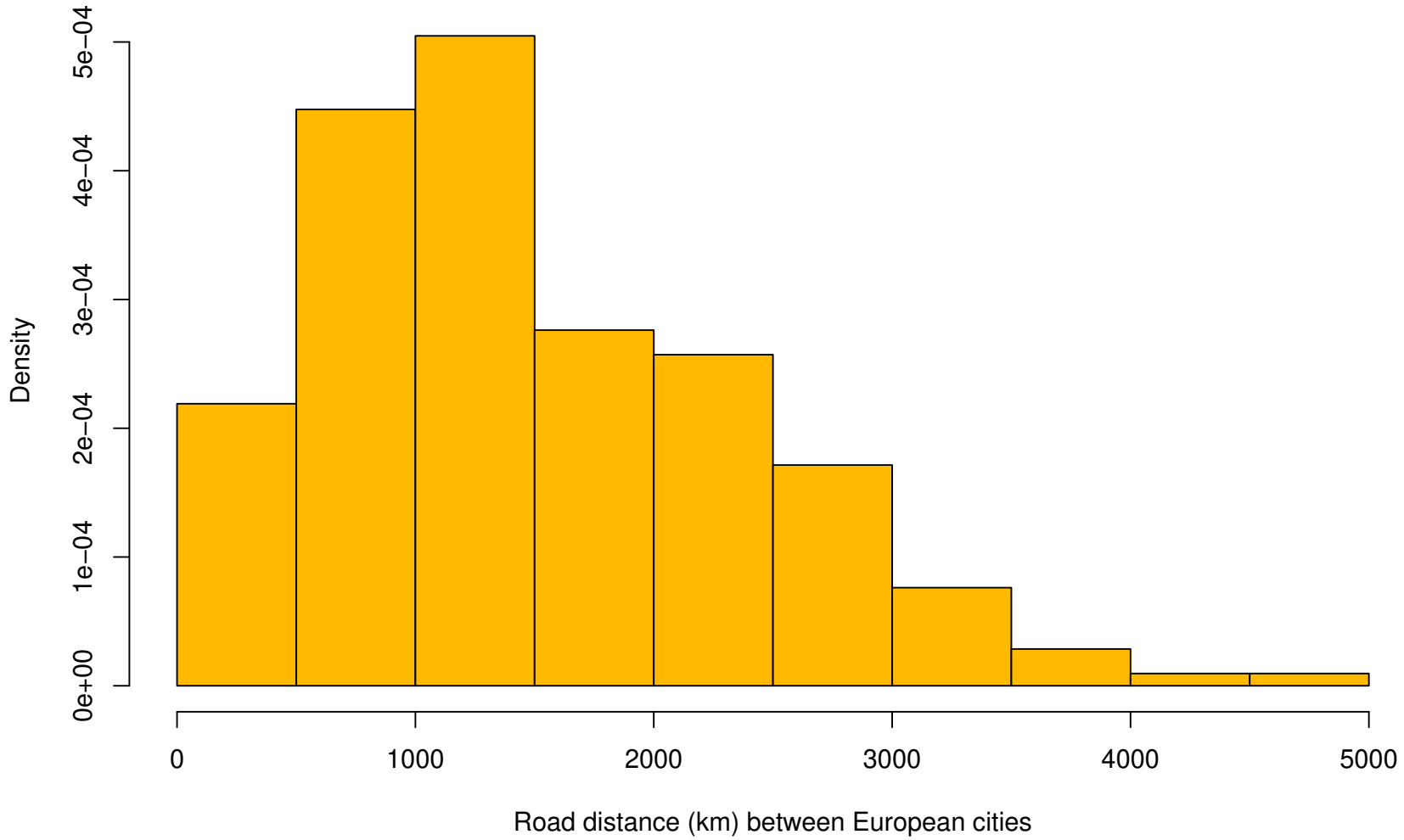


Figure 3: *Histogram of distance in km between 21 European cities.*

Barplots

- Barplots are somehow similar to histograms but for categorical variable or variable with finite numbers of possible outcomes.

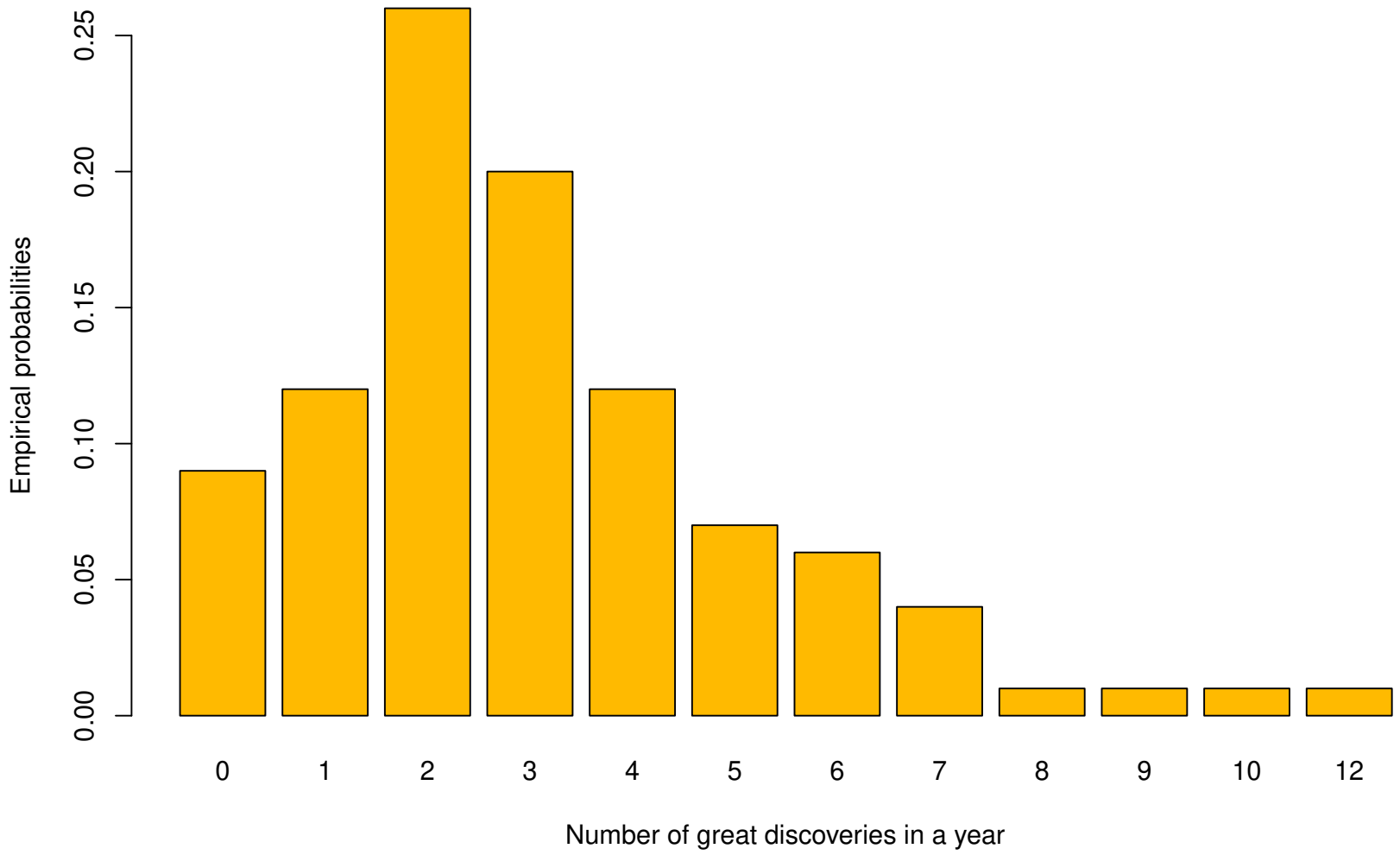


Figure 4: *Barplot of the number of yearly "great" discoveries from 1860 to 1959.*

Kernel density estimates

- If you have more than one variable, comparing histograms is a mess, i.e., looking side by side plots and barely readable overlapping histograms.
- It is much easier to compare probability functions estimates, i.e., kernel density estimation

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\mathbf{x} - \mathbf{X_i} h\right),$$

where $h > 0$ is the bandwidth and K is a kernel.

The larger the bandwidth the smoother will be the estimate $\hat{f}_h$.

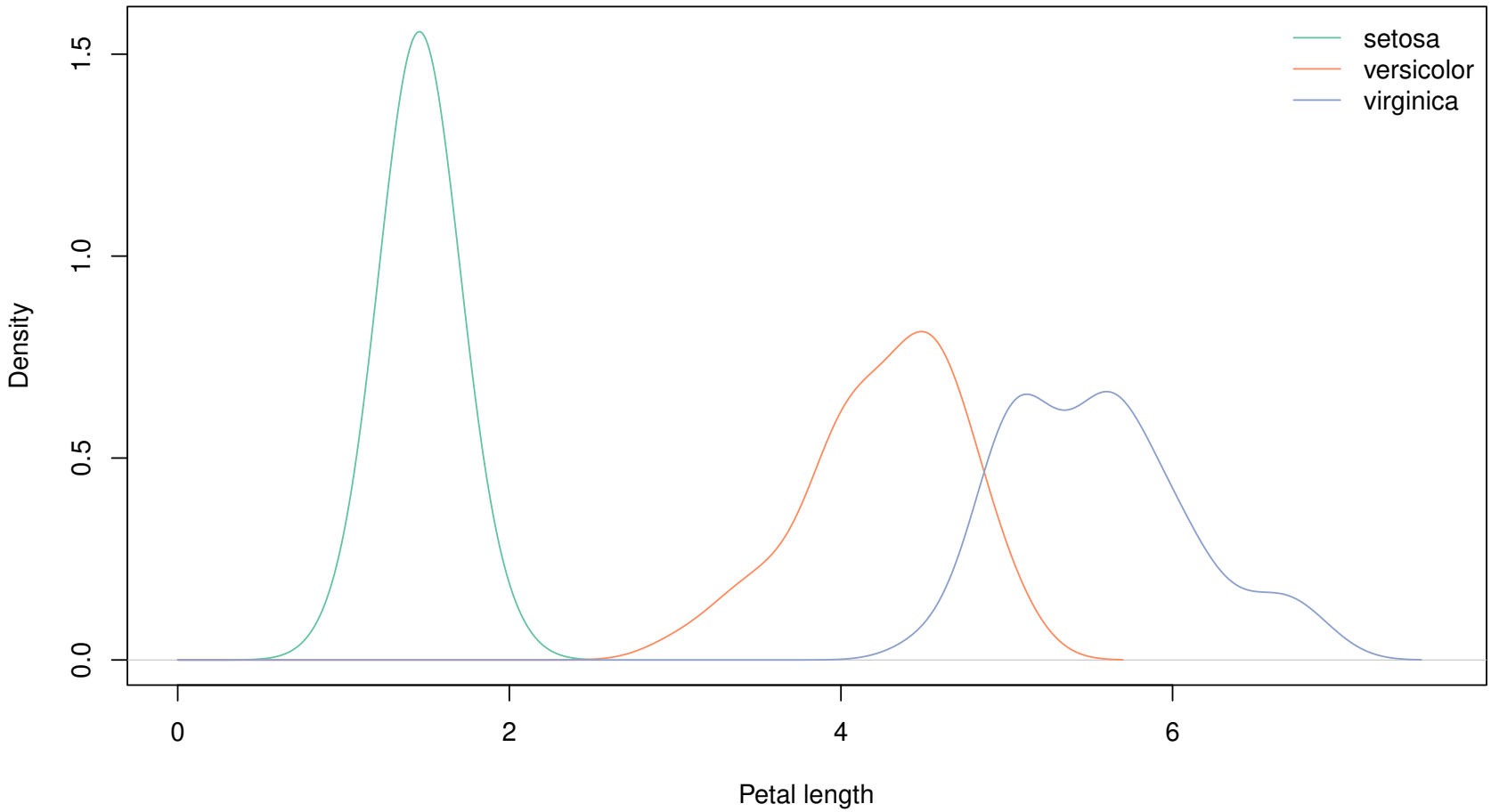


Figure 5: *Kernel denstiy estimates of the weights of chicks (g) with respect to their feed type supplements.*

Boxplots

- Boxplots also helps visualizing the distribution of the data but take less space.
- They are never used alone but rather in groups to spot any differences.
- It consists of a box (Q_1, Q_3 and the median) and whiskers defined as the closest observation[6] to $Q_{1,3} \mp 1.5IQR$.
- Observation outside those whiskers are usually denoted as outliers.

! Outliers are not spurious observations and should not be discarded. To do so, you need a justification such as measurement problem.

[6] towards the center of the distribution

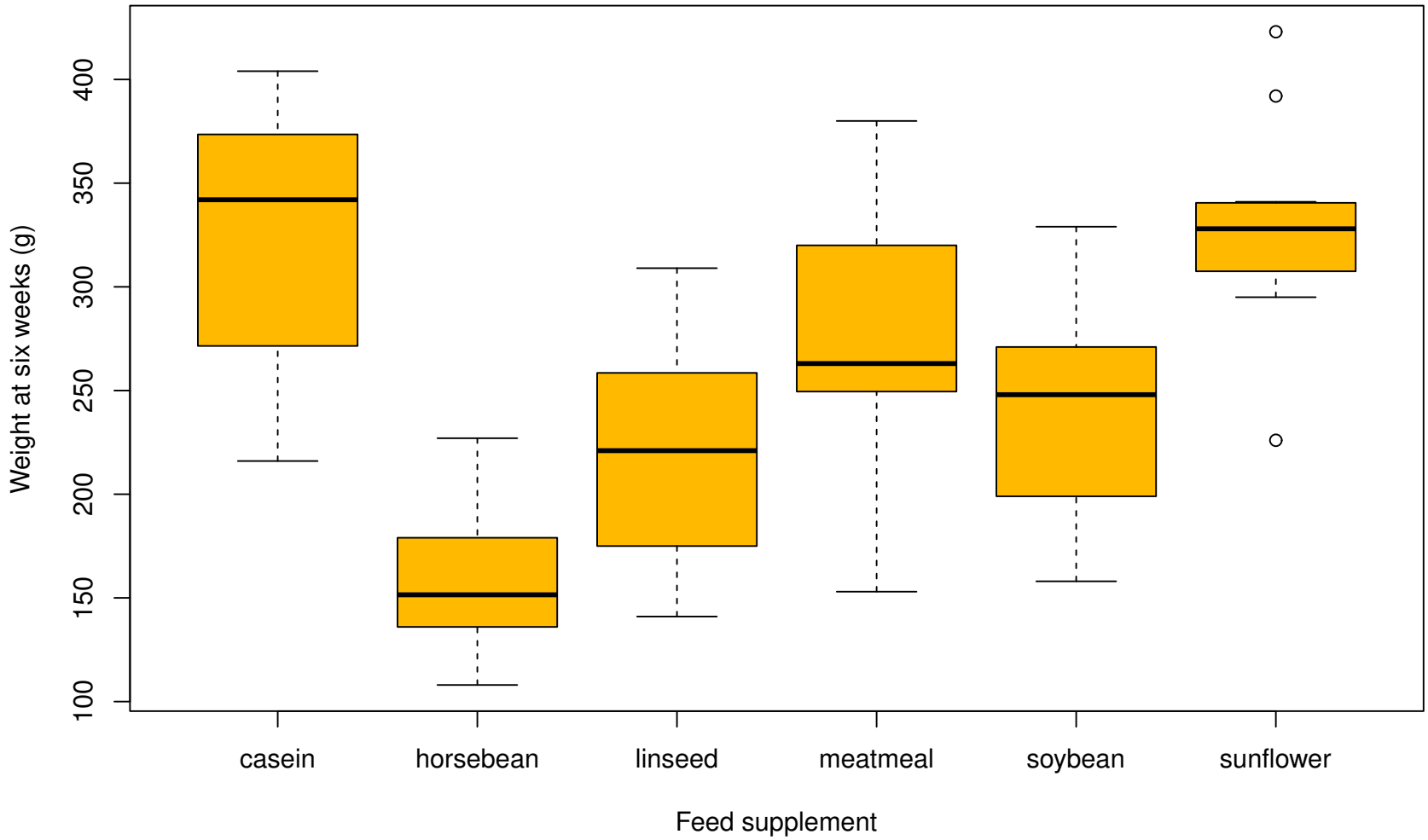


Figure 6: *Boxplots of the weights of chicks (g) with respect to their feed type supplements.*

Scatter plot

- Scatter plot aims at visualizing relationship between two variables
- Often but not necessarily, those variables are quantitative
- We just plot the points $\{(x_i, y_i) : i = 1, \ldots, n\}$.

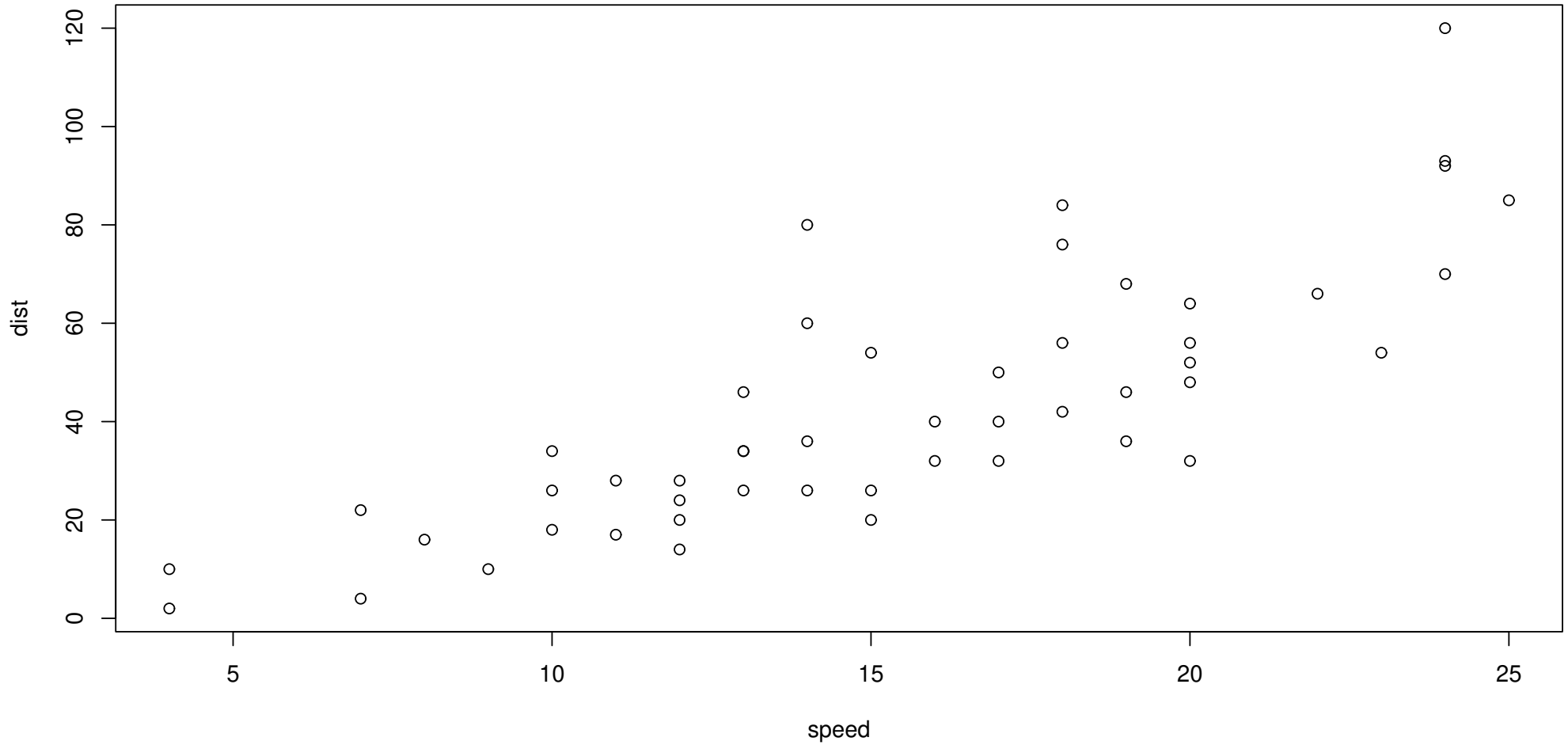
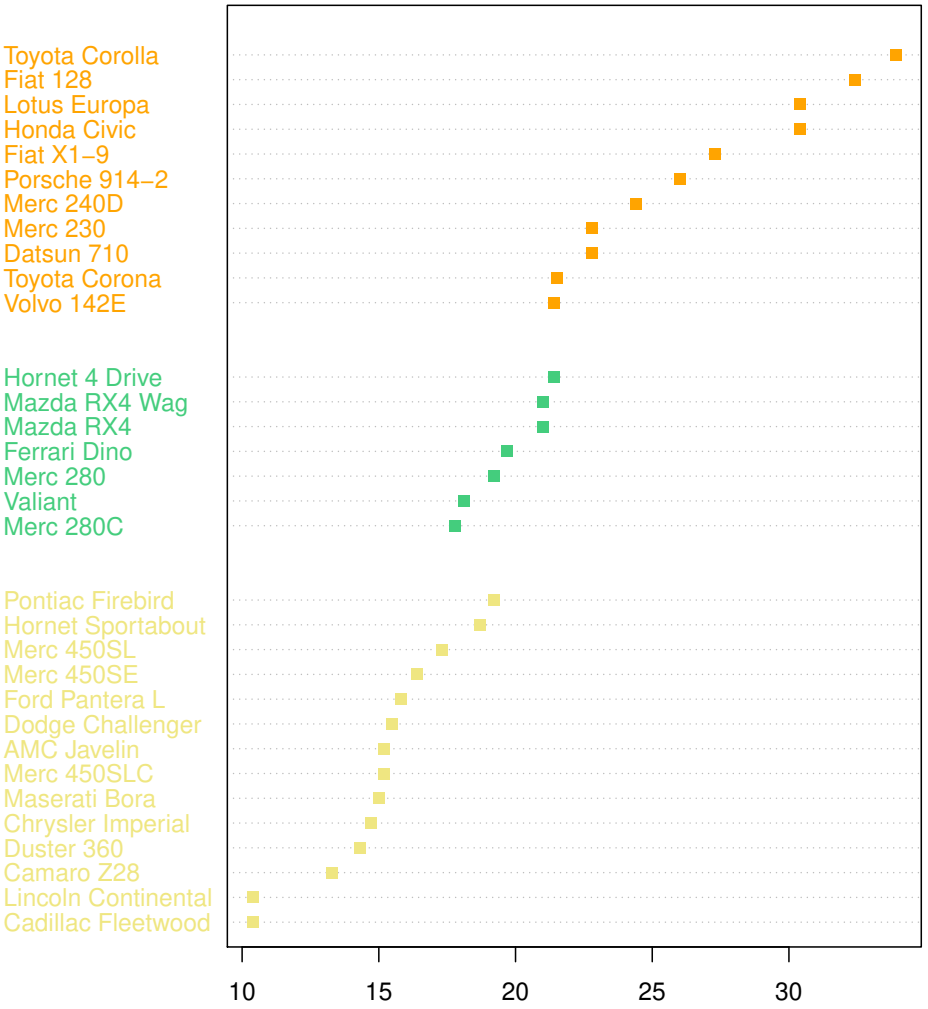


Figure 7: *Scatterplot of the distance taken to stop as the speed varies. A linear dependence seems to occur–theoretically, one would expect a quadratic one, wouldn't we?*

Dotchart



	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4
Hornet 4 Drive	21.4	6
Hornet Sportabout	18.7	8
Valiant	18.1	6
...		
Lotus Europa	30.4	4
Ford Pantera L	15.8	8
Ferrari Dino	19.7	6
Maserati Bora	15.0	8
Volvo 142E	21.4	4

[32 rows x 11 columns]

Figure 8: *Dotchart on the consumption of cars segmented on the number of cylinders.*

QQ-plot

- Quantile quantile plots are used to check whether:
  - two samples share the same distribution
  - a sample follows a given, e.g., fitted, distribution.
- The plot is based on ordered statistics

$$x_{1:n} \leq x_{2:n} \leq \cdots \leq x_{n:n}$$

- The first version is just a scatter plot of the ordered statistics of the two samples
- The second version is a scatter plot of the ordered statistics and the theoretical/fitted quantiles

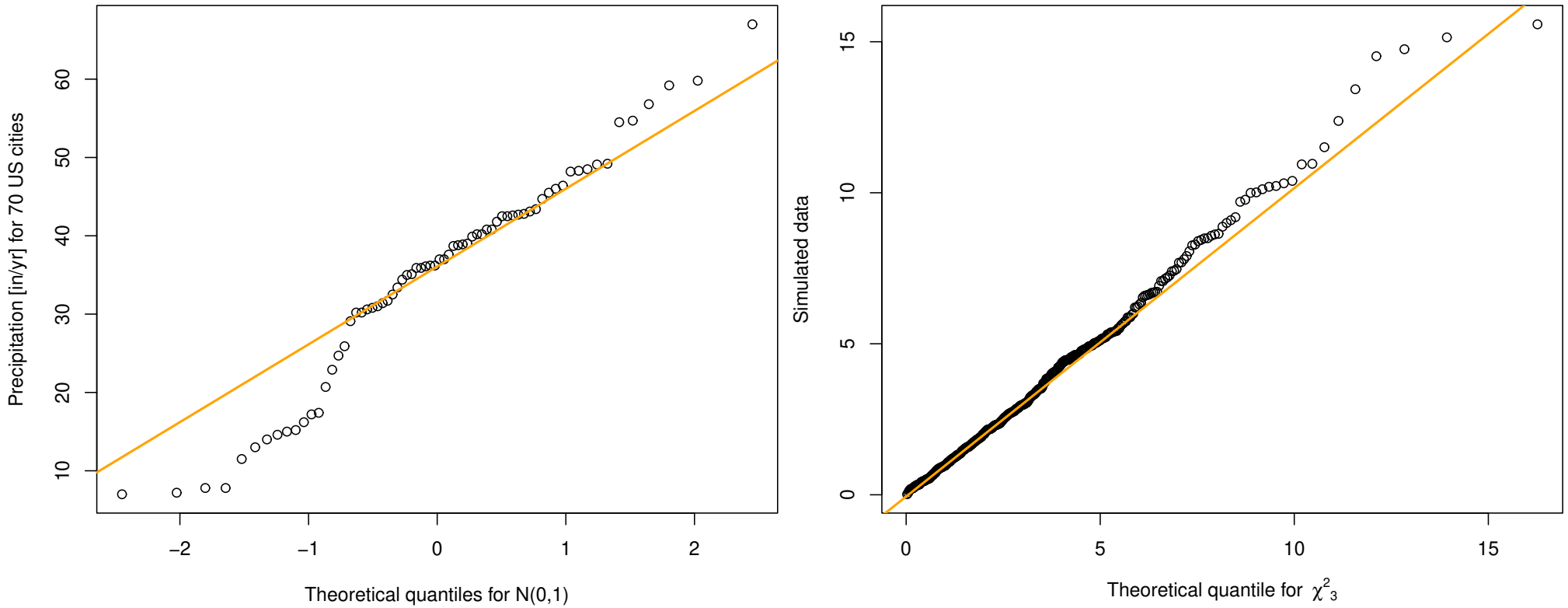


Figure 9: *Illustration on the use of qq-plots. Left: Precipitations doesn't appear to be Gaussian. In particular, the Gaussian distribution appears to overestimate the smallest precipitation amount. Right: The* χ^2_3 *distribution is reasonnable choice. (here confidence intervals are missing which is (very) unfortunate.)*

Probability density function

Definition 3. A probability density function, or density, is a non–negative function f defined on a (non finite) set E and such that

$$\int_E f(x)\mathrm{d}x = 1.$$

Definition 4. A probability mass function is just as a p.d.f. but for at most enumerable set E, i.e., a non negative function m and such that

$$\sum_{x \in E} m(x) = 1.$$

In general, a random variable can be a mixture of both discrete and continuous cases, e.g., rainfall.

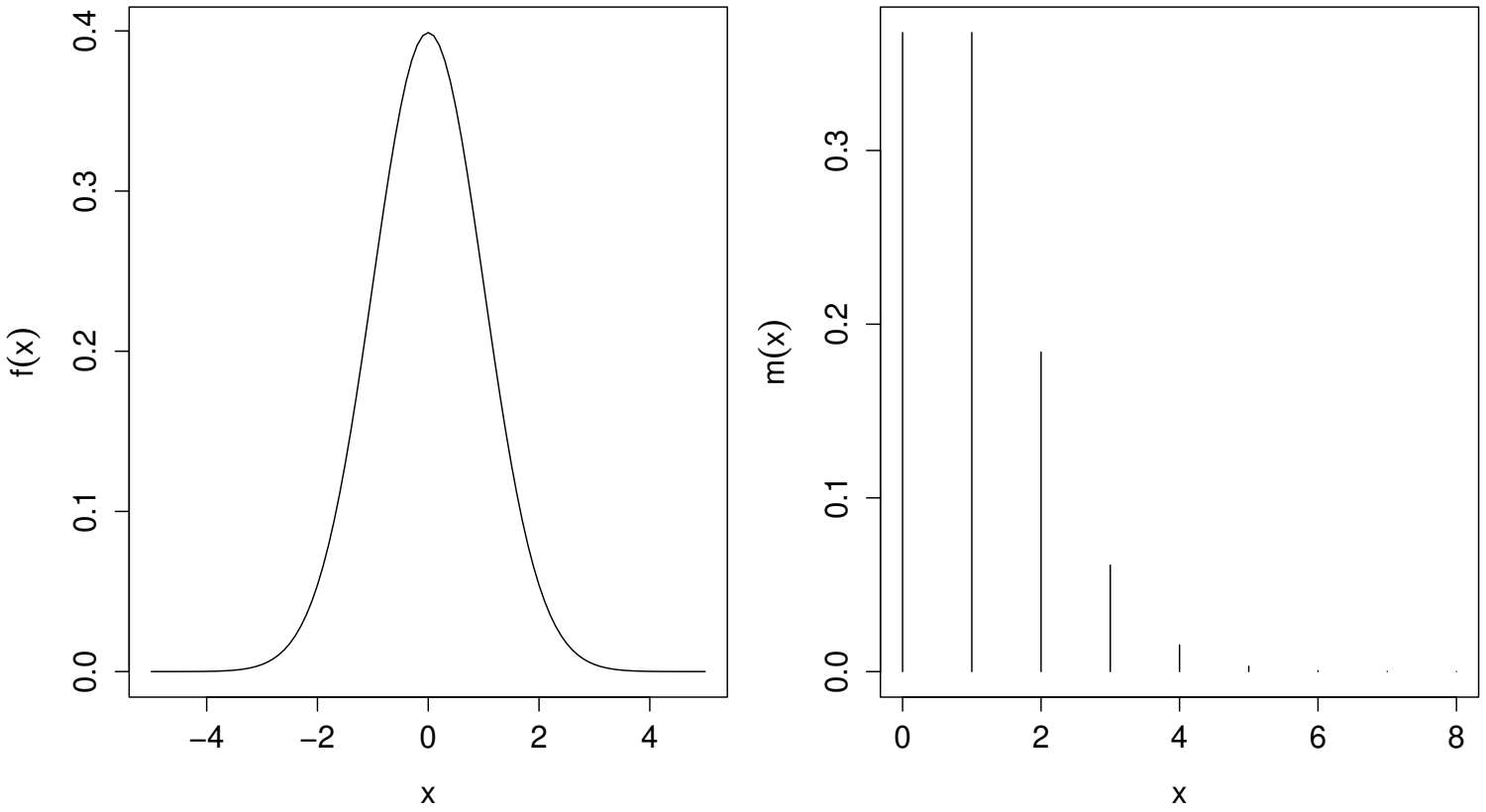


Figure 10: *Examples of probability density/mass functions. Left: Gaussian distribution. Right: Poisson distribution.*

Cumulative distribution function

Definition 5. A cumulative distribution function, or distribution, is a càd–làg function F given by

$$F(x) = \Pr(X \leq x), \qquad x \in E.$$

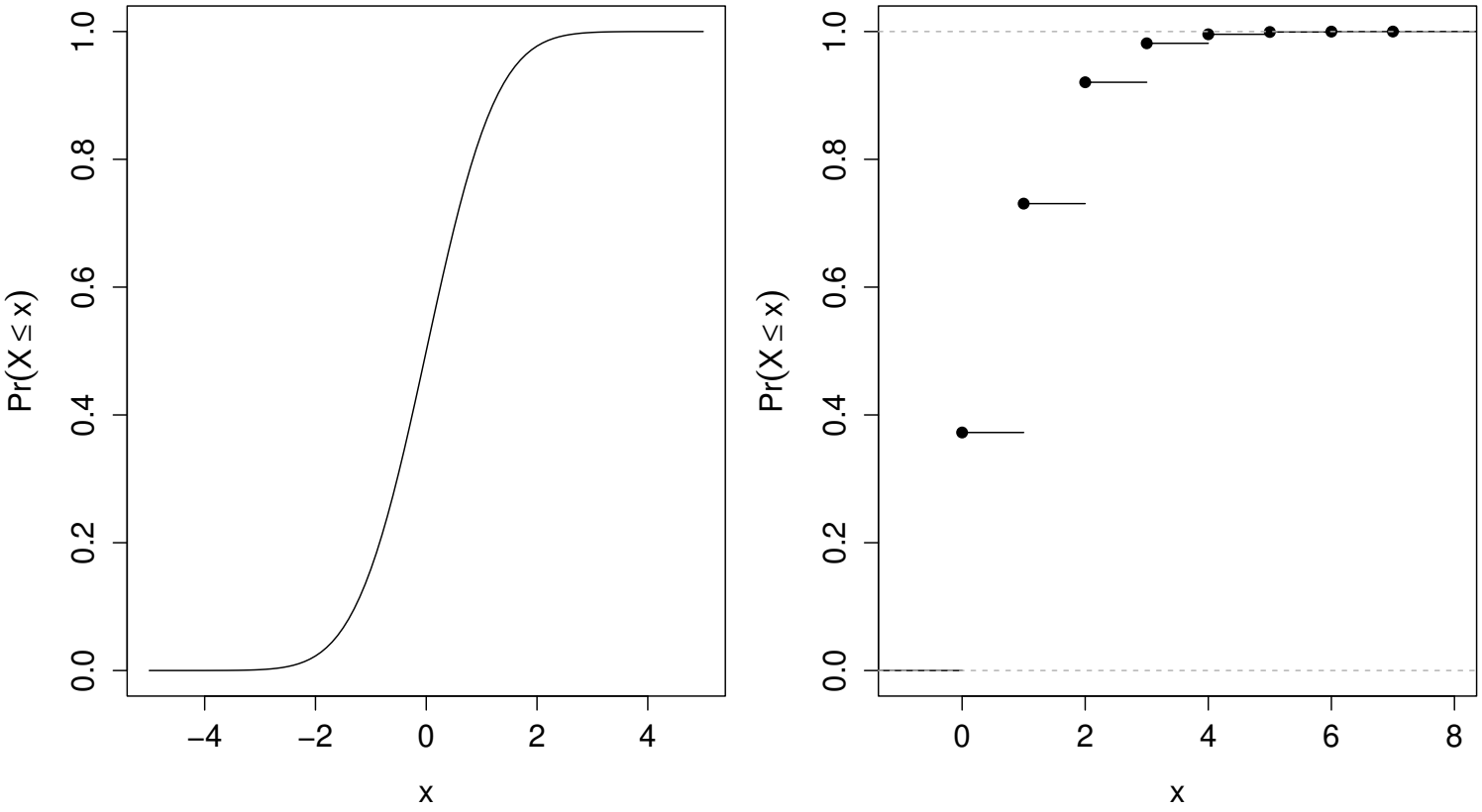


Figure 11: *Examples of cumulative distribution functions. Left: Gaussian distribution. Right: Poisson distribution.*

Definition 6. A parametric family of functions $\{f(x;\theta) : x \in E, \theta \in \Theta\}$ is a statistical model if, for any $\theta \in \Theta$, $x \mapsto f(x;\theta)$ is a probability density/mass function on E.
The sets Θ and E are respectively called parameter space and observational space.
The above model is said to be parametric if $\dim(\Theta) < \infty$.

Example 1. The Gaussian model, denoted $X \sim N(\mu, \sigma^2)$, is given by

$$f(x;\theta) = (2\pi\sigma^2)^{-1/2} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \qquad \theta = (\mu, \sigma^2), \quad E = \mathbb{R}, \quad \Theta = \mathbb{R}\times(0,\infty).$$

Example 2. The Poisson model, denoted $X \sim \text{Poisson}(\lambda)$, corresponds to

$$m(x;\lambda) = \frac{\lambda^x}{x!}\exp(-\lambda), \qquad E = \mathbb{N}, \quad \Theta = (0,\infty).$$

Some statistical models

Table 1: *Examples of useful statistical models*

Name	Support	Scope	p.d.f. // p.m.f.
		Continuous variable	
Gaussian	$\mathbb{R}$	General	$(2\pi\sigma^2)^{-1/2}\exp\left\{-(x-\mu)^2/2\sigma^2\right\}$
Student	$\mathbb{R}$	Heavy tailed	$(\nu\pi)^{-1/2}\Gamma(\nu/2)^{-1}\Gamma\{(\nu+1)/2)\left(1+x^2/\nu\right)^{-(\nu+1)/2}$
Log-normal	$(0,\infty)$	Positive	$(2\pi\sigma^2)^{-1/2}x^{-1}\exp\left\{-(\log x-\mu)^2/2\sigma^2\right\}$
Exponential	$(0,\infty)$	Duration	$\lambda\exp(-\lambda x)$
Weibull	$(0,\infty)$	Duration	$\kappa\lambda^{-\kappa}x^{\kappa-1}\exp\{-(x/\lambda)^\kappa\}$
Beta	$(0,1)$	Bounded	$B(\alpha,\beta)^{-1}x^{\alpha-1}(1-x)^{\beta-1}$
		Discrete variable	
Bernoulli	$\{0,1\}$	Binary	$p^x(1-p)^{1-x}$
Binomial	$\{0,\ldots,n\}$	# success	$\binom{n}{x}p^x(1-p)^{n-x}$
Geometric	$\mathbb{N}_*$	# attempt	$p(1-p)^{x-1}$
Poisson	$\mathbb{N}$	Counts	$\lambda^x\exp(-\lambda)/x!$
Categorical	$\{1,\ldots,k\}$	Factor	$p_j, j=1,\ldots,k$

Example: FC Nantes scoring abilities

We are interesting in modelling the number of goals scored by FC Nantes—or your favourite football team.

Example: FC Nantes scoring abilities

We are interesting in modelling the number of goals scored by FC Nantes—or your favourite football team.

Since the number of goals is a count a sensible statistical model may be the Poisson distribution

$$N_i \overset{\text{iid}}{\sim} \text{Poisson}(\lambda), \qquad i = 1, \ldots, n,$$

where $\lambda > 0$ is the unknown parameter to be estimated from data.

Example: FC Nantes scoring abilities

We are interesting in modelling the number of goals scored by FC Nantes—or your favourite football team.

Since the number of goals is a count a sensible statistical model may be the Poisson distribution

$$N_i \overset{\text{iid}}{\sim} \text{Poisson}(\lambda), \qquad i = 1, \ldots, n,$$

where $\lambda > 0$ is the unknown parameter to be estimated from data.

If you want to show off a bit, you can even invoke the law of rare events

$$\text{Binomial}(n, p_n) \xrightarrow{\text{d.}} \text{Poisson}(\lambda), \qquad n \to \infty, \quad np_n \to \lambda.$$

Ligue 1 dataset

	Div	Date	Time	HomeTeam	...	MaxCAHH	MaxCAHA	AvgCAHH	AvgCAHA
0	F1	06/08/2021	20:00	Monaco	...	2.03	1.99	1.97	1.89
1	F1	07/08/2021	16:00	Lyon	...	2.00	1.94	1.96	1.89
2	F1	07/08/2021	20:00	Troyes	...	2.04	2.00	1.91	1.95
3	F1	08/08/2021	12:00	Rennes	...	1.94	2.00	1.91	1.95
4	F1	08/08/2021	14:00	Bordeaux	...	1.89	2.10	1.84	2.03
..	..	...	...	...	...	...	...	...	...
375	F1	21/05/2022	20:00	Lorient	...	1.99	1.99	1.93	1.93
376	F1	21/05/2022	20:00	Marseille	...	1.91	2.15	1.87	1.99
377	F1	21/05/2022	20:00	Nantes	...	1.86	2.25	1.81	2.07
378	F1	21/05/2022	20:00	Paris SG	...	2.05	2.25	1.85	2.01
379	F1	21/05/2022	20:00	Reims	...	2.01	1.96	1.95	1.92

[380 rows x 105 columns]

The maximum likelihood estimator (sloppy)

- Having observed independent copies $\mathbf{Y} = (Y_1, \ldots, Y_n)$ we may want to fit our statistical model using the maximum likelihood estimator

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta; \mathbf{Y}), \qquad \ell(\theta; \mathbf{Y}) = \sum_{i=1}^{n} \log f(Y_i; \theta)$$

- Widely used in practice since (under regularity conditions) it is
  - consistent and asymptotically efficient
  - widely applicable and versatile
  - rather straightforward to implement
- With loose notations, and provided the sample size n is large enough,

$$\hat{\theta} \,\dot{\sim}\, N\left(\theta_*, \Sigma_n\right), \qquad \Sigma_n = -\left\{\nabla^2 \ell(\hat{\theta}; \mathbf{Y})\right\}^{-1}.$$

Theorem 1. *Let $\mathbf{Y}_n = (Y_1, \ldots, Y_n)$, $n \geq 1$, an n–sample of independent copies with p.d.f. $f(\cdot; \theta_*)$. Then, under regularity assumptions, the maximum likelihood estimator defined by*

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log f(Y_i; \theta)$$

satisfies

$$\sqrt{n}\left(\hat{\theta} - \theta_*\right) \xrightarrow{d.} N\left\{0, -H(\theta_*)^{-1}\right\}, \qquad n \to \infty,$$

where $H(\theta_*) = \mathbb{E}\{\nabla^2 \log f(X; \theta_*)\}$.

Proof. Taylor expansion + CLT + Slutsky □

Application: FC Nantes scoring

Exercise 1. Based on a sample $X_1, \ldots, X_n$, compute the MLE for a Poisson model. What is the (approximate) distribution for this estimator? Apply your results to the Ligue 1 data set.

Model checking

- Fitting a model is not enough, we have to check if our fitted model is actually good. It is model checking.
- One can use numerical quantities such as overall error, but if possible, graphical model checking has to be preferred
- Briefly the idea is to compare observations to predictions from the fitted model.
- Two cases arise:

  Discrete Compare the empirical p.m.f. to the fitted one;
  Continuous Produce a quantile-quantile plot

Application: FC Nantes

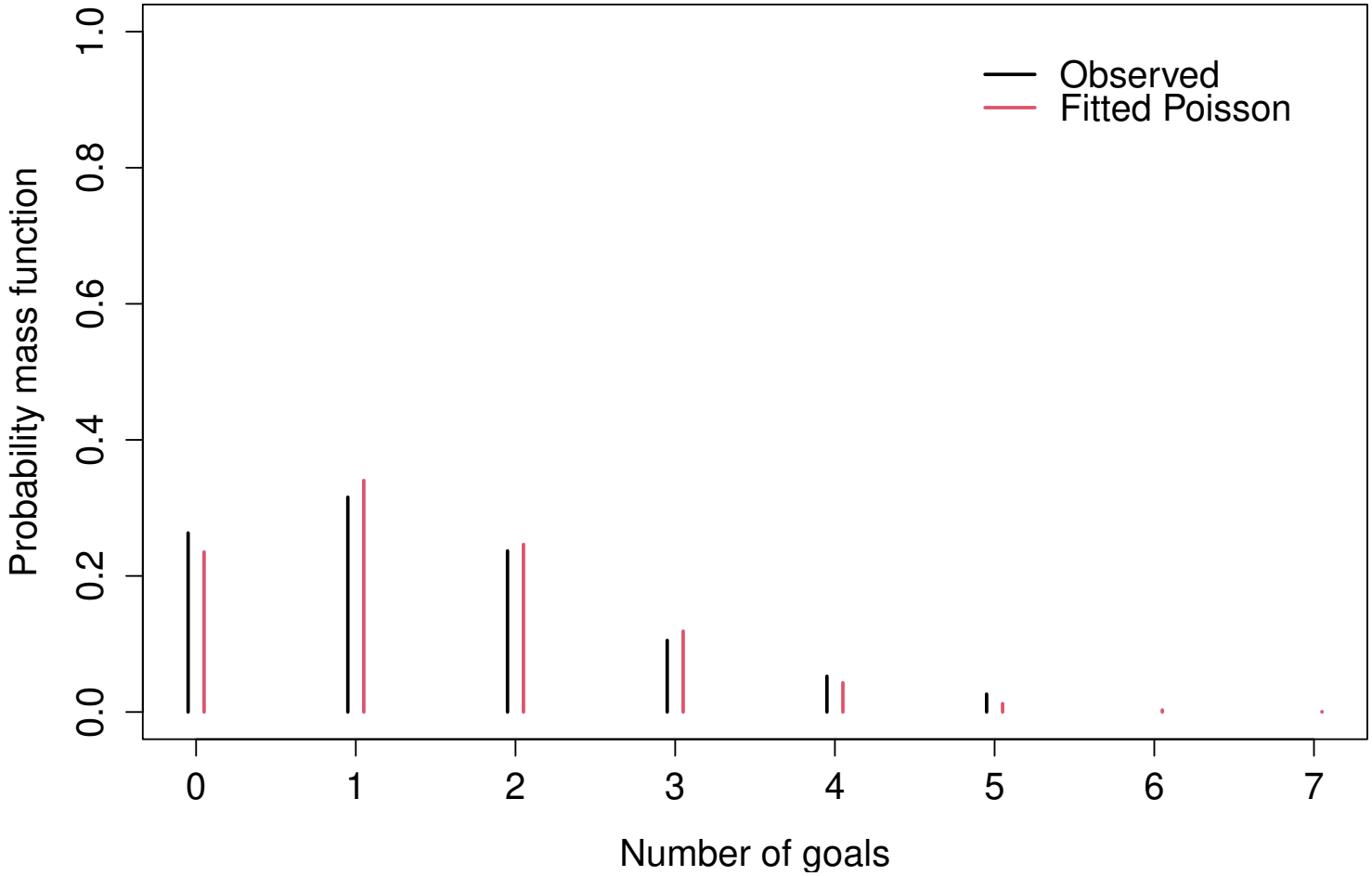


Figure 12: *Comparison of the empirical probability mass function and that from our fitted Poisson model.*

Standard errors

- Having an estimate of the parameter θ is not enough.
- It is (very!) good practice to show its respective standard errors

$$\text{std err}(\theta) = \sqrt{\text{Var}(\hat{\theta})}, \qquad \text{for some univariate parameter } \theta.$$

- Standard errors measure how precise is your estimate, i.e., the lower, the better.
- Going back to our MLE properties, i.e., $\hat{\theta} \dot{\sim} N\left(\theta_*, \Sigma_n\right)$ where $\Sigma_n = -\left\{\nabla^2 \ell(\hat{\theta}; \mathbf{Y})\right\}^{-1}$, we conclude that standard errors are thus the square root of the diagonal elements of Σ_n.

> **i** Most numerical optimizers can output Σ_n (or Σ_n^{-1}) so standard errors can easily be computed (and you have no excuse!).

Definition 7. A confidence interval of level $\alpha \in (0,1)$ for some unknown quantity $\theta_* \in \Theta$ is an interval $I_\alpha \subset \Theta$ such that $\Pr(\theta_* \in I_\alpha) = \alpha$.
Note that I_α is computed only from the sample $X_1, \ldots, X_n$ and is therefore a random interval.

- Confidence intervals can be:

  approximate in which case $\Pr(\theta_* \in I_\alpha) \geq \alpha$;
  asymptotic in which case $\Pr(\theta_* \in I_\alpha) \to \alpha$ as $n \to \infty$.

Using the asymptotic properties of the MLE, i.e., $\hat{\theta} \dot{\sim} N(\theta_*, \Sigma_n)$, a (symmetric) asymptotic confidence interval for θ_* is

$$\left[\hat{\theta} - \text{std. err.}(\hat{\theta}) z_{1-(1-\alpha)/2}; \hat{\theta} + \text{std. err.}(\hat{\theta}) z_{1-(1-\alpha)/2}\right],$$

where z_p is the quantile of a $N(0,1)$ of order p.

Beware

- Confidence intervals are often misinterpreted
- A wrong interpretation will be to say that

  "The true parameter θ_* belongs to **this** confidence interval with probability α." 👎

- The right interpretation is rather

  "If we were to replicate our experiment N times independently, i.e., Bernoulli experiments, we will thus have N independent confidence intervals and

$$\frac{1}{N}\sum_{j=1}^{N} 1_{\{\theta_* \text{ belongs to the } j\text{-th confidence interval}\}} \xrightarrow{\text{a.s.}} \alpha, \qquad N \to \infty.$$

  " 👍

> The first interpretation is that of **credible intervals** and refer to Bayesian statistics.

Application: Hold your breath

Exercise 2. Give a 95% (symmetric) confidence interval for the parameter of the Poisson distribution.

Homework

- ☐ Get the book `An introduction to Statistical Learning with Applications in R` from this link
- ☐ Read Chapter 3 and do the lab of Section 3.6

- Linear models is probably the simple statistical model for regression problem.
- Recall that regression problem aims at predicting some numerical value Y with respect to some covariates / features $\mathbf{x} = (x_1, \ldots, x_p)\top$.
- It is the simple model as extensions are possible such as:
  - generalized linear models
  - additive models
  - generalized additive models
  - regularized linear model such as ridge, lasso or elastic net.

Definition 8. Given a sample $\mathscr{D}_n = \{(Y_i, X_i) \in \mathbb{R} \times \mathbb{R}^p : i := 1, \ldots, n\}$, a statistical model is said to be a (gaussian) linear regression model if we assume

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \cdots + \beta_p X_{i,p} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $\varepsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. More compactly, this can be written (without the Gaussian assumption but only white noise)

$$\mathbb{E}(Y \mid X) = X^\top \boldsymbol{\beta}, \qquad \boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^\top.$$

Fitting a linear model

- Having observed a data set $\mathscr{D}_n = \{(Y_i, X_i) : i = 1, \ldots, n\}$, we want to fit our linear model, i.e., compute the least square estimator $\hat{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^{n} \left(Y_i - \mathbf{X}_i^\top \boldsymbol{\beta} \right)^2$$

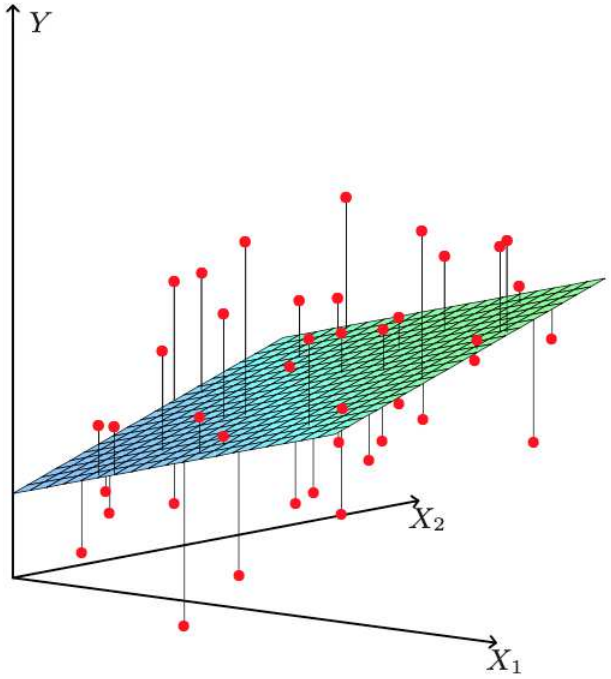


Figure 13: *Linear least square fitting with* $\mathbf{X} \in \mathbb{R}^{n \times 3}$. *[Taken from ESLII]*

Fitting a linear model

- Having observed a data set $\mathscr{D}_n = \{(Y_i, X_i) \colon i = 1, \ldots, n\}$, we want to fit our linear model, i.e., compute the least square estimator $\hat{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^{n} \left(Y_i - \mathbf{X}_i^\top \boldsymbol{\beta} \right)^2$$

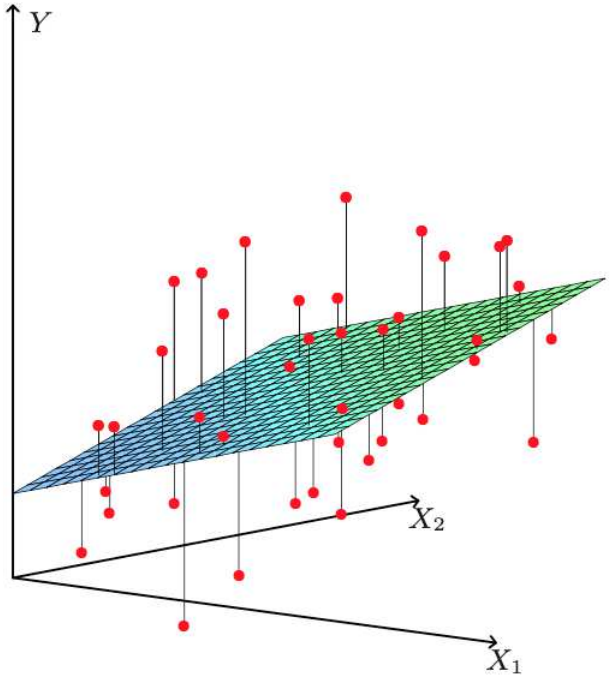


Figure 13: *Linear least square fitting with* $\mathbf{X} \in \mathbb{R}^{n \times 3}$. *[Taken from ESLII]*

- One can show that

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{Y},$$

where $\mathbf{X}$ is the design matrix whose i-th row is $\mathbf{x}_i$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)^\top$.

- This yields to the prediction

$$\hat{\mathbf{Y}} = H\mathbf{Y}, \quad H = \mathbf{X} \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top$$

Proposition 1. *For Gaussian noise, the MLE for the linear model is the least square solution. Indeed (conditionally on the features X_i) the log–likelihood is*

$$\ell(\theta; \mathcal{D}_n) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(Y_i - X_i^T\boldsymbol{\beta}\right)^2, \qquad \theta = (\boldsymbol{\beta}, \sigma^2).$$

Consequently, maximizing the above expression w.r.t. $\boldsymbol{\beta}$ consists in the least square problem

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n}\left(Y_i - X_i^\top\boldsymbol{\beta}\right)^2.$$

Least squares as the MLE

Proposition 1. *For Gaussian noise, the MLE for the linear model is the least square solution. Indeed (conditionally on the features X_i) the log–likelihood is*

$$\ell(\theta; \mathcal{D}_n) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(Y_i - X_i^T\boldsymbol{\beta}\right)^2, \qquad \theta = (\boldsymbol{\beta}, \sigma^2).$$

Consequently, maximizing the above expression w.r.t. $\boldsymbol{\beta}$ consists in the least square problem

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n}\left(Y_i - X_i^\top\boldsymbol{\beta}\right)^2.$$

☞ We can use all the properties we know about the maximum likelihood estimator!

Measure of goodness of fit

- It is common practice to measure how well the model fits the data.
- A common choice is the the coefficient of determination or percentage of variance explained R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(\hat{Y}_i - Y_i\right)^2}{\sum_{i=1}^{n} \left(\bar{Y} - Y_i\right)^2} = 1 - \frac{\text{residual sum of squares (RSS)}}{\text{total sum of squares (TSS)}}, \; \bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

- It measures how your model increases the prediction performance compared to the baseline model, e.g., unknown intercept.
- Clearly $R^2 = 1$ for perfect predictions.

Measure of goodness of fit

- It is common practice to measure how well the model fits the data.
- A common choice is the the coefficient of determination or percentage of variance explained R^2

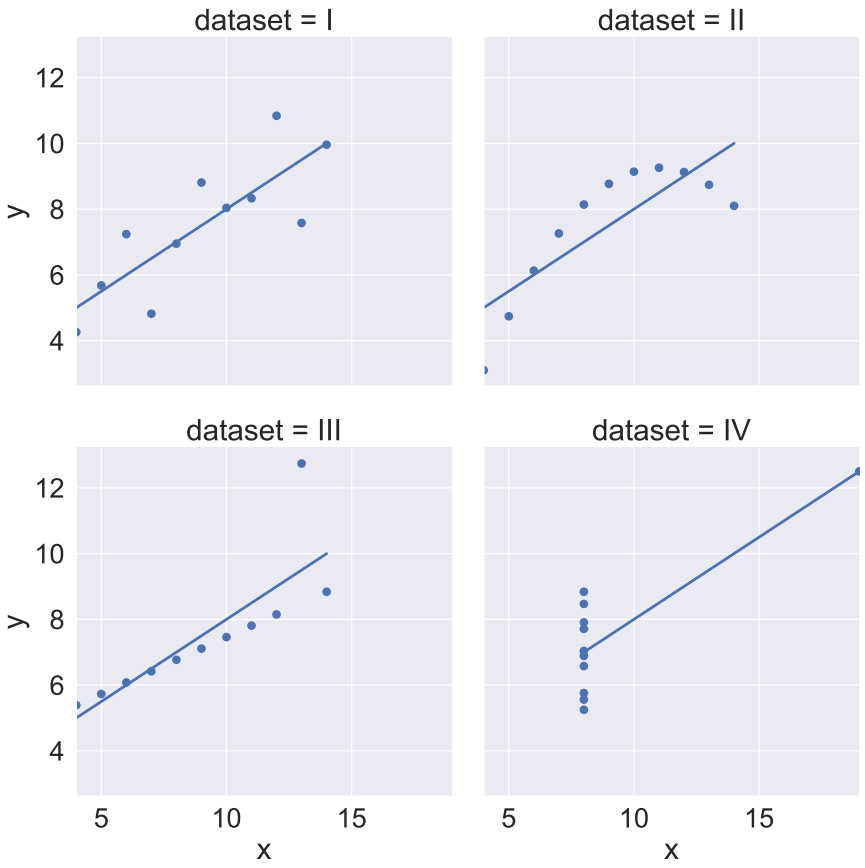
$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(\hat{Y}_i - Y_i\right)^2}{\sum_{i=1}^{n} \left(\bar{Y} - Y_i\right)^2} = 1 - \frac{\text{residual sum of squares (RSS)}}{\text{total sum of squares (TSS)}}, \; \bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

- It measures how your model increases the prediction performance compared to the baseline model, e.g., unknown intercept.
- Clearly $R^2 = 1$ for perfect predictions.

☞ Watchout if your model has no intercept the above formula is incorrect and one must use $R^2 = \text{corr}(\hat{Y}, Y)^2$.

Warning: Never trust a single numerical value!

```
>>> import seaborn as sns
>>> df = sns.load_dataset("anscombe")
>>> df
   dataset     x      y
0        I  10.0   8.04
1        I   8.0   6.95
     .
     .
     .
42      IV   8.0   7.91
43      IV   8.0   6.89
>>> df.groupby("dataset").corr().iloc[::2,-1]**2
dataset
I        x    0.666542
II       x    0.666242
III      x    0.666324
IV       x    0.666707
Name: y, dtype: float64
```



Species in Galápagos Islands (Faraway, 2014)

- Response `Species`: Number of the species found on each of the 30 islands of the Galápagos
- 5 features : `Elevation`: highest elevation of the island, `Nearest` distance from the nearest island, `Scruz` distance from the Santa Cruz island, `Adjacent` the area of the adjacent island

Interpretation

- Suppose we have fitted the following linear model

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p,$$

  we may wonder what is the meaning of $\hat{\beta}_1$ for instance?
- Sometimes (rarely), it is a physical constant but most often it has no real physical meaning as we are just building an empirical model approximating reality.

Interpretation

- Suppose we have fitted the following linear model

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p,$$

  we may wonder what is the meaning of $\hat{\beta}_1$ for instance?
- Sometimes (rarely), it is a physical constant but most often it has no real physical meaning as we are just building an empirical model approximating reality.
- Naive Interpretation:

  > A unit change in X_1 will produce on average a change of $\hat{\beta}_1$ in the response

- Such a reasoning is correct provided that:
  - the model is correct and you are not extrapolating
  - covariates are orthogonal—which is typically not the case.

Interpretation

- Suppose we have fitted the following linear model

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p,$$

  we may wonder what is the meaning of $\hat{\beta}_1$ for instance?
- Sometimes (rarely), it is a physical constant but most often it has no real physical meaning as we are just building an empirical model approximating reality.
- Right Interpretation:

  A unit change in X_1 with the other features held constant will produce on average a change of $\hat{\beta}_1$ in the response

Interpretation

- Suppose we have fitted the following linear model

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p,$$

  we may wonder what is the meaning of $\hat{\beta}_1$ for instance?
- Sometimes (rarely), it is a physical constant but most often it has no real physical meaning as we are just building an empirical model approximating reality.
- Right Interpretation:

  A unit change in X_1 with the other features held constant will produce on average a change of $\hat{\beta}_1$ in the response

☞ Beware we are talking about correlation but not causality. Think about observing a positive correlation between shoe sizes and reading abilities—we missed lurking variable age of the child! Causality analysis is difficult!

Fitting a linear model (sklearn)

```
>>> import faraway.datasets.galapagos ##just for the dataset
>>> galapagos = faraway.datasets.galapagos.load()
>>> galapagos.head()
           Species    Area  Elevation  Nearest  Scruz  Adjacent
Baltra          58   25.09        346      0.6    0.6      1.84
Bartolome       31    1.24        109      0.6   26.3    572.33
Caldwell         3    0.21        114      2.8   58.7      0.78
Champion        25    0.10         46      1.9   47.4      0.18
Coamano          2    0.05         77      1.9    1.9    903.82

>>> X = galapagos.iloc[:, 1:]
>>> Y = galapagos.Species
>>> fit = LinearRegression().fit(X, Y)
>>> fit.coef_
array([-0.02393834,  0.31946476,  0.00914396, -0.24052423, -0.07480483])
```

☞ The analysis we just made is clearly too basic and we need more theory to do it properly.

☞ We will use `statsmodels` rather since `sklearn` is very limited

Fitting a linear model (statsmodels)

```
>>> import statsmodels.formula.api as smf
>>> fit = smf.ols('Species ~ Area + Elevation + Nearest + Scruz + Adjacent', data = galapagos).fit()
>>> fit.summary()
                            OLS Regression Results
==============================================================================
Dep. Variable:                Species   R-squared:                       0.766
Model:                            OLS   Adj. R-squared:                  0.717
Method:                 Least Squares   F-statistic:                     15.70
Date:                Mon, 20 Jun 2022   Prob (F-statistic):           6.84e-07
Time:                        16:28:18   Log-Likelihood:                -162.54
No. Observations:                  30   AIC:                             337.1
Df Residuals:                      24   BIC:                             345.5
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.0682     19.154      0.369      0.715     -32.464      46.601
Area          -0.0239      0.022     -1.068      0.296      -0.070       0.022
Elevation      0.3195      0.054      5.953      0.000       0.209       0.430
Nearest        0.0091      1.054      0.009      0.993      -2.166       2.185
Scruz         -0.2405      0.215     -1.117      0.275      -0.685       0.204
Adjacent      -0.0748      0.018     -4.226      0.000      -0.111      -0.038
==============================================================================
Omnibus:                       12.683   Durbin-Watson:                   2.476
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               13.498
Skew:                           1.136   Prob(JB):                      0.00117
Kurtosis:                       5.374   Cond. No.                     1.90e+03
==============================================================================
```

Fitting a linear model (R)

```
> library(faraway) ## for the dataset
> head(gala[,-2])
            Species  Area Elevation Nearest Scruz Adjacent
Baltra           58 25.09       346     0.6   0.6     1.84
Bartolome        31  1.24       109     0.6  26.3   572.33
Caldwell          3  0.21       114     2.8  58.7     0.78
Champion         25  0.10        46     1.9  47.4     0.18
Coamano           2  0.05        77     1.9   1.9   903.82
Daphne.Major     18  0.34       119     8.0   8.0     1.84

> fit <- lm(Species ~ Area + Elevation + Nearest + Scruz  + Adjacent,
data=gala)
> fit

Call:
lm(formula = Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
    data = gala)

Coefficients:
(Intercept)         Area    Elevation      Nearest        Scruz     Adjacent
   7.068221    -0.023938     0.319465     0.009144    -0.240524    -0.074805
```

$$H_0 : \beta_j = 0 \qquad \text{vs} \qquad H_1 : \beta_j \neq 0$$

- Under the null H_0 (and with a gaussian noise), one can show that the test statistic statisfies

$$T = \frac{\hat{\beta}_j - 0}{\text{std. err.}(\hat{\beta}_j)} \sim t_{n-p-1}$$

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.0682     19.154      0.369      0.715     -32.464      46.601
Area          -0.0239      0.022     -1.068      0.296      -0.070       0.022
Elevation      0.3195      0.054      5.953      0.000       0.209       0.430
Nearest        0.0091      1.054      0.009      0.993      -2.166       2.185
Scruz         -0.2405      0.215     -1.117      0.275      -0.685       0.204
Adjacent      -0.0748      0.018     -4.226      0.000      -0.111      -0.038
==============================================================================
```

t–test in linear model (R)

```
> summary(fit)

Call:
lm(formula = Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
    data = gala)

Residuals:
     Min       1Q   Median       3Q      Max
-111.679  -34.898   -7.862   33.460  182.584

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.068221  19.154198   0.369 0.715351
Area        -0.023938   0.022422  -1.068 0.296318
Elevation    0.319465   0.053663   5.953 3.82e-06 ***
Nearest      0.009144   1.054136   0.009 0.993151
Scruz       -0.240524   0.215402  -1.117 0.275208
Adjacent    -0.074805   0.017700  -4.226 0.000297 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 60.98 on 24 degrees of freedom
Multiple R-squared:  0.7658,Adjusted R-squared:  0.7171
F-statistic:  15.7 on 5 and 24 DF,  p-value: 6.838e-07
```

Analysis of variance (ANOVA) (statsmodels)

$$H_0 : \beta_1 = \cdots = \beta_p = 0 \qquad \text{against} \qquad H_1 : \beta_j \neq 0 \text{ for some } j \in \{1, \ldots, p\}$$

- Under the null H_0 (and with a gaussian noise), one can show that the test statistic statisfies

$$T = \frac{(\text{TSS} - \text{RSS})/(p-1)}{\text{RSS}/(n-p)} \sim F_{p-1,n-p}$$

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Species   R-squared:                       0.766
Model:                            OLS   Adj. R-squared:                  0.717
Method:                 Least Squares   F-statistic:                     15.70
Date:                Mon, 20 Jun 2022   Prob (F-statistic):           6.84e-07
Time:                        16:28:18   Log-Likelihood:                -162.54
No. Observations:                  30   AIC:                             337.1
Df Residuals:                      24   BIC:                             345.5
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
```

Analysis of variance (ANOVA) (R)

$$H_0 : \beta_1 = \cdots = \beta_p = 0 \qquad \text{against} \qquad H_1 : \beta_j \neq 0 \text{ for some } j \in \{1, \ldots, p\}$$

- Under the null H_0 (and with a gaussian noise), one can show that the test statistic statisfies

$$T = \frac{(\text{TSS} - \text{RSS})/(p-1)}{\text{RSS}/(n-p)} \sim F_{p-1,n-p}$$

```
Residual standard error: 60.98 on 24 degrees of freedom
Multiple R-squared:  0.7658,Adjusted R-squared:  0.7171
F-statistic:  15.7 on 5 and 24 DF,  p-value: 6.838e-07
```

Anova (II) (statsmodels)

$H_0 : \beta_{\text{Area}} = \beta_{\text{Adjacent}} = 0$ against H_1 : at least one of the two is non null

```
>>> import faraway.datasets.galapagos
>>> import statsmodels.api as sm
>>> import statsmodels.formula.api as smf
>>>
>>> galapagos = faraway.datasets.galapagos.load()
>>>
>>> form = 'Species ~ Area + Elevation + Nearest + Scruz + Adjacent'
>>> form0 = 'Species ~ Elevation + Nearest + Scruz'
>>> fit = smf.ols(form, galapagos).fit()
>>> fit0 = smf.ols(form0, galapagos).fit()

>>> sm.stats.anova_lm(fit0, fit)
  df_resid            ssr  df_diff       ss_diff         F   Pr(>F)
0      26.0  158291.628568      0.0           NaN       NaN      NaN
1      24.0   89231.366330      2.0  69060.262238  9.287352  0.00103
```

Anova (II) (R)

```
> library(faraway)
> data(gala)
> fit <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, data = gala)
> fit0 <- lm(Species ~ Elevation + Nearest + Scruz, data = gala)
> anova(fit, fit0)
Analysis of Variance Table

Model 1: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
Model 2: Species ~ Elevation + Nearest + Scruz
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     24  89231
2     26 158292 -2    -69060 9.2874 0.00103 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Information criterion

- Rather than using hypothesis test, one could rely on information criterion.
- Information criterion is just a numeric value that summarizes the overall quality of a fitted model. The lower the better.
- Two widely used information criterion are:
  - The Akaike Information Criterion (AIC)

$$AIC(\mathcal{M}) = \underbrace{-2\ell(\hat{\theta})}_{\text{goodness of fit}} + \underbrace{2\dim(\hat{\theta})}_{\text{model complexity}}, \qquad \hat{\theta} \text{ MLE of model } \mathcal{M}.$$

  - The Baysesian/Schwarz Information Criterion (BIC)

$$BIC(\mathcal{M}) = -2\ell(\hat{\theta}) + \dim(\hat{\theta}) \log n, \qquad \hat{\theta} \text{ MLE of model } \mathcal{M}.$$

Information criterion

- Rather than using hypothesis test, one could rely on information criterion.
- Information criterion is just a numeric value that summarizes the overall quality of a fitted model. The lower the better.
- Two widely used information criterion are:
  - The Akaike Information Criterion (AIC)

$$AIC(\mathcal{M}) = \underbrace{-2\ell(\hat{\theta})}_{\text{goodness of fit}} + \underbrace{2\dim(\hat{\theta})}_{\text{model complexity}}, \qquad \hat{\theta} \text{ MLE of model } \mathcal{M}.$$

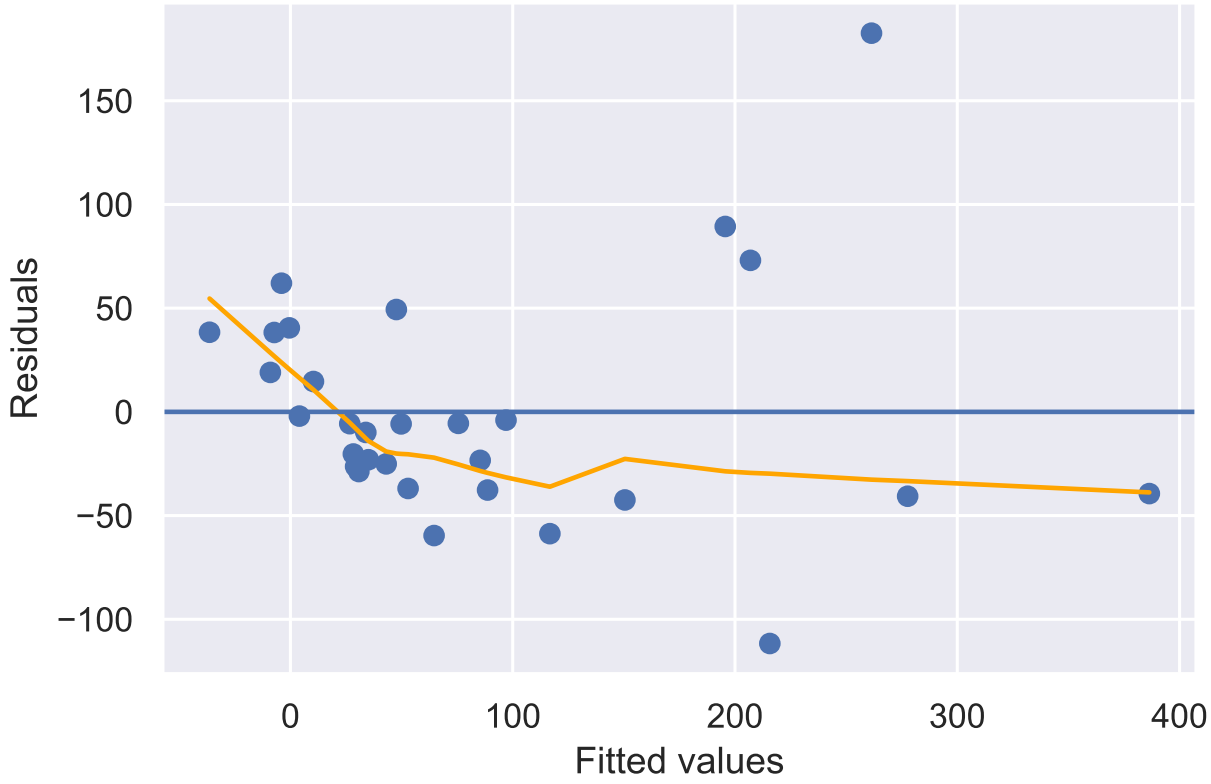
  - The Baysesian/Schwarz Information Criterion (BIC)

$$BIC(\mathcal{M}) = -2\ell(\hat{\theta}) + \dim(\hat{\theta}) \log n, \qquad \hat{\theta} \text{ MLE of model } \mathcal{M}.$$

☞ AIC and BIC have the advantage that it can be applied to non nested models! But beware AIC is not consistent while BIC is.

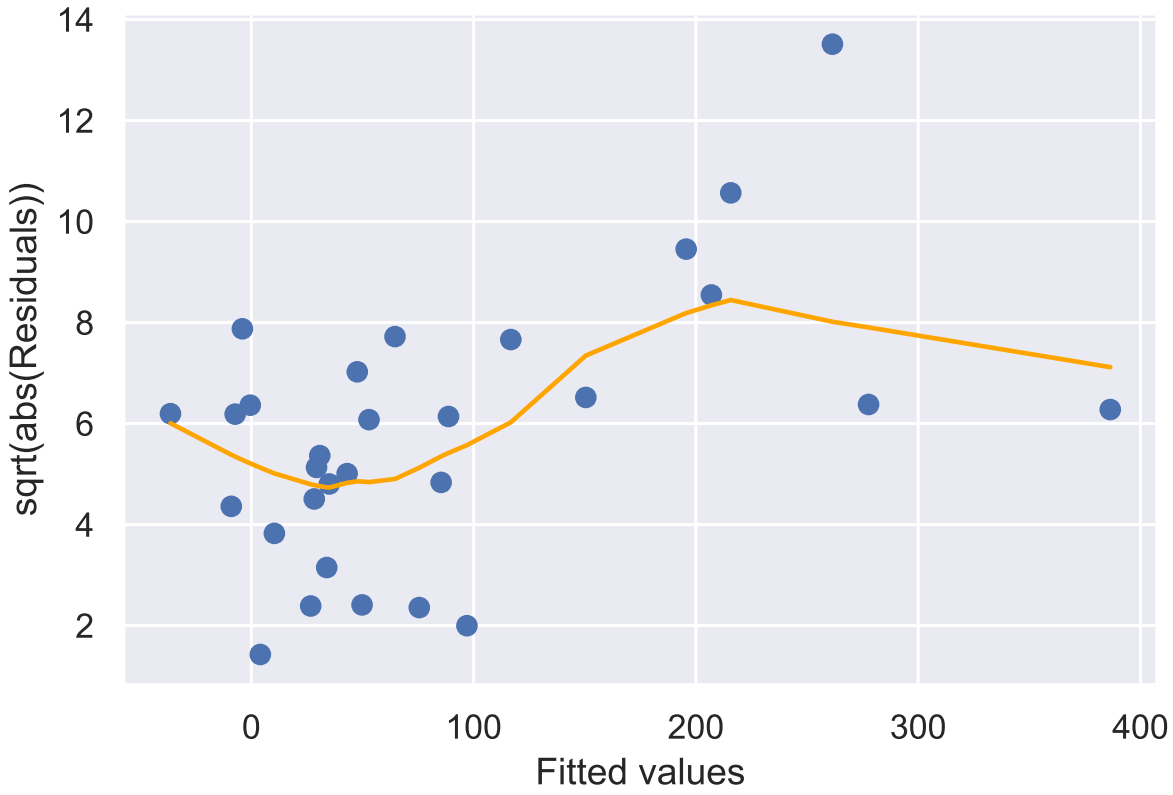
Residuals analysis

- Typically we check if the model assumptions are valid using plots :
  - white noise $\rightarrow$ plot residuals vs fitted values;
  - homoscedasticity $\rightarrow$ plot $\sqrt{|\text{residuals}|}$ vs fitted values;
  - Normality (if gaussian noise) using quantile-quantile plots



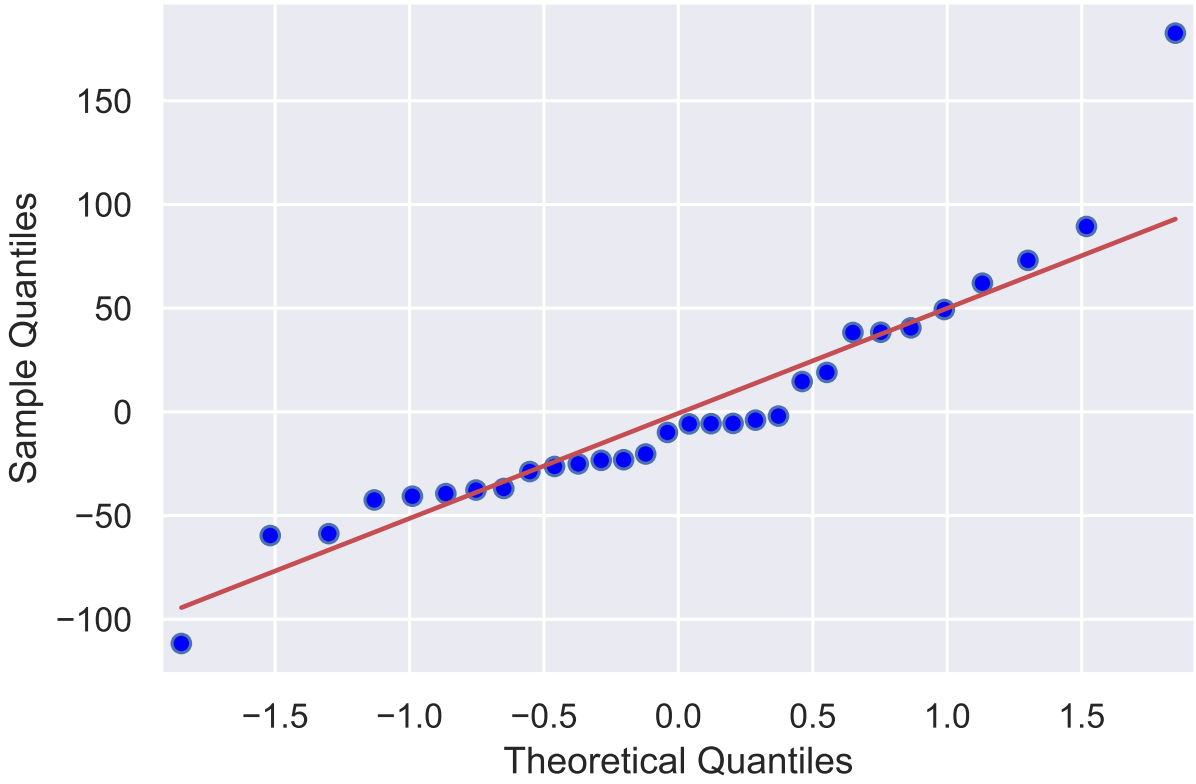
Residuals analysis

- Typically we check if the model assumptions are valid using plots :
  - white noise $\rightarrow$ plot residuals vs fitted values;
  - homoscedasticity $\rightarrow$ plot $\sqrt{|\text{residuals}|}$ vs fitted values;
  - Normality (if gaussian noise) using quantile-quantile plots



Residuals analysis

- Typically we check if the model assumptions are valid using plots :
  - white noise $\rightarrow$ plot residuals vs fitted values;
  - homoscedasticity $\rightarrow$ plot $\sqrt{|\text{residuals}|}$ vs fitted values;
  - Normality (if gaussian noise) using quantile-quantile plots

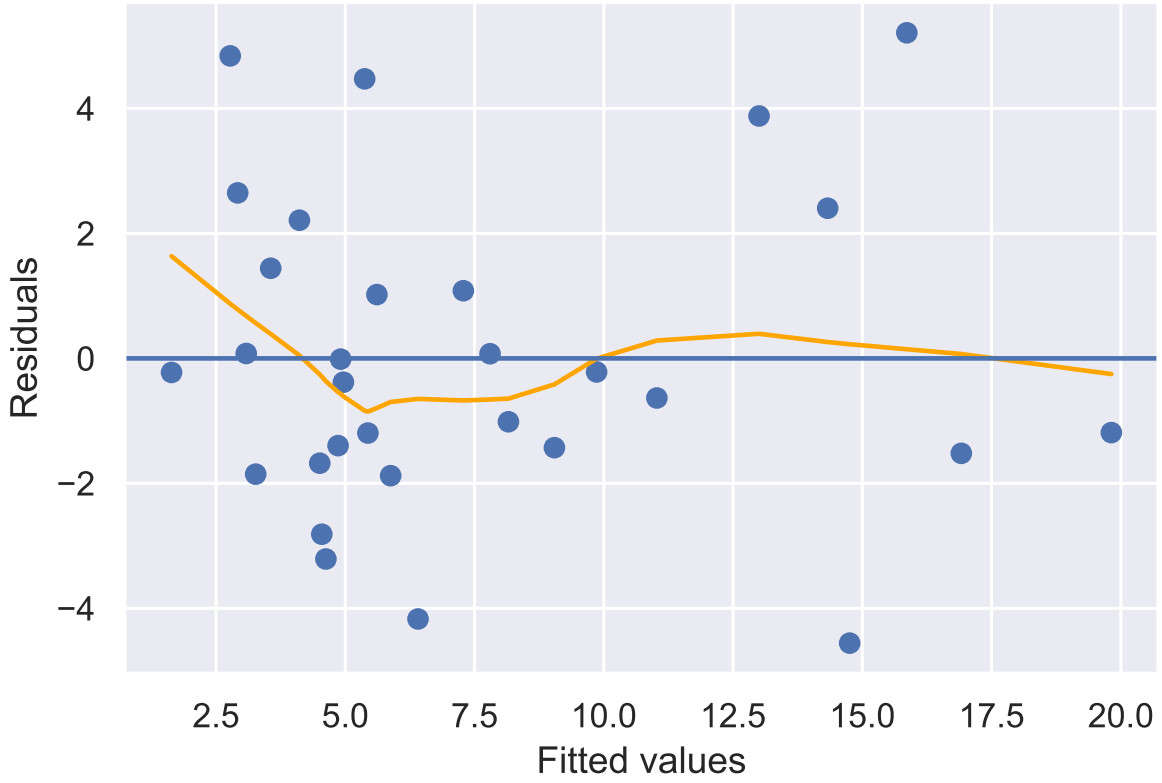


Galapagos revisited

- The two first diagnostic plots suggest problems.
- One way to fix it is to transform the response variable.
- Theory tells that a sensible transformation for counts is $y \mapsto \sqrt{y}$.

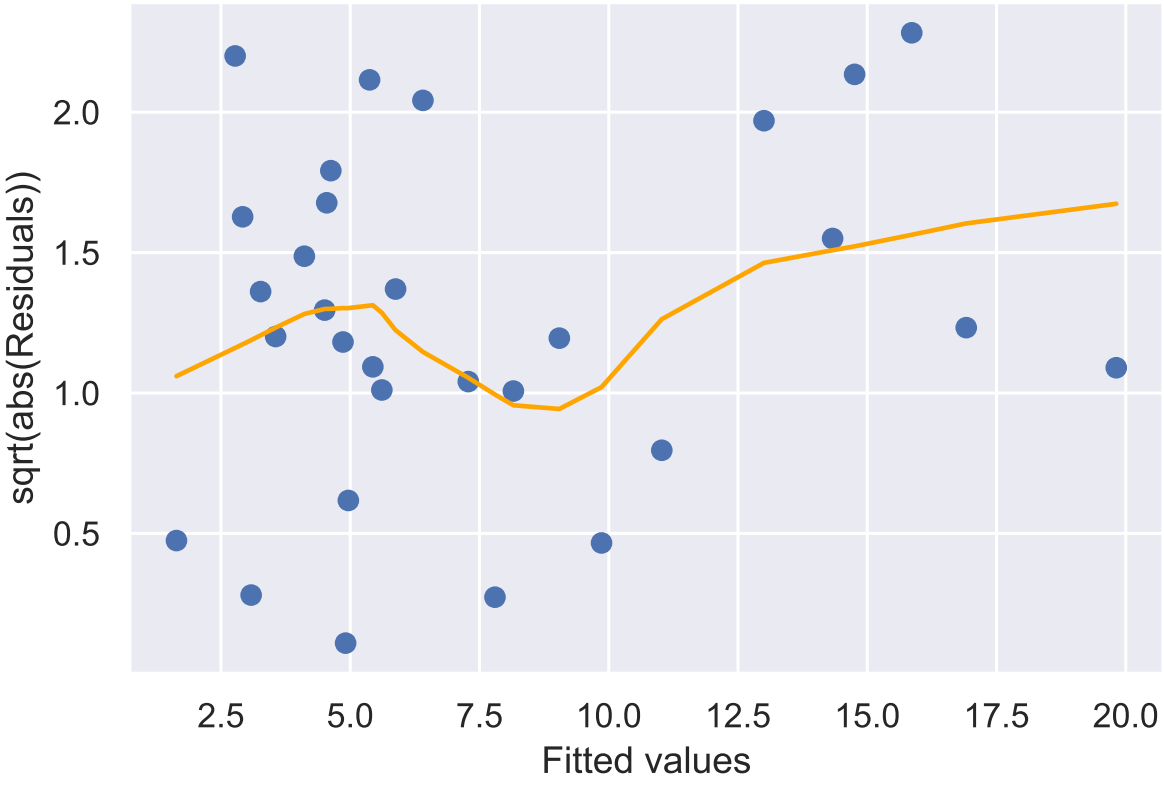
Galapagos revisited

- The two first diagnostic plots suggest problems.
- One way to fix it is to transform the response variable.
- Theory tells that a sensible transformation for counts is $y \mapsto \sqrt{y}$.



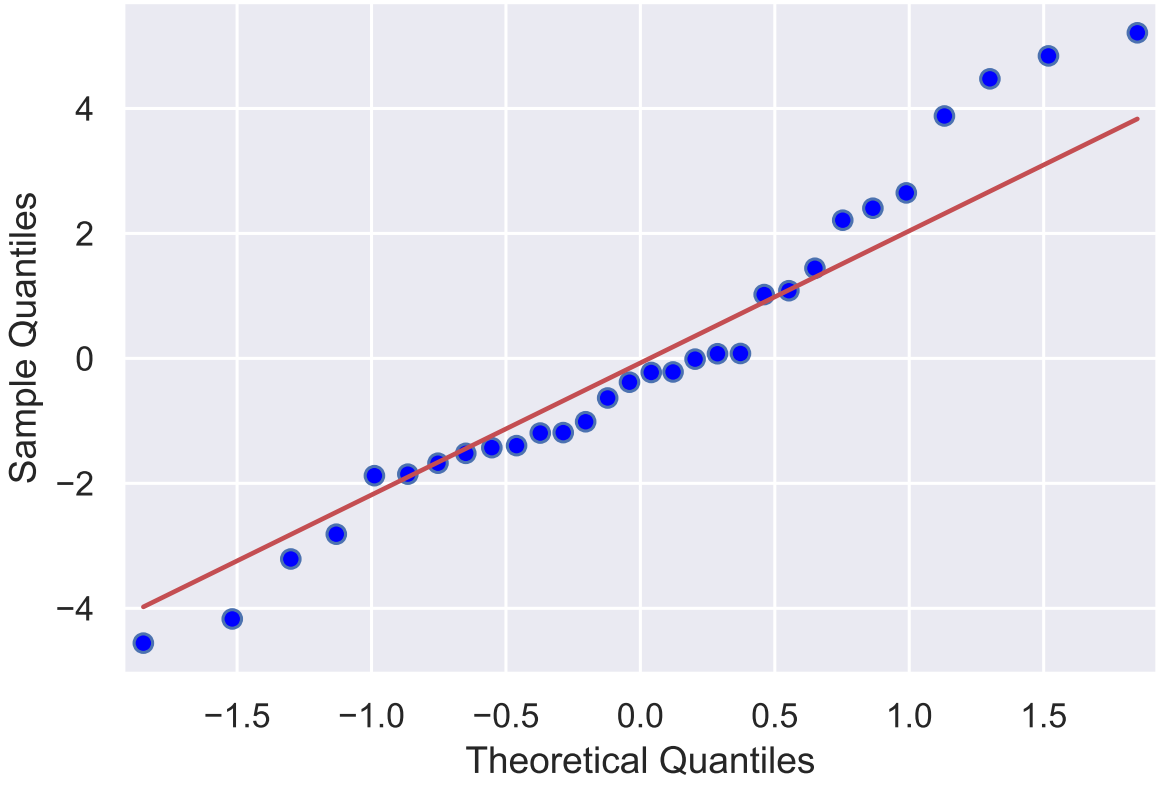
Galapagos revisited

- The two first diagnostic plots suggest problems.
- One way to fix it is to transform the response variable.
- Theory tells that a sensible transformation for counts is $y \mapsto \sqrt{y}$.



Galapagos revisited

- The two first diagnostic plots suggest problems.
- One way to fix it is to transform the response variable.
- Theory tells that a sensible transformation for counts is $y \mapsto \sqrt{y}$.



Logistic regression

- Logistic regression is, to some extent, very similar to linear regression except that the response is binary, i.e., $Y \in \{0, 1\}$.
- Why this situation deserves a close attention?

Logistic regression

- Logistic regression is, to some extent, very similar to linear regression except that the response is binary, i.e., $Y \in \{0, 1\}$.
- Why this situation deserves a close attention?
- Because in many situations one want to have a binary response such as:
  - email is spam or not spam;
  - should I bring my jacket or not today?
  - should a bank grant a loan to you or not?
- Logistic regression is therefore often considered as a supervised classifier.

Logistic regression

- Logistic regression is, to some extent, very similar to linear regression except that the response is binary, i.e., $Y \in \{0, 1\}$.
- Why this situation deserves a close attention?
- Because in many situations one want to have a binary response such as:
  - email is spam or not spam;
  - should I bring my jacket or not today?
  - should a bank grant a loan to you or not?
- Logistic regression is therefore often considered as a supervised classifier.

☞ Logistic regression could be extented to more than 2 classes but most often different approaches are used in such situations.

Let's build the model together

- The response Y is binary and a sensible choice to model Y is thus the Bernoulli(p) distribution whose p.m.f. is

$$m(y) = p^y(1-p)^{1-y}, \qquad y \in \{0,1\}, \quad p = \Pr(Y=1) = \mathbb{E}(Y) \in [0,1]$$

- Now since it is sensible to let the probability of "success" p depends on some covariates x, we now have

$$Y \mid X = x \sim \text{Bernoulli}(p(x)).$$

- Working in a parametric setting and paralleling the linear regression model, we may assume the linear form

$$p(x) = x^\top \beta.$$

☞ Clearly not relevant since $x^\top \beta \in \mathbb{R}$!

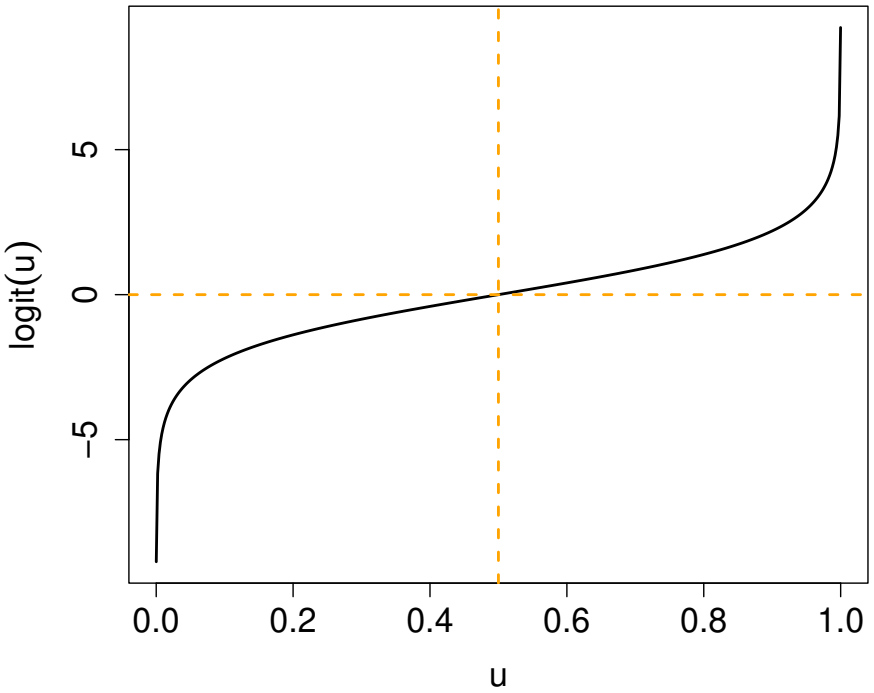
- To bypass this hurdle we thus need to define a one–one mapping η such that

$$\begin{aligned} \eta \colon (0,1) &\longrightarrow \mathbb{R} \\ u &\longmapsto \eta(u) \end{aligned}$$

and set $\eta(p(x)) = x^\top \beta$.

- Clearly the linear assumption on $\eta(p(x))$ now makes sense.
- The logistic regression model assumes that η is the logit function, i.e.,

$$\begin{aligned} \text{logit} \colon (0,1) &\longrightarrow \mathbb{R} \\ u &\longmapsto \log \frac{u}{1-u} \end{aligned}$$



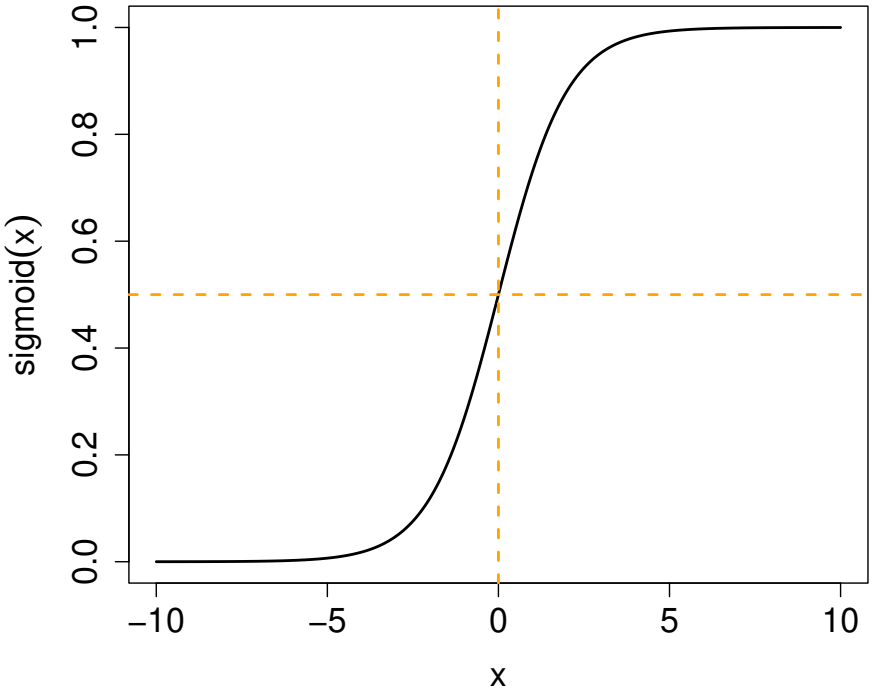
An aside: Sigmoid function

- We just defined the logistic function

$$\begin{aligned} \text{logit}\colon (0,1) &\longrightarrow \mathbb{R} \\ u &\longmapsto \log \frac{u}{1-u}. \end{aligned}$$

- The reciprocal of the logit function is nowadays very popular due to the hype of Neural Networks.
- It is known as the sigmoid function

$$\begin{aligned} \text{sigmoid}\colon \mathbb{R} &\longrightarrow (0,1) \\ x &\longmapsto \log \frac{\exp(x)}{1+\exp(x)}. \end{aligned}$$



An aside: Generalized Linear Models

- Actually both the linear and logistic regression models are special cases of Generalized Linear Models (GLM), i.e.,

$$\eta\left\{\mathbb{E}(Y \mid X)\right\} = x^\top \beta,$$

  where η is the link function.
- Here are some example of link functions and the corresponding model:

  Linear $\eta(u) = u$
  Logistic $\eta(u) = \operatorname{logit} u$
  Poisson $\eta(u) = \log u$
  Gamma $\eta(u) = -u^{-1}$

Fitting a logistic regression model

- Apart from the trivial case $\operatorname{logit} p(x) = \beta_0$, there is no closed form expression for the MLE;
- Gradient based optimization is typically used—most often Newton–Raphson that makes use of the Hessian matrix, i.e.,

$$\theta_{t+1} = \theta_t + \left\{\nabla^2_\theta \ell(\theta_t; \mathscr{D}_n)\right\}^{-1} \nabla_\theta \ell(\theta_t; \mathscr{D}_n),$$

  where $\nabla^2_\theta \ell(\theta_t; \mathscr{D}_n)$ is the Hessian matrix of $\ell(\theta_t; \mathscr{D}_n)$.
- Where for this particular model we have

$$\begin{aligned}\nabla_\theta \ell(\theta; \mathscr{D}_n) &= \mathbf{X}^\top \{\mathbf{Y} - p(\mathbf{X})\} \\ \nabla^2_\theta \ell(\theta; \mathscr{D}_n) &= -\mathbf{X}^\top \mathbf{W} \mathbf{X},\end{aligned}$$

  where $\mathbf{W}$ is a diagonal matrix whose diagonal is $p(\mathbf{X})\{1 - p(\mathbf{X})\}$.

☞ The above algorithm is known as the Fisher's scoring algorithm.

Predictions

- There are two types of predictions in a logistic regression model:

  response predictor which estimates $p(x)$ using $\hat{p}(x) = \text{sigmoid}(x^\top \hat{\beta})$;
  linear predictor which predicts $\text{logit}\, p(x) = x^\top \beta$ using $x^\top \hat{\beta}$.

- Both can serve as a guideline to predict the outcome Y given $X = x$.
- More precisely we use the following (binary) classifier

$$\hat{Y} \mid \{X = x\} = 1_{\{\hat{p}(x) > u\}} = 1_{\{x^\top \hat{\beta} > \text{logit}\, u\}},$$

  where u is a given threshold, i.e., most often but not invariably $u = 0.5$.

☞ In some cases you might not want to have to many "false alarms", i.e, $\hat{Y} = 1$ while $Y = 0$. Think about a spam filter. You can achieve this by increasing u, e.g., $u = 0.8$.

Residuals analysis

- Recall that residuals are given by

$$r_i = Y_i - \hat{Y}_i, \qquad i = 1, \ldots, n$$

- However since Y is binary, we thus have $r_i \in \{-1, 0, 1\}$ which is unfortunate to do diagnostic plots (but see later).
- Hence for logistic regression we rather define residuals as

$$r_i = Y_i - x^\top \hat{\beta}.$$

- Note however that there is still a side effect since

$$r_i = \begin{cases} 1 - x^\top \hat{\beta}, & Y_i = 1 \\ -x^\top \hat{\beta}, & Y_i = 0 \end{cases}$$

and thus provides artificial patterns.

What are the odds?

Definition 9. Given a probability p of some events, the associated odds are given by

$$\text{odds}(p) = \frac{p}{1-p} \in (0, \infty).$$

- The odds helps in dermining if an event having probability p to occcur is likely or not.
- More precisely,
  - $\text{odds}(p) > 1$ indicates the event is more likely to occur than it does not;
  - $\text{odds}(p) < 1$ indicates the event is less likely to occur than it does not.

Odds in a logistic regression model: quantitative case

- Recall that in logistic regression we have $p(x) = \Pr(Y = 1 \mid X = x)$.
- Hence

$$\text{odds}(p(x)) := \text{odds}(x) = \frac{p(x)}{1 - p(x)} = \exp\{\text{logit}\, p(x)\} = \exp\left(x^\top \beta\right).$$

- A typical interpretation of odds is when you add a "one unit increase" in quantitative covariate x_j and state how increased/decreased the odds.
- Indeed let $x_* = x$ except for, say, the p–th element which is $x_{*,p} = x_p + 1$. We get

$$\text{odds}(x_*) = \exp\left(x_*^\top \beta\right) = \exp\left(x^\top \beta + \beta_p\right) = \text{odds}(x) \exp(\beta_p).$$

☞ Depending on the sign of β_p, and all other covariates being fixed, we can tell if one unit increase in x_p increases the odds or not and even quantify the change from $\exp(\beta_p)$.

Odds in a logistic regression model: qualitative case

- For categorical variables the "one unit increase" has no sense, think about "blue + 1"!
- We can however still interpret the effect of a categorical variable, say x_p, on the odds.
- To this aim we first fix a reference level for x_p, e.g., blue.[7]
- Recall that if x_p has m levels, i.e., $x_p \in \{1, \ldots, m\}$, $x^\top \beta_p$ actually reads

$$\beta_{p,2} 1_{\{x_p=2\}} + \cdots + \beta_{p,m} 1_{\{x_p=m\}}.$$

- In the above expression level 1 is the baseline level.

☞ As previously, $\exp(\beta_{p,\ell})$ quantify the changes on the odds as we switch from baseline level to the ℓ–th one.

[7] It is also needed to ensure identifiability of the model parameters.

South African Heart Disease (Rousseauw et al., 1983)

- Coronary risk factor study survey carried out in 3 rural areas of the Western Cape in South Africa
- Aim: Establish the intensity of coronary heart disease (`chd`) factors in that high incidence region
- Data : While males between 15 and 64 and response variable is the presence or absence of myocardal infarction (MI)
- Overall prevalence in this region is 5.1%
- There are 160 cases in our data set and a sample of 302 controls.
- Motivation for this study was to educate people to have a balanced diet

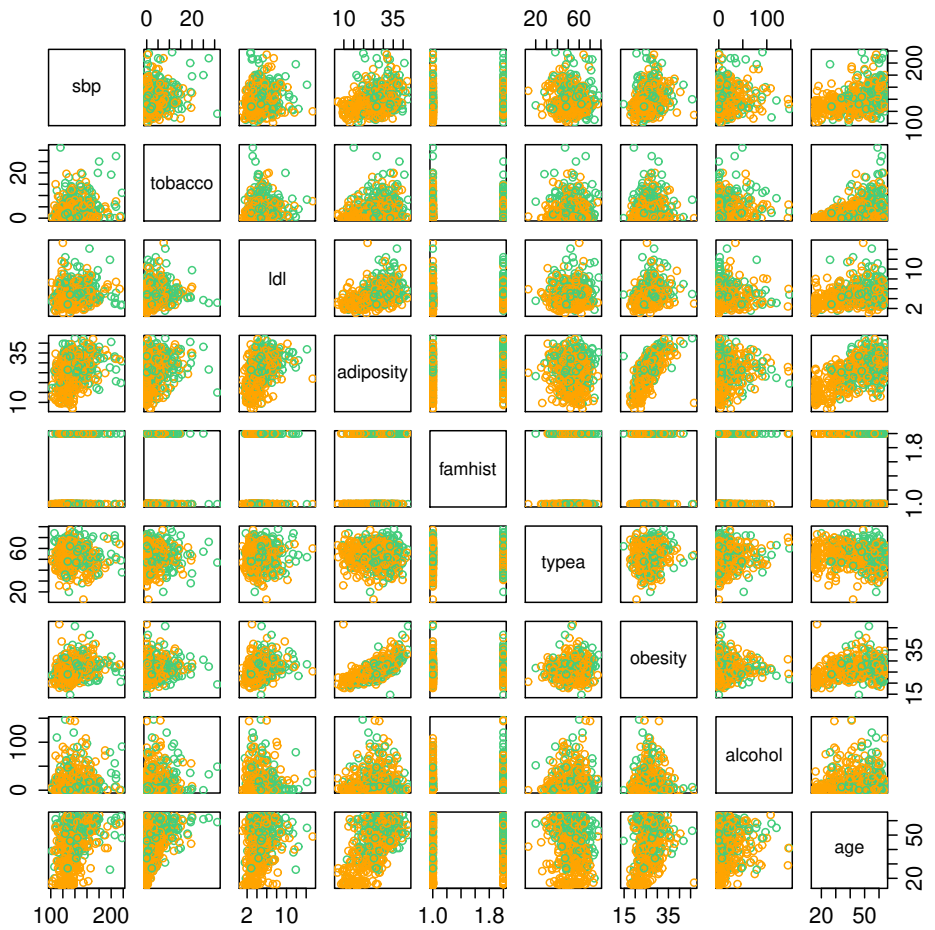


Figure 14: *Scatterplot of the South African heart disease dataset. Green: MI; col1: Control;* `famhist`*: 1 if family history of heart disease.*

```
> fit <- glm(chd ~ ., data = data, family = binomial)
> summary(fit)

Call:
glm(formula = chd ~ ., family = binomial, data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7781  -0.8213  -0.4387   0.8889   2.5435

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)    -6.1507209  1.3082600  -4.701 2.58e-06 ***
sbp             0.0065040  0.0057304   1.135 0.256374
tobacco         0.0793764  0.0266028   2.984 0.002847 **
ldl             0.1739239  0.0596617   2.915 0.003555 **
adiposity       0.0185866  0.0292894   0.635 0.525700
famhistPresent  0.9253704  0.2278940   4.061 4.90e-05 ***
typea           0.0395950  0.0123202   3.214 0.001310 **
obesity        -0.0629099  0.0442477  -1.422 0.155095
alcohol         0.0001217  0.0044832   0.027 0.978350
age             0.0452253  0.0121298   3.728 0.000193 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 472.14  on 452  degrees of freedom
AIC: 492.14

Number of Fisher Scoring iterations: 5
```

- The results sound a bit weird...

- The results sound a bit weird... Systolic blood pressure (`sbp`) is not significant??!!!???
- Idem for `obesity` which in addition is negative!

- The results sound a bit weird... Systolic blood pressure (`sbp`) is not significant??!!!???
- Idem for `obesity` which in addition is negative!
- This is a consequence of correlation between covariates. Need proof?

- The results sound a bit weird... Systolic blood pressure (`sbp`) is not significant??!!!???
- Idem for `obesity` which in addition is negative!
- This is a consequence of correlation between covariates. Need proof?

```
> summary(glm(chd ~ obesity, data = data, family = binomial))

Call:
glm(formula = chd ~ obesity, family = binomial, data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3396  -0.9257  -0.8558   1.4021   1.7116

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.92831    0.61692  -3.126  0.00177 **
obesity      0.04942    0.02318   2.132  0.03302 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 591.53  on 460  degrees of freedom
AIC: 595.53

Number of Fisher Scoring iterations: 4
```

Lesson to be learned

- You should interpret with caution non-significance of group of covariates.
- Ideally you should remove sequentially the least significant covariate until you couldn't drop anything
- Or, if you're a bit reckless, use `stepAIC` or variants

```
> library(MASS)
> fit.step <- stepAIC(fit)
> summary(fit.step)

Call:
glm(formula = chd ~ tobacco + ldl + famhist + typea + age, family = binomial,
    data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9165  -0.8054  -0.4430   0.9329   2.6139

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -6.44644    0.92087  -7.000 2.55e-12 ***
tobacco         0.08038    0.02588   3.106  0.00190 **
ldl             0.16199    0.05497   2.947  0.00321 **
famhistPresent  0.90818    0.22576   4.023 5.75e-05 ***
typea           0.03712    0.01217   3.051  0.00228 **
age             0.05046    0.01021   4.944 7.65e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation

```
> summary(fit.step)

Call:
glm(formula = chd ~ tobacco + ldl + famhist + typea + age, family = binomial,
    data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9165  -0.8054  -0.4430   0.9329   2.6139

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -6.44644    0.92087  -7.000 2.55e-12 ***
tobacco         0.08038    0.02588   3.106  0.00190 **
ldl             0.16199    0.05497   2.947  0.00321 **
famhistPresent  0.90818    0.22576   4.023 5.75e-05 ***
typea           0.03712    0.01217   3.051  0.00228 **
age             0.05046    0.01021   4.944 7.65e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- How do we interpret for instance `famhistPresent`?
- If a patient has a family history heart disease, it increases the odds of coronary heart disease of $\exp(0.90818) \approx 2.5$ or equivalently 150%.
- And a 95% confidence interval for this odds ratio is

$$\exp(0.90818 \pm 1.96 \times 0.22576) \approx [2, 3].$$

Logistic regression as a binary classifier

- Remember that the outcome Y for the logistic regression is binary.
- We suppose as well as the probability of "success", i.e., having $Y = 1$, depends on some covariates x as follows

$$\operatorname{logit} p(x) = x^T \beta, \qquad p(x) = \Pr(Y = 1 \mid X = x).$$

- Given some features x_*, how could we say that Y should be 1 or 0?
- One widely used way is to take

$$\hat{Y} = \begin{cases} 1, & p(x) \geq 0.5 \\ 0, & p(x) < 0.5. \end{cases}$$

Remark. The cutoff value $u = 0.5$ is arbitrary[8] and, depending on the application, one could use different levels $u \in (0, 1)$. Think about fraud detection.

[8] but has theoretical justifications

2. Regularized linear regression

When things goes wrong...

- We saw that, provided $(X^\top X)^{-1}$ exists, the least squares estimator in linear regression is given by

$$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top Y$$

- It may happens that $(X^\top X)$ is singular because:
  - some features are linearly dependent, i.e., X is not of full rank;
  - we have more features than observations, i.e., $p \gg n$.
  - the matrix $X^\top X$ is numerically instable—often the case when $p \approx n$.
- To overcome this issue one option consists in regularizing the optimisation problem by adding a penalty, i.e.,

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p+1}}{\arg\max} \, \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda \text{Penalty}(\boldsymbol{\beta}), \qquad \text{for some fixed } \lambda > 0.$$

Ridge regression

- The ridge regression consists in taking an ℓ_2 penalty, i.e., the optimization problem becomes

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\max}\ \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2.$$

- This is equivalent to, for some $T > 0$ related to λ,

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\max}\ \|Y - X\boldsymbol{\beta}\|_2^2 \qquad s.t. \quad \|\boldsymbol{\beta}\|_2^2 \leq T.$$

- Clearly there are two limiting situations:
  - When $\lambda \to 0$ (or $T \to \infty$), $\hat{\boldsymbol{\beta}}_\lambda \to \hat{\boldsymbol{\beta}}$;
  - When $\lambda \to \infty$ (or $T \to 0$), $\hat{\boldsymbol{\beta}}_\lambda \to \mathbf{0}$.
- For $\lambda > 0$ fixed, the solution to the ridge regression is given by

$$\hat{\boldsymbol{\beta}}_\lambda = (X^\top X + \lambda \mathsf{Id})^{-1} X^\top Y, \qquad \mathsf{Id} \text{ identity matrix.}$$

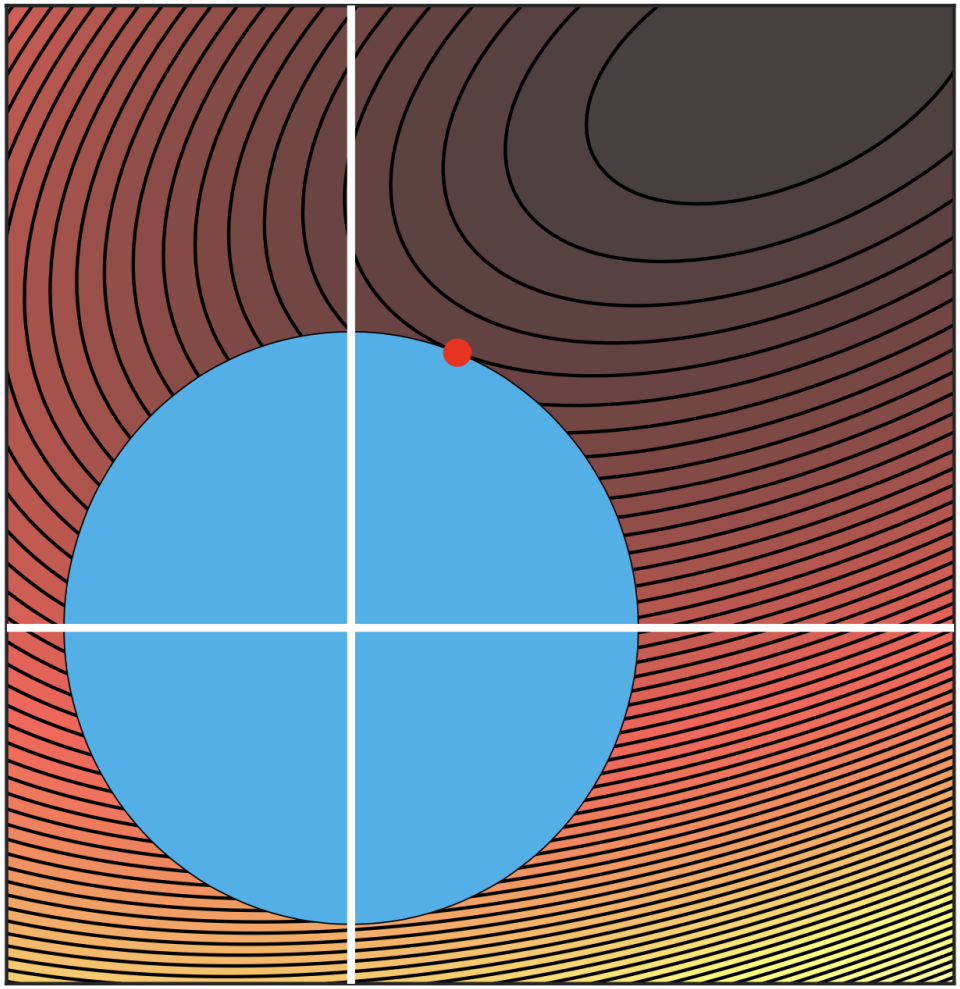


Figure 15: *Optimization with ℓ_2 penalization. The plots shows contours of the objective function and the ℓ_2 constraint $\|\beta\|_2^2 \leq T$. The solution to this constrained optimization problem is the red blob. [Figure taken from lecture note of Verzelen and Salmon].*

Beware

- What's the difference between these two linear models?

$$\text{Weight (kg)} = \beta_0 + \beta_1 \text{Height (cm)} + \beta_2 \text{IQ} + \varepsilon$$
$$\text{Weight (g)} = \beta_0 + \beta_1 \text{Height (m)} + \beta_2 \text{IQ} + \varepsilon$$

- None! However in the first model the two features should be (hopefully ;-)) of the same order while in the second we expect a ratio of 100.
- In other words, you don't want that the penalty depends on the order of magnitude of your features.

Beware

- What's the difference between these two linear models?

$$\text{Weight (kg)} = \beta_0 + \beta_1 \text{Height (cm)} + \beta_2 \text{IQ} + \varepsilon$$
$$\text{Weight (g)} = \beta_0 + \beta_1 \text{Height (m)} + \beta_2 \text{IQ} + \varepsilon$$

- None! However in the first model the two features should be (hopefully ;-)) of the same order while in the second we expect a ratio of 100.
- In other words, you don't want that the penalty depends on the order of magnitude of your features.

☞ Always scale (or check if it is done directly within your favourite program) your features.

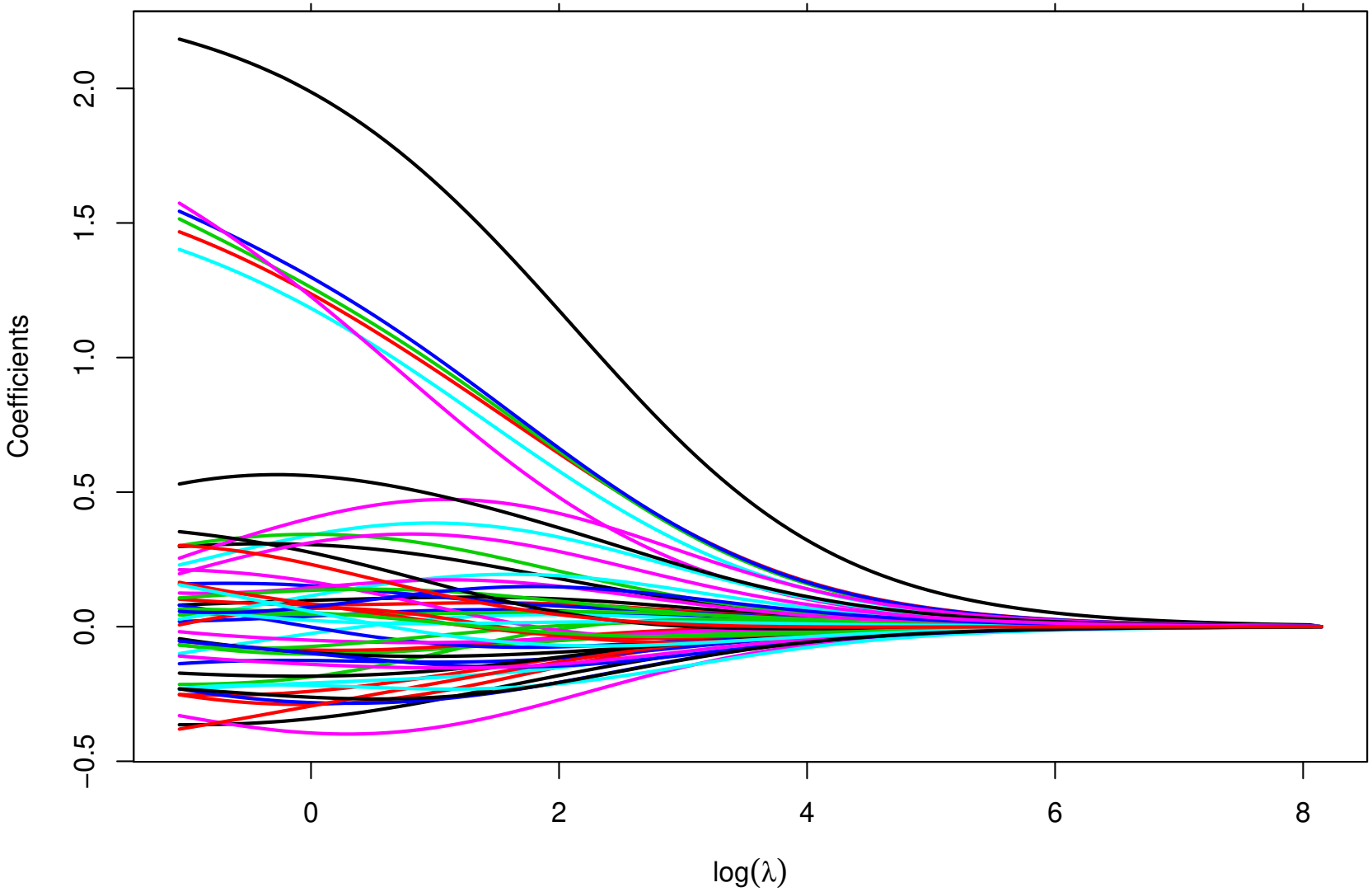


Figure 16: *Path of the ridge regression estimate $\hat{\beta}_\lambda$ as λ increases.*

How to choose λ ?

- Well we already know how to do it, right?
- The penalty coefficient λ is an example of an hyperparameter—a parameter that is tuned not estimated.
- Consequently we can tune it using cross validation as we talk about earlier in this lecture.

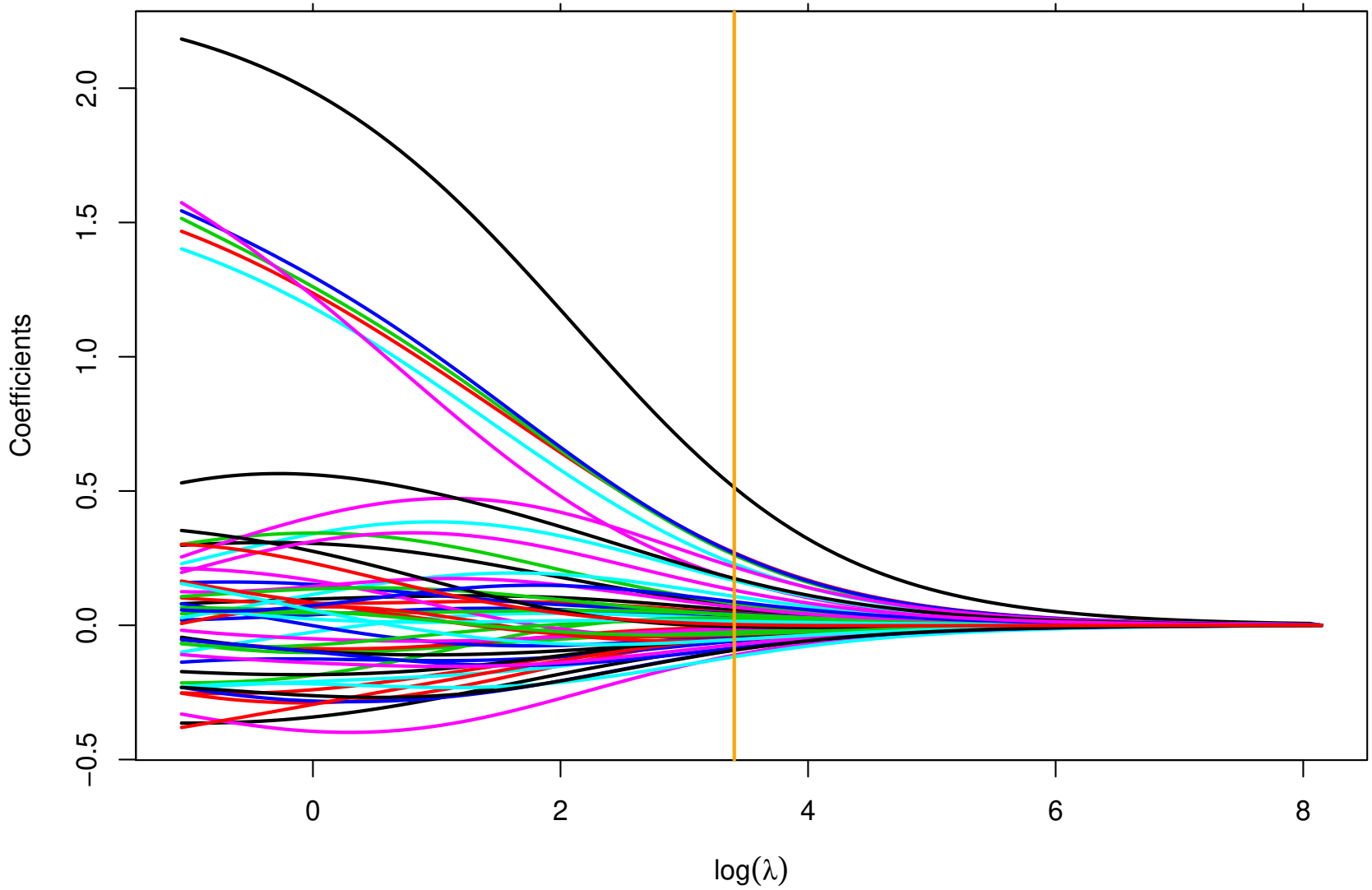


Figure 17: *Path of the ridge regression estimate $\hat{\beta}_\lambda$ as λ increases with the optimal choice for λ w.r.t. a 5−fold cross validation approach.*

- Ridge regression made a step forward for ill-posed regression problems
- However it is a bit unsatisfactory when $p \gg n$.
- Indeed in such situation we would like to have a sparse estimator for β. Why?
  - "automagically" identify the relevant covariates;
  - ease of interpretation;
  - numerical performance when $p \gg 1$.
- The Lasso was introduced to this aim and is very (very!) popular right now!

Back to feature selection

- We saw previously how to use backward, forward, stepwise selection to identify relevant predictors.
- Those approaches were greedy since addition/dropping out variables were done in an iterative way.
- Would it be possible to rephrase our original optimization problem to favor parsimony?

Definition 10. For $x \in \mathbb{R}^n$, we define

$$\|x\|_p = \left(\sum_{j=1}^{n} |x_j|^p \right)^{1/p}, \qquad p > 0,$$

and with the special cases

$$\|x\|_p = \begin{cases} \sum_{j=1}^{n} 1_{\{x_j \neq 0\}}, & p = 0 \\ \max\{|x_j| : j = 1, \ldots, n\}, & p = \infty. \end{cases}$$

Remark. When $p \geq 1$, $\|\cdot\|_p$ is a norm and therefore is convex (as a consequence of the triangle inequality). Remind that ℓ_0 is not a norm and is neither convex!

Limitation of the ℓ_0 penalty

- To force parsimony on the estimation of $\boldsymbol{\beta}$ one may tempted to solve

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_0$$

- Because $\|\cdot\|_0$ is non convex, usual optimization techniques do not apply.
- It is actually a combinatorial optimization problem with 2^p possibilities. Hence intractable in practice.

Limitation of the ℓ_0 penalty

- To force parsimony on the estimation of $\boldsymbol{\beta}$ one may tempted to solve

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_0$$

- Because $\|\cdot\|_0$ is non convex, usual optimization techniques do not apply.
- It is actually a combinatorial optimization problem with 2^p possibilities. Hence intractable in practice.

☞ We aim at getting the "closest" penalty that is convex. Therefore ℓ_1 is a sensible candidate for this!

Lasso (Least Absolute Shrinkage and Selection Operator)

- The lasso consists in taking an ℓ_1 penalty, i.e., the optimization problem becomes

$$\tilde{\boldsymbol{\beta}}_\lambda \in \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \; \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1.$$

- This is equivalent to, for some $T > 0$ related to λ,

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\max} \; \|Y - X\boldsymbol{\beta}\|_2^2 \qquad s.t. \quad \|\boldsymbol{\beta}\|_1 \leq T.$$

- Clearly there are two limiting situations:
  - When $\lambda \to 0$ (or $T \to \infty$), $\tilde{\boldsymbol{\beta}}_\lambda \to \hat{\boldsymbol{\beta}}$;
  - When $\lambda \to \infty$ (or $T \to 0$), $\tilde{\boldsymbol{\beta}}_\lambda \to \mathbf{0}$.
- There is not always a unique solution—think about having two identical columns
- No closed form expression for $\tilde{\boldsymbol{\beta}}_\lambda$.

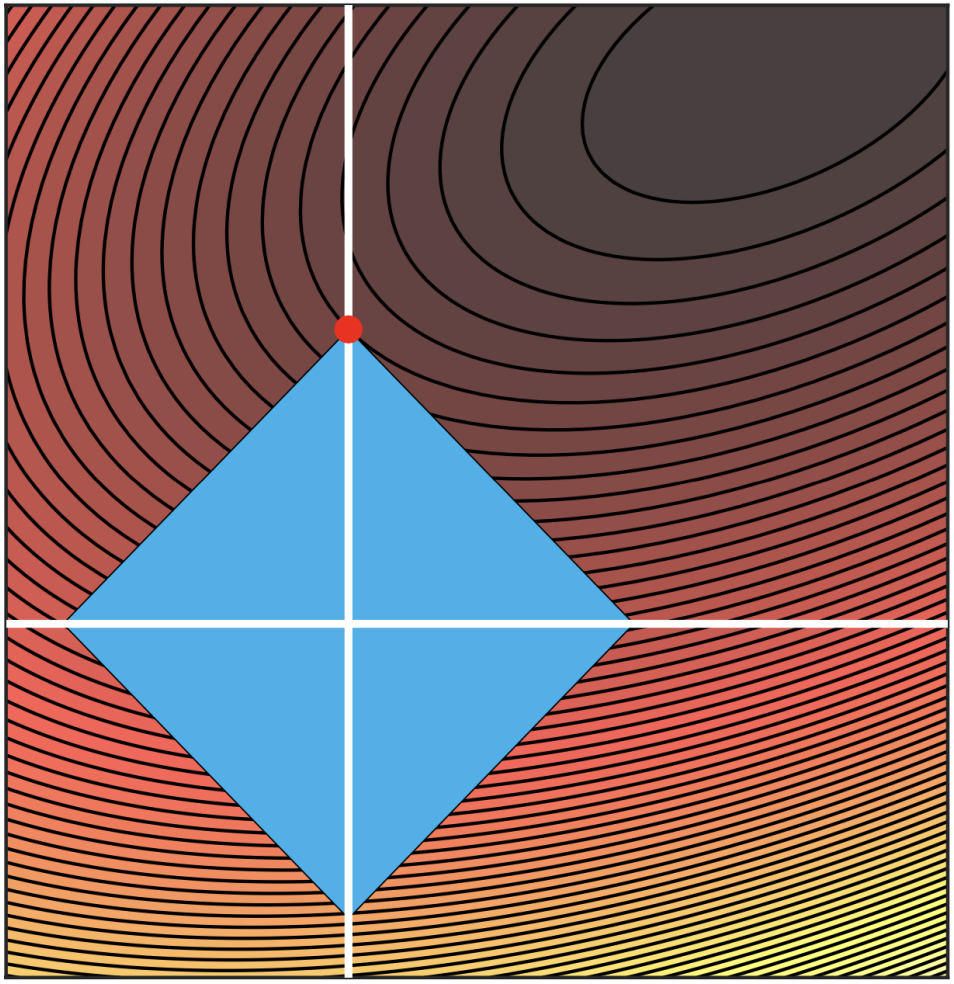


Figure 18: *Optimization with ℓ_1 penalization. The plots shows contours of the objective function and the ℓ_1 constraint $\|\beta\_1 \leq T$. The solution to this constrained optimization problem is the red blob. [Figure taken from lecture note of Verzelen and Salmon].*

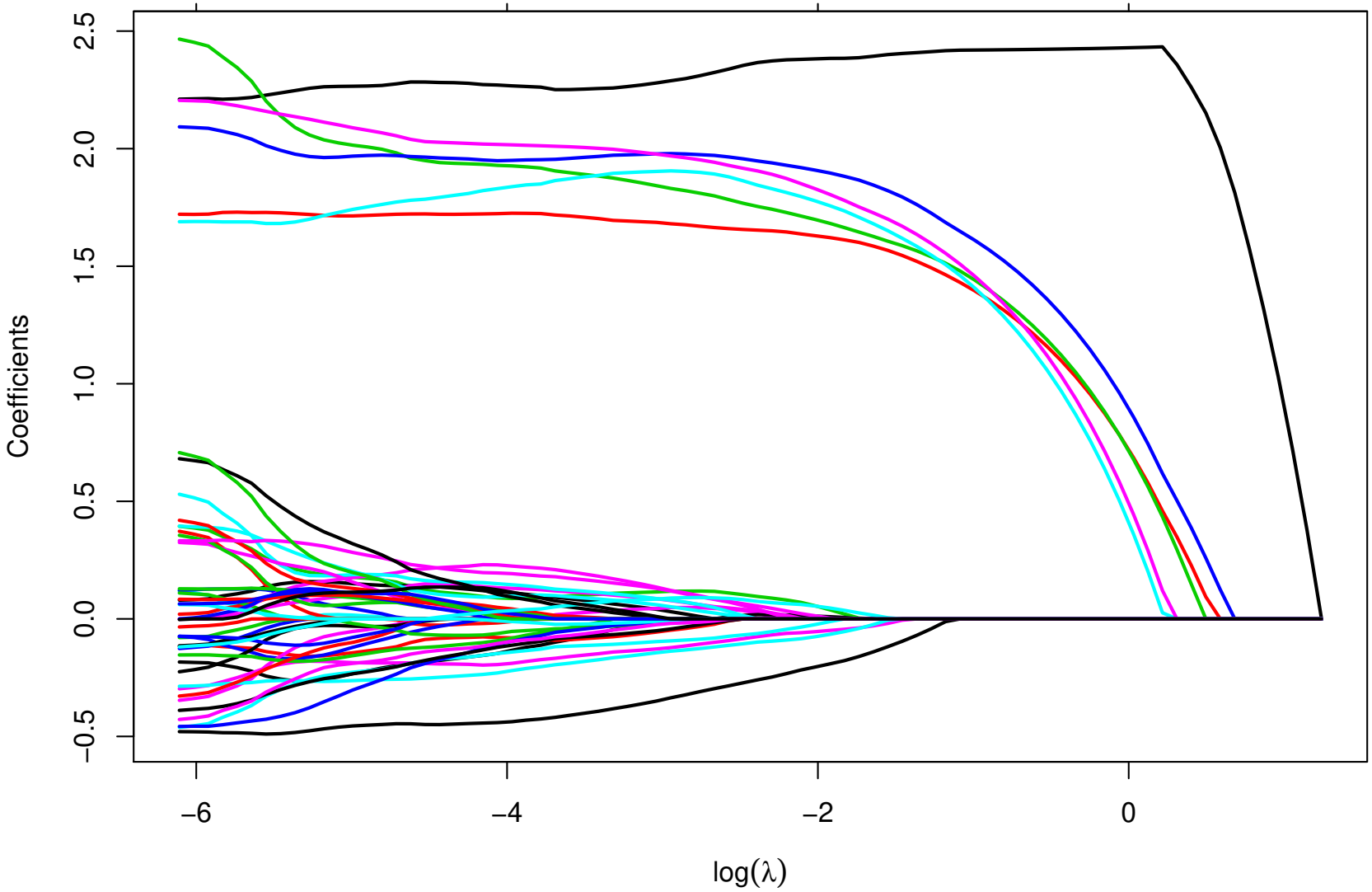


Figure 19: *Path of the Lasso regression estimate $\hat{\beta}_\lambda$ as λ increases.*

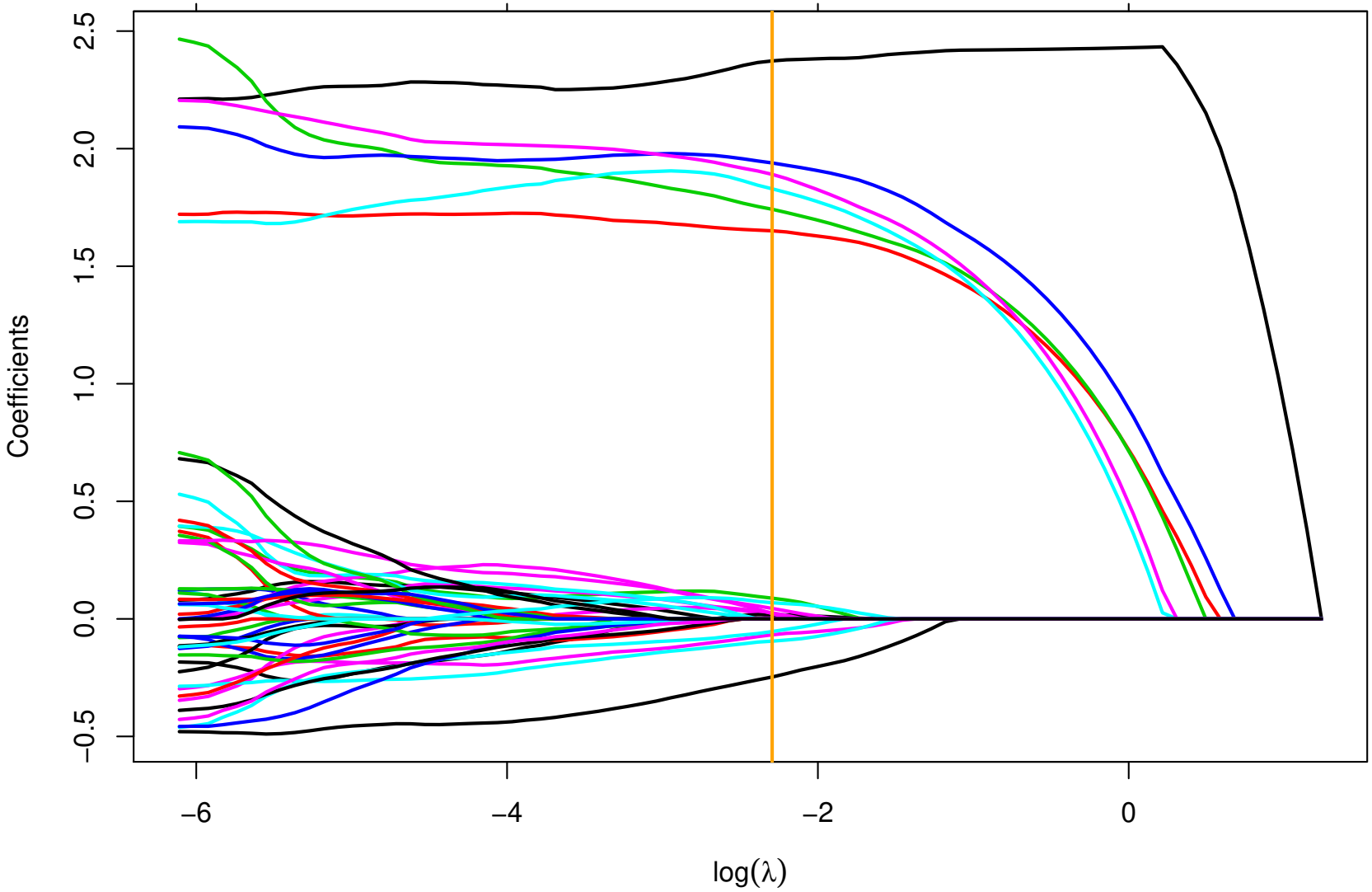


Figure 20: *Path of the Lasso regression estimate $\hat{\beta}_\lambda$ as λ increases with the optimal choice for λ w.r.t. a 5−fold cross validation approach.*

Elastic net

- Elastic net is a combination of the ridge and lasso penalties, i.e.,

$$\tilde{\boldsymbol{\beta}}_{\lambda,\alpha} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \; \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda \left(\alpha \|\boldsymbol{\beta}\|_1 + \frac{1-\alpha}{2} \|\boldsymbol{\beta}\|_2^2 \right), \qquad 0 \leq \alpha \leq 1.$$

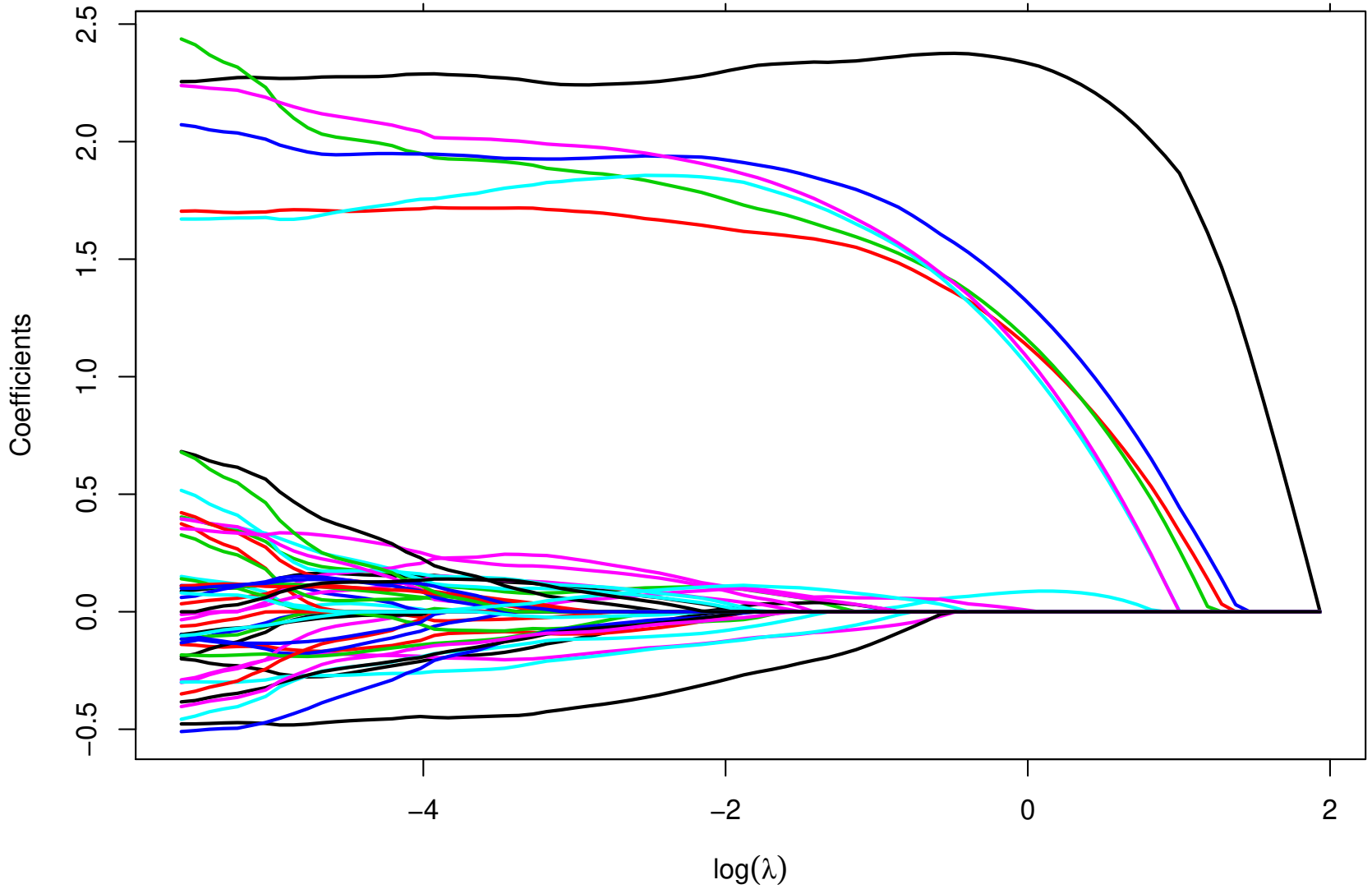


Figure 21: *Path of the Elastic Net regression estimate* $\hat{\beta}_{\lambda,\alpha}$ *with* $\alpha = 0.5$ *as* λ *increases with* $\alpha = 0.5$.

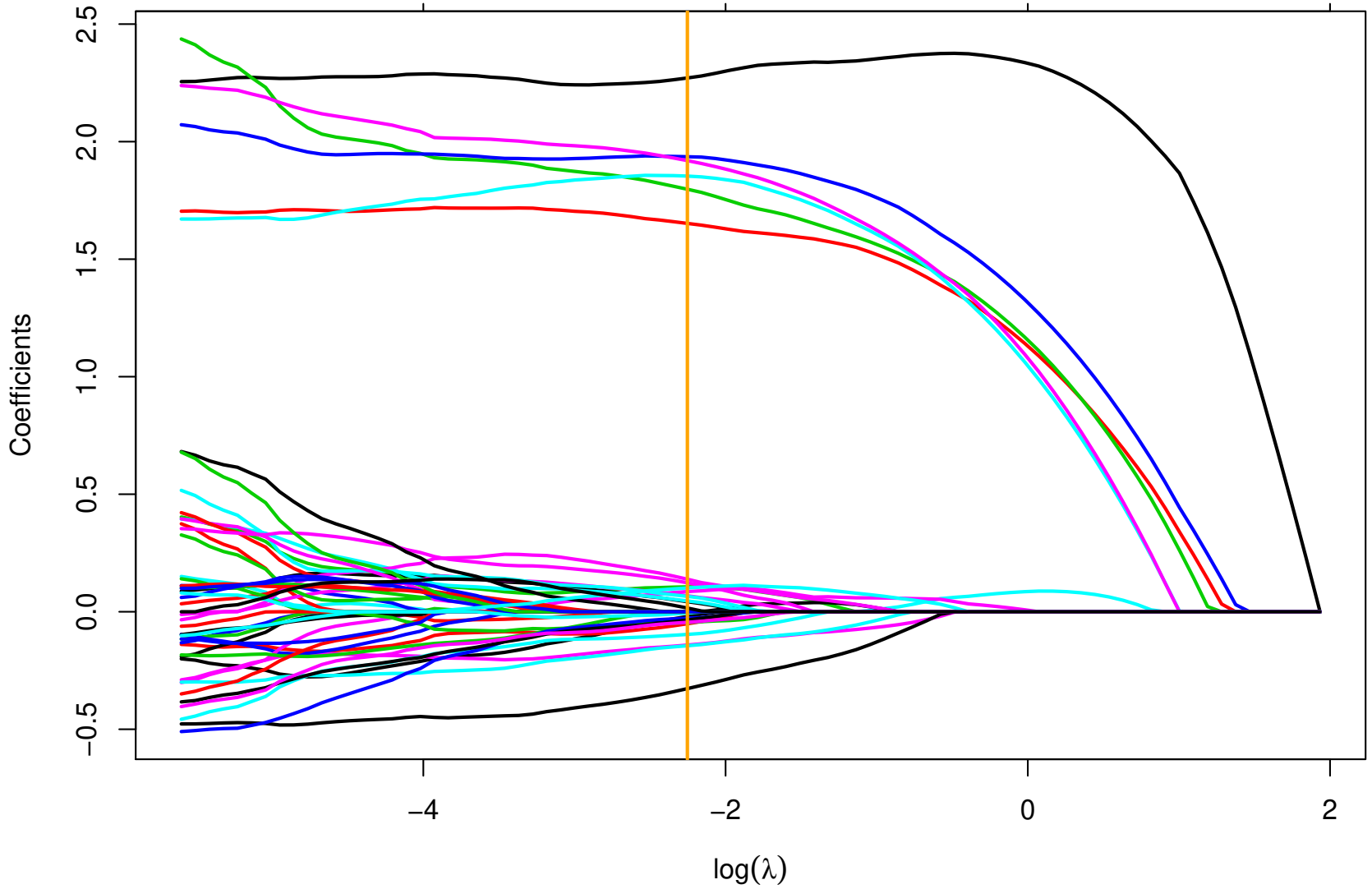


Figure 22: *Path of the Elastic Net regression estimate $\hat{\beta}_{\lambda,\alpha}$ with $\alpha = 0.5$ as λ increases with the optimal choice for λ w.r.t. a 5=fold cross validation approach.*

LET'S MOVE TO THE LAB!

3. Neural networks

Motivations

$$Y = f(X; \theta) + \varepsilon$$

- So far we saw that linear structures a widely used in statistical modelling
- This assumption of linearity is a bit restrictive
- There exists statistical models that are non linear
- One of those is neural networks
- Neural nets put a very specific structure on f, i.e.,

$$f(x) = f_d \circ f_{d-1} \circ \cdots \circ f_1(x),$$

where obviously, at least one of the f_ℓ is non linear.

The basic element: Artificial neuron

We define an artificial neuron (also called unit) as

$$f_\ell(z) = \varphi_\ell(\omega_\ell^\top z + b_\ell),$$

where $\theta_\ell = (\omega_\ell, \beta_\ell)$ are parameters to estimate and φ_ℓ is a, most often non linear, function called the activation function.

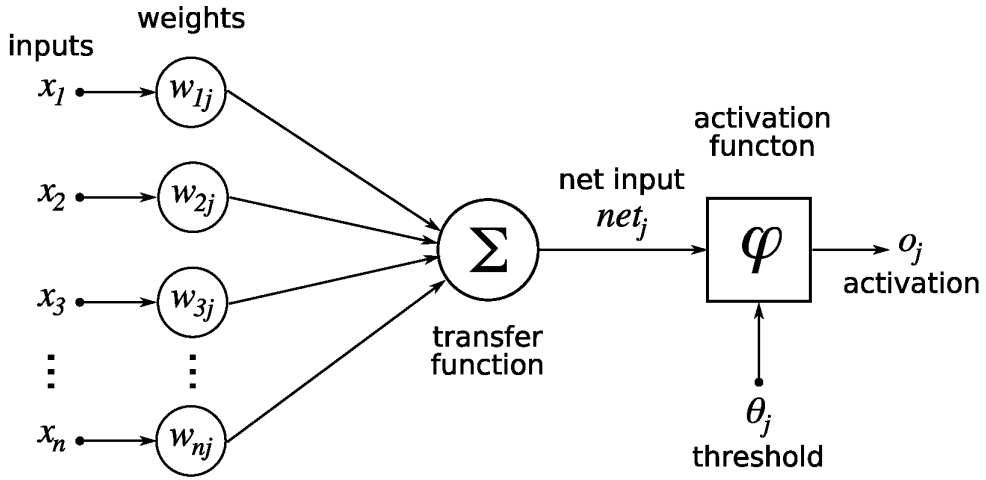


Figure 23: *Representation of a artificial neuron. Source wikibook*

Common activation functions

Identity $\varphi(z) = z$
RELU Rectified Linear Units $\varphi(z) = \max(0, z)$
Sigmoïd Reciprocal of the logit function we already seen

$$\varphi(z) = \frac{1}{1 + \exp(-z)}$$

Tanh Hyperbolic tangent

$$\varphi(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$

Softmax

$$\varphi(z) = \left(\frac{\exp(z_1)}{\sum_{j=1}^{p} \exp(z_j)}, \ldots, \frac{\exp(z_p)}{\sum_{j=1}^{p} \exp(z_j)} \right)^\top$$

Neural networks

- Neural nets are just a collection of (dependent) artificial neurons stacked into layers
- We say that it is a deep neural net when there are many hidden layers.
- Therefore fitting a neural networks amounts to:
  - Define the architecture of the neural network
  - Estimate the parameter $\theta = \{(\omega_j, b_j) : j = 1, \ldots, L\}$.

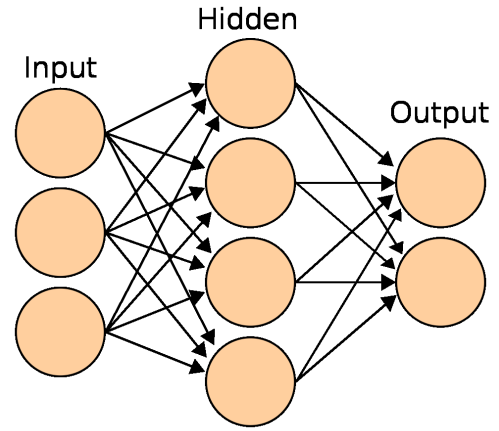


Figure 24: *A one (hidden) layer neural network. Source wikipedia.*

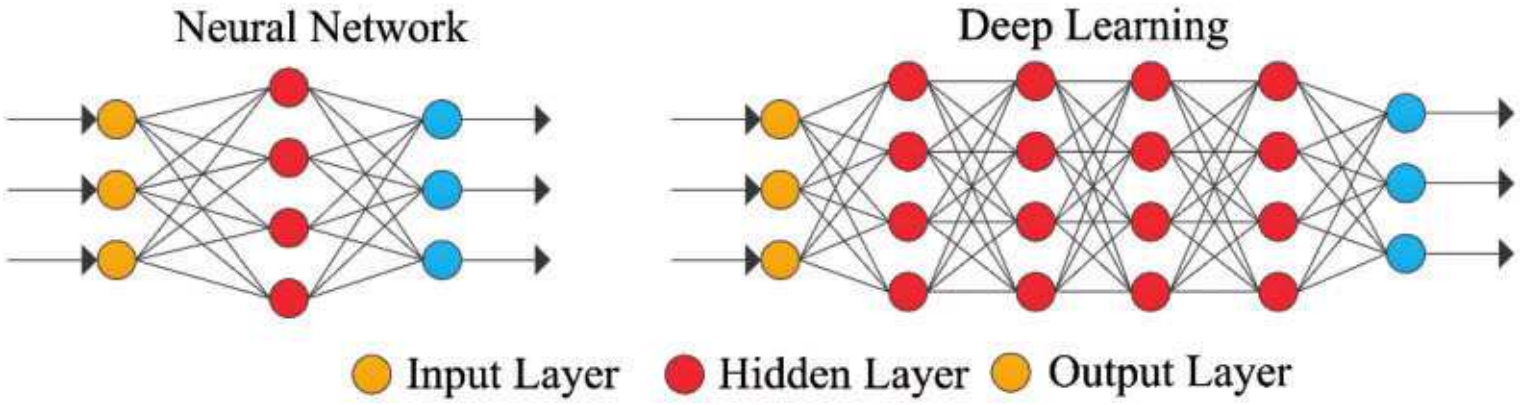


Figure 25: *A simple neural network and a deep one. Taken from ResearchGate*

Focus on one layer

- Let focus on the the ℓ-th layer, $\ell \in \{2, \ldots, L\}$.
- It consists of
  - s_ℓ artificial neurons, typically of the form $s_\ell = 2^{n_\ell}$;
  - an activation function φ_ℓ.
- Both need to be specified by the user
- s_ℓ is called the width of the layer
- Using matrix notation, one can write this layer has

$$a_\ell = \varphi_\ell(\mathbf{W}_\ell^\top a_{\ell-1} + \mathbf{b}_\ell),$$

  where $\mathbf{W}_\ell$ has columns $\omega_{\ell,1}, \ldots, \omega_{\ell,s_\ell}$, $\mathbf{b}_\ell = (b_{\ell,1}, \ldots, b_{\ell,s_\ell})^\top$ and z_ℓ is the output of the previous layer.
- Consequently $\mathbf{W}_\ell$ is a $s_{\ell-1} \times s_\ell$ matrix and $a_\ell, \mathbf{b}_\ell \in \mathbb{R}^{s_\ell}$.

Focus on the input and output layers

Input layer

- The neural net is fed with the features $X \in \mathbb{R}^p$
- As a consequence the weight matrix $\mathbf{W}_1$ has dimension (p, s_1).

Output layer

- The output of the neural net is driven by the response Y
- As a consequence the output layer will have a width 1 for (univariate) regression and width K for a K–label classification problem.

Which activation function you I use?

- Apart from the output layer where, as we will see in the next slide, there is no general answer to this question.
- However RELU is (by far) the most popular choice because:
  - \+ A kind of "least non linear" activation function (helps optimizing)
  - \+ Sparsification effect
  - \- Derivative is 0 on $(-\infty, 0) \Rightarrow$ slows the gradient descent algorithm
- To bypass this last issue, some people suggests the use of a leaky RELU

$$LeakyRELU(z) = \begin{cases} z, & z > 0 \\ \alpha z, & z \leq 0 \end{cases}$$

for some fixed (but small) $\alpha > 0$—e.g., $\alpha = 0.3$.

Activation function for the output layer

- Compared to the other ones, the output layer is very specific in that it should mimic the response Y.
- In a regression context, i.e., estimating $\mathbb{E}(Y \mid X)$, it is desirable to use the identity activation function

$$\varphi_L(z) = z.$$

- In a classification context with K labels, it is desirable to use the softmax activation function

$$\varphi_L(z) = \left(\frac{\exp(z_1)}{\sum_{k=1}^K z_k}, \ldots, \frac{\exp(z_K)}{\sum_{k=1}^K z_k} \right)^\top.$$

Why neural nets work?

Theorem 2 (Cybenko (1989)). *Let σ be a continuous monotone function such that $\lim_{t\to-\infty}\sigma(t)=0$ and $\lim_{t\to\infty}\sigma(t)=1$. Then the set of functions of the form $f(x)=\sum_j \alpha_j\sigma(\omega_j^\top x+b_j)$ is dense in $\mathscr{C}([0,1]^n)$ (equipped with the metric $d(f,g)=\sup|f(x)-g(x)|$).*

- This theorem tells us that we can approximate any continuous function (whose support is compact) with a one layer neural network (with a large width)
- However... practice shows that it is preferable to have several layers with smaller widths
- There is no cooking recipe for finding the "best" neural network architecture

(Empirical) Risk functions for Neural Nets

- Recall that fitting any statistical learning model, e.g., a neural net, often amounts to minimize the empirical risk

$$R(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i; \theta)),$$

where $\ell(\cdot, \cdot)$ is a loss function whose choice is driven by your application.

(Empirical) Risk functions for Neural Nets

- Recall that fitting any statistical learning model, e.g., a neural net, often amounts to minimize the empirical risk

$$R(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i; \theta)),$$

where $\ell(\cdot, \cdot)$ is a loss function whose choice is driven by your application.

- In practice we typically use

Regression a quadratic loss

$$\ell(y, y') = \|y - y'\|_2^2$$

Classification (with K labels) the cross-entropy

$$\ell(y, y') = -y^\top \ln y', \quad y, y' \in \mathbb{S}_{K-1} = \{u \in [0,1]^K : \sum_{k=1}^{K} u_k = 1\}.$$

(Theoretical) loss functions

- As said previously, you should never forget that actually we would like to minimize the theoretical risk

$$\arg\min_{\theta\in\Theta} \mathbb{E}\{\ell(Y; f(X;\theta))\},$$

  where the expectation is taken w.r.t. the unknown joint distribution of (X, Y).
- Since this optimization problem is out of reach, we rather consider an approximation of it

$$\arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell(Y; f(X;\theta)),$$

  hence justifying the phrasing empirical loss function.
- This fully justify why we often, if not always, use the train + validation + test splitting.

Summing up neural networks

- A neural network has:
  - a deepness L which consists of L layers; layer 1 and L being respectively the input and output layers.
  - Each layer $\ell \in \{1, \ldots, L\}$ has a width s_ℓ which corresponds to number of artificial neurons.
  - Layers are connected to other ones, possibly themselves, through an activation function.
- Hence defining the architecture of a neural net consists in specifying the number of layers L, the width s_ℓ of each layers and the activation functions φ_ℓ.
- Fitting a given neural net consists in
  - defining a loss function also called error or cost
  - minimizing the empirical risk w.r.t. to the $\theta = \{(\mathbf{W}_\ell, \mathbf{b}_\ell) : \ell = 1, \ldots, L\}$.

How to minimize the empirical risk?

- First I really insists on the fact that we minimize

$$\frac{1}{n}\sum_{i=1}^{n}\ell(Y_i, f(X;\theta)) \qquad \text{w.r.t. } \theta.$$

- This means that here the architecture of the neural net is fixed! We will come back to the problem of defining an appropriate architecture later.
- Hence given a neural net, how do we find $\hat{\theta}$?
- Nothing really sophisticated here and we just use (variants) of gradient descent.

[MODE NEURAL NETWORK OFF]

Gradient descent

- Consider the optimization problem

$$x_* = \arg\min_{x \in \mathbb{R}^p} f(x),$$

  where f is the objective function (supposed to be differentiable).
- The gradient descent is an iterative procedure producing a sequence $\{x_n : n \geq 0\}$ with

$$x_{n+1} = x_n - \eta \nabla \ell(x_n), \qquad \eta > 0,$$

  such that (hopefully) $x_n \to x_*$ as $n \to \infty$.
- The hyper-parameter η is (nowadays) called the learning rate.

Definition 11. A function $f : \mathbb{R}^p \to \mathbb{R}$ is said to be convex if for all $x, y \in \mathbb{R}^p$ and $\lambda \in [0, 1]$ we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y).$$

When f is differentiable, an alternative definition is

$$f(y) \geq f(x) + (y - x)^\top \nabla f(x).$$

- Convexity ensures that if x_* is a local minimum then it is a global minimum.
- Hence unless the function $\ell(\theta)$ is convex, there is no guarantee that gradient descent will converge to the global minimum.
- In real world application, it is thus mandatory to define suitable starting values θ_0.

Definition 12. A function $f : \mathbb{R}^p \to \mathbb{R}$ is said to be L–smooth, $L > 0$, if $f \in \mathscr{C}^1(\mathbb{R}^p)$ and for all $x, y \in \mathbb{R}^p$ we have

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2.$$

Proposition 2. *If f is L–smooth, then for all $x, y \in \mathbb{R}^p$ we have*

$$f(y) \leq f(x) + (y - x)^\top \nabla f(x) + \frac{L}{2}\|x - y\|_2^2.$$

Definition 12. A function $f : \mathbb{R}^p \to \mathbb{R}$ is said to be L–smooth, $L > 0$, if $f \in \mathscr{C}^1(\mathbb{R}^p)$ and for all $x, y \in \mathbb{R}^p$ we have

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2.$$

Proposition 2. *If f is L–smooth, then for all $x, y \in \mathbb{R}^p$ we have*

$$f(y) \leq f(x) + (y - x)^\top \nabla f(x) + \frac{L}{2}\|x - y\|_2^2.$$

☞ Idea: (Sequentially) Minimize the r.h.s. of the above inequation

Proof. Start with

$$f(y) = f(x) + \int_0^1 (y-x)^\top \nabla f(x + t(y-x)) \mathrm{d}t$$
$$= f(x) + (y-x)^\top \nabla f(x) + \int_0^1 (y-x)^\top \{\nabla f(x + t(y-x)) - \nabla f(x)\} \mathrm{d}t.$$

Hence

$$\begin{aligned} |f(y) - f(x) - (y-x)^\top \nabla f(x)| &\leq \int_0^1 |(y-x)^\top \{\nabla f(x + t(y-x)) - \nabla f(x)\}| \mathrm{d}t \\ &\leq \int_0^1 \|(y-x)\|_2 \|\nabla f(x + t(y-x)) - \nabla f(x)\|_2 \mathrm{d}t \\ &\leq \int_0^1 L\|t(y-x)\|_2 \|y-x\|_2 \mathrm{d}t \\ &\leq \frac{L}{2}\|y-x\|_2^2. \end{aligned}$$

□

Why gradient descent works?

- Recall that since f is L–smooth

$$f(y) \leq f(x) + (y-x)^\top \nabla f(x) + \frac{L}{2}\|x-y\|_2^2.$$

- So, to minimize $f(y)$, we can sequentially try to minimize this upper bound

$$\begin{aligned}\underset{y\in\mathbb{R}^p}{\arg\min}\ f(x) + (y-x)^\top \nabla f(x) + \frac{L}{2}\|x-y\|_2^2 &\iff \underset{y\in\mathbb{R}^p}{\arg\min}\ y^\top y - 2y^\top \left\{x - \frac{1}{L}\nabla f(x)\right\} \\ &\iff \underset{y\in\mathbb{R}^p}{\arg\min}\ \left\|y - \left(x - \frac{1}{L}\nabla f(x)\right)\right\|_2^2\end{aligned}$$

- This suggests the sequence $x_{n+1} = x_n - \frac{1}{L}\nabla f(x_n)$.

Learning rate η

- Recall the gradient descent algorithm

$$x_{n+1} = x_n - \eta \nabla f(x_n), \qquad \eta > 0.$$

- In practice we don't know the constant L of our (potentially) L–smooth function f. Hence we cannot set $\eta = 1/L$.
- The learning rate η is not a parameter but rather a hyperparameter, i.e., you do not estimate it but rather calibrate / tune it!
- Convergence of the sequence is strongly impacted by η.

Learning rate η

- Recall the gradient descent algorithm

$$x_{n+1} = x_n - \eta \nabla f(x_n), \qquad \eta > 0.$$

- In practice we don't know the constant L of our (potentially) L–smooth function f. Hence we cannot set $\eta = 1/L$.
- The learning rate η is not a parameter but rather a hyperparameter, i.e., you do not estimate it but rather calibrate / tune it!
- Convergence of the sequence is strongly impacted by η.

☞ Taking η too small will lead to slow convergence, taking it too large may ruined convergence. Should compromise or take adaptive learning rate, i.e., $\eta \searrow 0$ with iterations.

A stupid (but illuminating?) example

Example 3. Consider the very challenging problem of minimizing $f(x) = x^2$, $x \in \mathbb{R}$. The gradient descent produces the sequence

$$x_{n+1} = x_n - \eta 2x_n = x_n(1 - 2\eta) = x_0(1 - 2\eta)^{n+1},$$

and the sequence converges only when $|1 - 2\eta| \leq 1$ (or $x_0 = 0$).
Note that f is 2–smooth so according to our previous results we would have set $\eta = 1/2$.

Slow convergence and momentum

$$x_{n+1} = x_n - \underbrace{\eta \nabla f(x_n)}_{\text{update step}}, \qquad \eta > 0.$$

- If for some reason the sequence $\{x_n : n \geq 1\}$ happenned to fall into a region $F_\varepsilon = \{x \in \mathbb{R}^p : \|\nabla f(x)\|_2 < \varepsilon\}$, then the sequence will move slowly. That's OK if we're near x_* but if not...

Slow convergence and momentum

$$x_{n+1} = x_n - \underbrace{\eta \nabla f(x_n)}_{\text{update step}}, \qquad \eta > 0.$$

- If for some reason the sequence $\{x_n : n \geq 1\}$ happenned to fall into a region $F_\varepsilon = \{x \in \mathbb{R}^p : \|\nabla f(x)\|_2 < \varepsilon\}$, then the sequence will move slowly. That's OK if we're near x_* but if not...
- To bypass this issue, one can use momentum method which can be thought as a child skiing down a mountain (if it helps).
- Momentum justs keeps tracks of the past gradients. More formally,

$$S_{n+1} = \gamma S_n + (1-\gamma)\nabla f(x_n)$$
$$x_{n+1} = x_n - \eta S_{n+1},$$

where $\gamma > 0$ (typical value $\gamma = 0.9$) is an additional hyper-parameter to be tuned and $S_0 = 0$.

Nesterov accelerated gradient

- If you are not experienced in skiing you will probably be overpowered with your speed. Wouldn't it be better to anticipate things?
- Nesterov's idea is related to this be defining the sequence

$$S_{n+1} = \gamma S_n + (1-\gamma)\nabla f(x_n - \gamma\delta_n)$$
$$x_{n+1} = x_n - \eta S_n,$$

where $\gamma > 0$ (typical value is again ≈ 0.9) and $S_0 = 0$.

RMSprop

- The RMSprop, for Root Mean Square Propagation, updating scheme is given by

$$S_{t+1}^2 = \gamma S_t^2 + (1-\gamma)\nabla f(x_t) \odot \nabla f(x_t)$$
$$x_{t+1} = x_t - \frac{\eta}{\sqrt{S_{t+1}^2 + \varepsilon}} \nabla f(x_t),$$

  where in the above fraction algebra is done componentwise and for some fixed value $\varepsilon > 0$ to ensure that we never divide by 0 (typical value $\varepsilon = 10^{-6}$).

Adam

- The Adam, for Adaptive Moment, is a kind of mix between momentum and RMSprop.
- The Adam updating scheme is given by

$$
\begin{aligned}
S_{t+1} &= \gamma_1 S_t + (1-\gamma_1)\nabla f(x_t) \\
S^2_{t+1} &= \gamma S^2_t + (1-\gamma)\nabla f(x_t) \odot \nabla f(x_t) \\
x_{t+1} &= x_t - \eta \frac{S_{t+1}}{\sqrt{S^2_{t+1} + \varepsilon}} \nabla f(x_t),
\end{aligned}
$$

where typical values for the hyperparameters γ_1 and γ_2, both lying in $[0, 1]$, are respectively 0.9 and 0.99 and $\varepsilon = 10^{-6}$.

CPU demanding cost function

- Suppose that computing $f(x)$ for a given $x \in \mathbb{R}^p$ is too CPU demanding
- For instance, think about minimizing the empirical risk

$$\frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, f(\theta; X_i)), \qquad \text{when } n \gg 1$$

- If we use gradient descent we will wait ages before getting the answer!
- To bypass this hurdle we can use two very simple ideas:
  - mini-batches;
  - stochastic gradient descent.
- Having said that we thus aim at minimizing f of the following form

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

Algorithm 4: Mini-batch gradient descent.

input : A differentiable function $f(x) = n^{-1} \sum_{i=1}^{n} f_i(x)$, a batch size B such that $n = B \times T$ and a initial value x_0.
output: A (hopefully) minima of f.

`/* The set` $\{1, \ldots, n\}$ `is partitioned into mini--batches, i.e.,` $\{1, \ldots, n\} = B_1 \cup \cdots \cup B_T$ `with` $B_i \cap B_j = \emptyset$ `for` $i \neq j$ `*/`

while *not converged* **do**
  for $t \leftarrow 1$ **to** T **do**
    `/* Work on the mini batch` B_t `only */`
    Compute the update step (here I use pure gradient descent)

$$\delta = \frac{1}{B} \sum_{i \in B_t} \nabla f_i(x)$$

    Update $x \leftarrow x - \eta\delta$;
Return x;

Remark. One pass over the "for" loop, i.e., T gradient descent step, is called an epoch. Other updating schemes than pure gradient descent can be used, e.g., nesterov, ADAM...

Algorithm 5: Stochastic gradient descent.

input : A differentiable function $f(x) = n^{-1} \sum_{i=1}^{n} f_i(x)$ and a initial value x_0.
output: A (hopefully) minima of f.

while *not converged* **do**
    /* Randomization stage */
    Let σ be a random permutation of $\{1, \ldots, n\}$;
    for $i \leftarrow 1$ **to** n **do**
        Compute the update step (here I use pure gradient descent)

$$\delta = \nabla f_{\sigma(i)}(x)$$

        Update $x \leftarrow x - \eta\delta$;
Return x;

Remark. As before one pass over the "for" loop is called an epoch. You can also used a mini-batch version of it leading to mini-batch stochastic gradient descent.

A note on mini-batch and stochastic gradient

- These two approaches compute the gradient on a subset only
- Typical sizes for batch are $\{62, 64, 128, 256\}$.
- Convergence is more erratic as the (sample) path of $n \mapsto f(x_n)$ is erratic and is not always non increasing.
- To ensure convergence it is necessary to use an adaptive learning rate, i.e., such that $\gamma \equiv \gamma_n \searrow 0$ as $n \to \infty$.

[MODE NEURAL NETWORK ON]

Chain rule: Multivariate case

- Once when we were kids we learnt the chain rule

$$(f \circ g(x))' = g'(x) f'(g(x)), \qquad x \in \mathbb{R}.$$

- Now that we are older, let have a look a its multivariate extension.
- Consider the case where $a(x) = \varphi(z_1(x), \ldots, z_p(x)) \in \mathbb{R}$, with $x \in \mathbb{R}^d$. We now have

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{p} \frac{\partial a}{\partial z_j} \frac{\partial z_j}{\partial x_i} = \nabla_z \varphi^\top \frac{\partial \mathbf{z}}{\partial x_i},$$

  where $\mathbf{z} = (z_1(x), \ldots, z_p(x))^\top$.
- Computing the gradient of neural nets heavily relies on this multivariate chain rule.

- To simplify, focus on the weights only and consider the case

$$\ell(\theta) = (Y - f(X;\theta))^2 \Longrightarrow \nabla\ell(\theta) = -2(Y - f(X;\theta))\nabla_\theta f(X;\theta).$$

- Let $z_L = \mathbf{W}_L^\top a_{L-1} + \mathbf{b}_L$. We have

$$\frac{\partial}{\partial \mathbf{W}_{L,i,j}} f(X;\theta) = \frac{\partial}{\partial \mathbf{W}_{L,i,j}} \varphi_L(z_L) = \varphi_L'(z_L) a_{L-1,i}.$$

- Now differentiate w.r.t. to the previous layer, we have

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} f(X;\theta) &= \frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} \varphi_L(z_L) \\
&= \varphi_L'(z_L) \frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} z_L, z_L = \mathbf{W}_L^\top \varphi_{L-1}(z_{L-1}) + \mathbf{b}_L \\
&= \varphi_L'(z_L) \mathbf{W}_L^\top \frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} \varphi_{L-1}(z_{L-1}) \\
&= \varphi_L'(z_L) \mathbf{W}_L^\top \varphi_{L-1}'(z_{L-1}) a_{L-1,i}
\end{aligned}$$

Keep on browsing the neural net we have

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} f(X;\theta) &= \frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} \varphi_L(a_L) \\
&= \varphi_L'(a_L) \frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} a_L, z_L = \varphi_{L-1}(\mathbf{W}_{L-1}^\top z_{L-1} + \mathbf{b}_{L-1}) \\
&= \varphi_L'(a_L) \mathbf{W}_L^\top \frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} \varphi_{L-1}(a_{L-1}) \\
&= \varphi_L'(a_L) \mathbf{W}_L^\top \varphi_{L-1}'(a_{L-1}) \frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} a_{L-1} \\
&= \varphi_L'(a_L) \mathbf{W}_L^\top \varphi_{L-1}'(a_{L-1}) \mathbf{W}_{L-1}^\top \varphi_{L-2}'(a_{L-2}) z_{L-2,i}
\end{aligned}$$

To compare with

$$\frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} f(X;\theta) = \varphi_L'(a_L) \mathbf{W}_L^\top \varphi_{L-1}'(a_{L-1}) z_{L-1,i}$$

- We thus have

$$\frac{\partial}{\partial \mathbf{W}_{L,i,j}} f(X;\theta) = \varphi'_L(a_L) z_{L,i}$$
$$\frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} f(X;\theta) = \varphi'_L(a_L) \mathbf{W}_L^\top \varphi'_{L-1}(a_{L-1}) z_{L-1,i}$$
$$\frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} f(X;\theta) = \varphi'_L(a_L) \mathbf{W}_L^\top \varphi'_{L-1}(a_{L-1}) \mathbf{W}_{L-1}^\top \varphi'_{L-2}(a_{L-2}) z_{L-2,i}$$

- Now if we let $\delta_L = \varphi'_L(a_L)$ and $\delta_\ell = \mathbf{W}_{\ell+1}^\top \delta_{\ell+1} \varphi'_\ell(a_\ell)$, we then have

$$\frac{\partial}{\partial \mathbf{W}_{L,i,j}} f(X;\theta) = \delta_L z_{L,i}$$
$$\frac{\partial}{\partial \mathbf{W}_{L-1,i,j}} f(X;\theta) = \delta_{L-1} z_{L-1,i}$$
$$\frac{\partial}{\partial \mathbf{W}_{L-2,i,j}} f(X;\theta) = \delta_{L-2} z_{L-2,i}$$

Backpropagation

- Backpropagation = nice way to compute the gradient for neural nets
- backpropagation because network is browsed backward

Algorithm 6: A generic backpropagation algorithm.

input : A neural net, a cost function $C(\theta)$ and a parameter value $\theta = \{(\mathbf{W}_\ell, \mathbf{b}_\ell) : \ell = 1, \ldots, L\}$.
output: The gradient of the cost function at θ.

Compute the activation a_1 for the input layer;
for $\ell \leftarrow 2$ **to** L **do**
    /* This is the forward pass */
    Compute $z_\ell = \mathbf{W}_\ell^\top a_{\ell-1} + \mathbf{b}_\ell$;
    Compute $a_\ell = \varphi_\ell(z_\ell)$;
Compute the output error $\delta_L = \nabla_a C(\theta) \odot \varphi_L'(z_L)$;
for $\ell \leftarrow L - 1$ **to** 2 **do**
    /* This is the backward pass */
    Compute layer error $\delta_\ell = \mathbf{W}_{\ell+1}^\top \delta_{\ell+1} \odot \varphi_\ell'(z_\ell)$;
The gradient of the cost function is then:

$$\frac{\partial}{\partial \mathbf{W}_{\ell,ij}} C(\theta) = a_{\ell-1,k}\delta_{\ell,j}, \qquad \frac{\partial}{\partial \mathbf{b}_{\ell,i}} C(\theta) = \delta_{\ell,i}$$

Algorithm 7: Pseudo code for fitting a neural network.

input : An architecture of a neural net, a cost function to be minimized $C(\theta)$ and a parameter value $\theta_0 = \{(\mathbf{W}_\ell, \mathbf{b}_\ell) : \ell = 1, \ldots, L\}$.

output: A (hopefully) local minima of the cost function.

while *convergence is not met* **do**
    Using backpropagation compute the gradient $\nabla C(\theta)$;
    Update θ using your favourite updating scheme, e.g., RMSProp, Adam, ...;
Return θ;

Algorithm 7: Pseudo code for fitting a neural network.

input : An architecture of a neural net, a cost function to be minimized $C(\theta)$ and a parameter value $\theta_0 = \{(\mathbf{W}_\ell, \mathbf{b}_\ell) : \ell = 1, \ldots, L\}$.

output: A (hopefully) local minima of the cost function.

while *convergence is not met* **do**
    Using backpropagation compute the gradient $\nabla C(\theta)$;
    Update θ using your favourite updating scheme, e.g., RMSProp, Adam, ...;
Return θ;

☞ Of course you can use stochastic gradient and mini-batch as well!

Why backpropagation changed the deal?

- Just because from a single forward pass followed by a backward one, we can compute all partial derivatives!
- Just compare that to using the naive finite difference approach, e.g.,

$$\frac{C(\theta + he_j) - C(\theta)}{h}, \qquad j = 1, \ldots, n_p,$$

  where n_p denotes the number of parameters of the neural network.
- If you were to do so, then you will have to evaluate $n_p + 1$ times the cost function. Remember current values for $n_p > 10^6$.
- This is why neural nets became popular (again) in the late 80's.
- However for deep neural net backpropagation alone is not enough...
- We will now see more recent techniques that enable the use of deep network.

Regularization

- State of the art deep neural networks have millions of parameters
- Clearly this would lead to overfitting
- Several strategies have been proposed to avoid overfitting:
  - Early stopping
  - Penalization
  - Dropout
- We will cover these options in turn.

Early stopping

- The idea behind early stopping is very simple. During the optimization stage, if updating the parameter θ does not decrease the cost function $C(\theta)$ computed on the validation/test data set, stop optimization.
- In practice you may terminate the optimization stage when you see no improvements after T iterations, e.g., $T \in \{10, 50\}$.

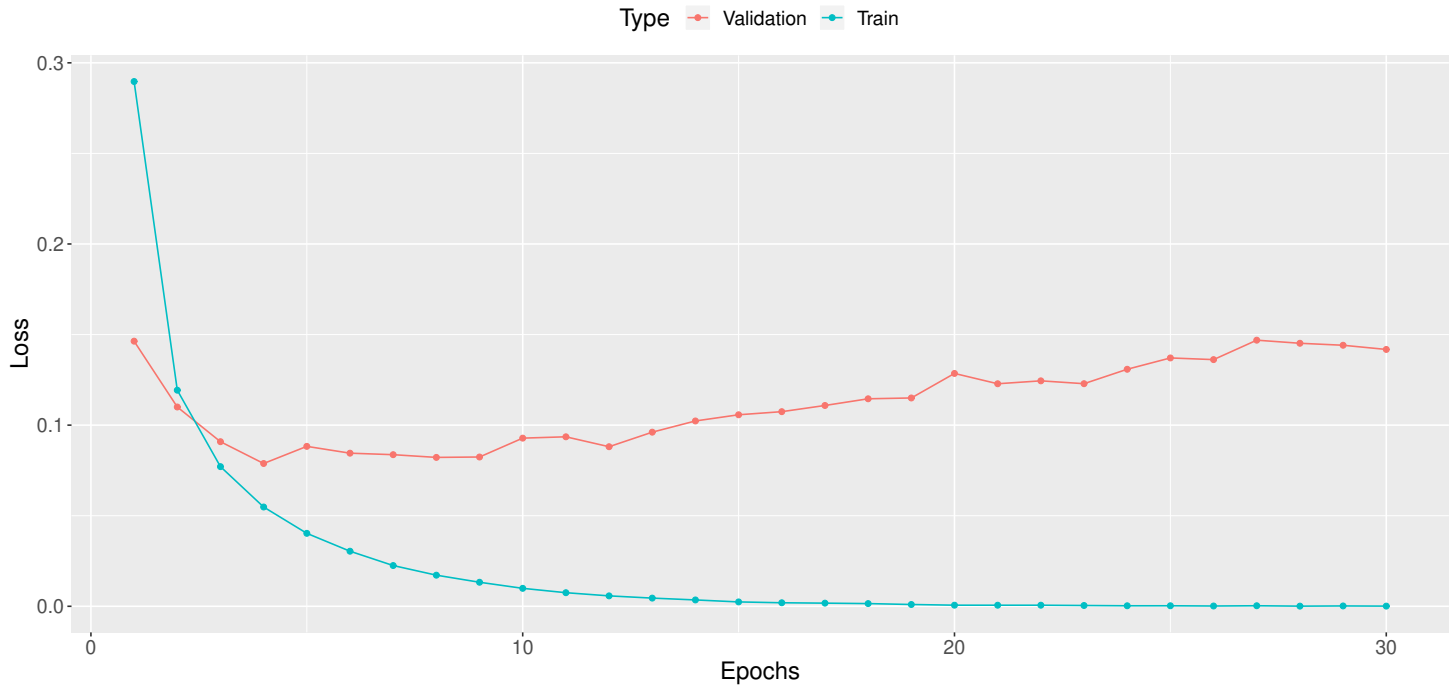


Figure 26: *Illustration of the early stopping regularization. Here we might want to stop after 5–10 epochs.*

Penalization

- In this lecture we already talked about regularization techniques, e.g., lasso and ridge regression.
- Here we just do the same and consider a new optimization problem

$$\arg\min_{\theta \in \Theta} C(\theta) + \lambda \text{Penalty}(\theta),$$

where typically

$$\text{Penalty}(\theta) = \begin{cases} \sum_{\ell=1}^{L} \sum_{i,j} \mathbf{W}_{\ell,i,j}^2, & \text{if } \ell_2 \text{ regularization} \\ \sum_{\ell=1}^{L} \sum_{i,j} |\mathbf{W}_{ell,i,j}|, & \text{if } \ell_1 \text{ regularization} \end{cases}$$

Penalization

- In this lecture we already talked about regularization techniques, e.g., lasso and ridge regression.
- Here we just do the same and consider a new optimization problem

$$\arg\min_{\theta \in \Theta} C(\theta) + \lambda \text{Penalty}(\theta),$$

where typically

$$\text{Penalty}(\theta) = \begin{cases} \sum_{\ell=1}^{L} \sum_{i,j} \mathbf{W}_{\ell,i,j}^2, & \text{if } \ell_2 \text{ regularization} \\ \sum_{\ell=1}^{L} \sum_{i,j} |\mathbf{W}_{ell,i,j}|, & \text{if } \ell_1 \text{ regularization} \end{cases}$$

☞ Note that only the weights are penalized!

Dropout (Stochastic regularization)

- Dropouting a neural net consists in sampling a thinned network from it.
- More formally with a dropout net, at each forward pass during the training stage of the net, we have, for $\ell \in \{2, \ldots, L-1\}$,

$$
\begin{aligned}
D_{\ell-1,i} &\overset{\text{iid}}{\sim} \text{Bernoulli}(p), \qquad i = 1, \ldots, s_{\ell-1} \\
\tilde{a}_{\ell-1} &= a_{\ell-1} \odot D_{\ell-1} \\
a_\ell &= \varphi(\mathbf{W}_\ell^\top \tilde{a}_{\ell-1} + \mathbf{b}_\ell).
\end{aligned}
$$

- However for prediction, e.g., forward pass on the test set, we have, for $\ell \in \{2, \ldots, L-1\}$,

$$
\check{a}_{\ell-1} = p a_{\ell-1}, \qquad a_\ell = \varphi(\mathbf{W}_\ell^\top \check{a}_{\ell-1} + \mathbf{b}_\ell),
$$

to ensure that $\mathbb{E}(\tilde{a}_{\ell-1}) = \check{a}_{\ell-1}$.

Dropout in picture (1)

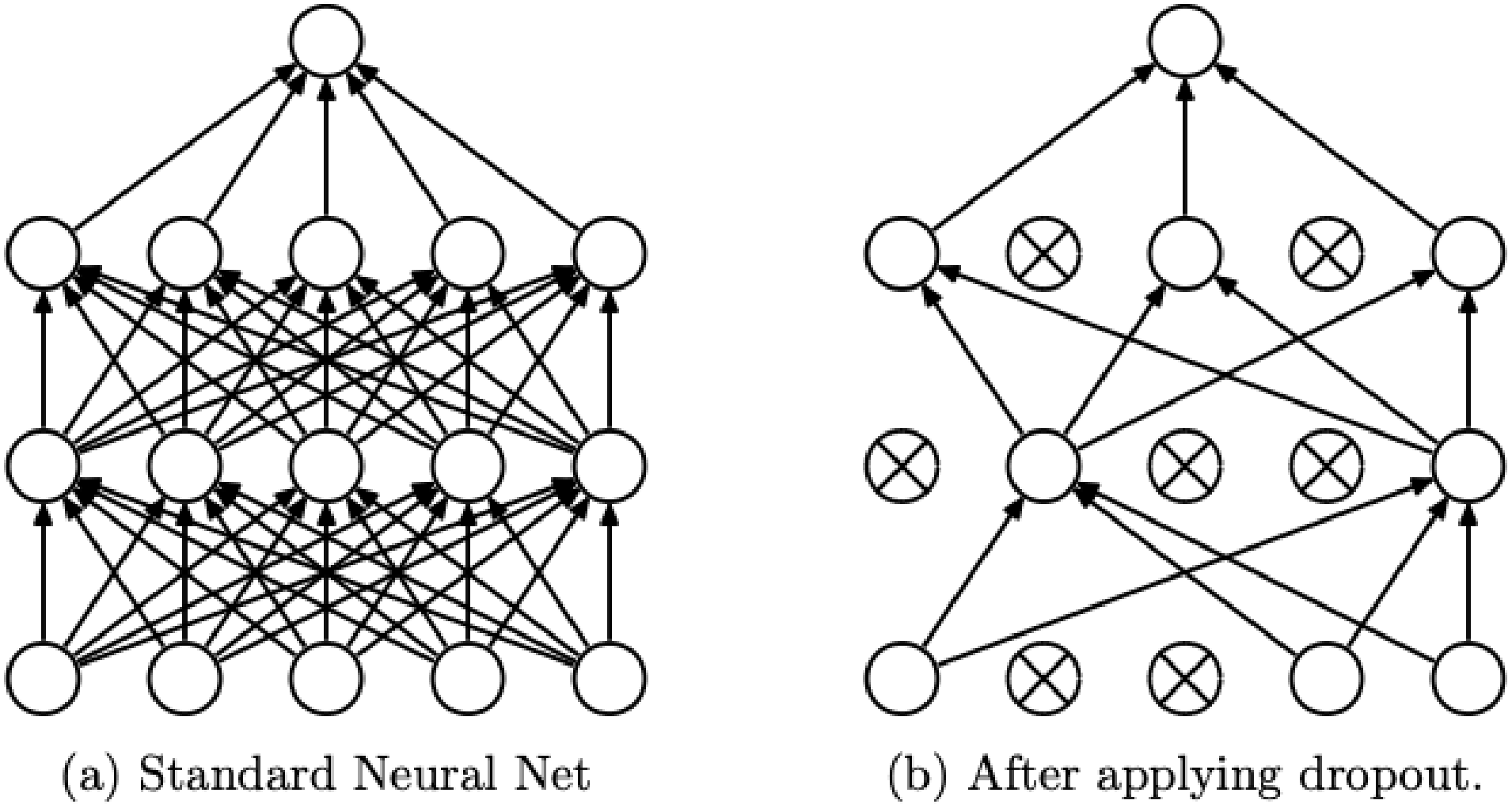


Figure 27: *Dropout neural net. Left: A standard neural net with 2 hidden layers. Right: A realization of a thinned net from the left one. Crossed units have been dropped. Figure taken from Srivastava et al. (2014).*

Dropout in picture (2)

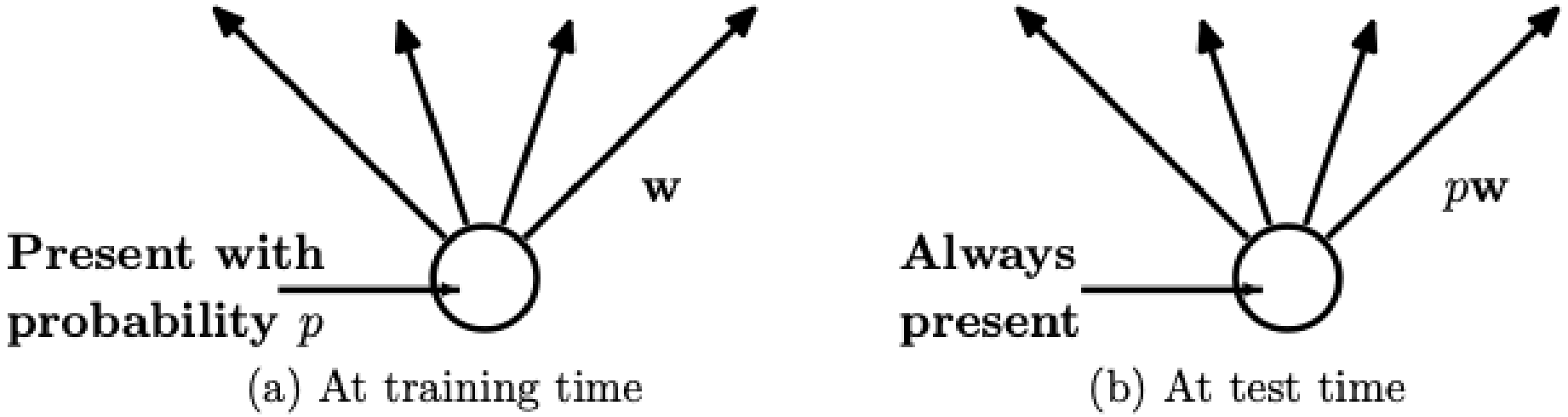
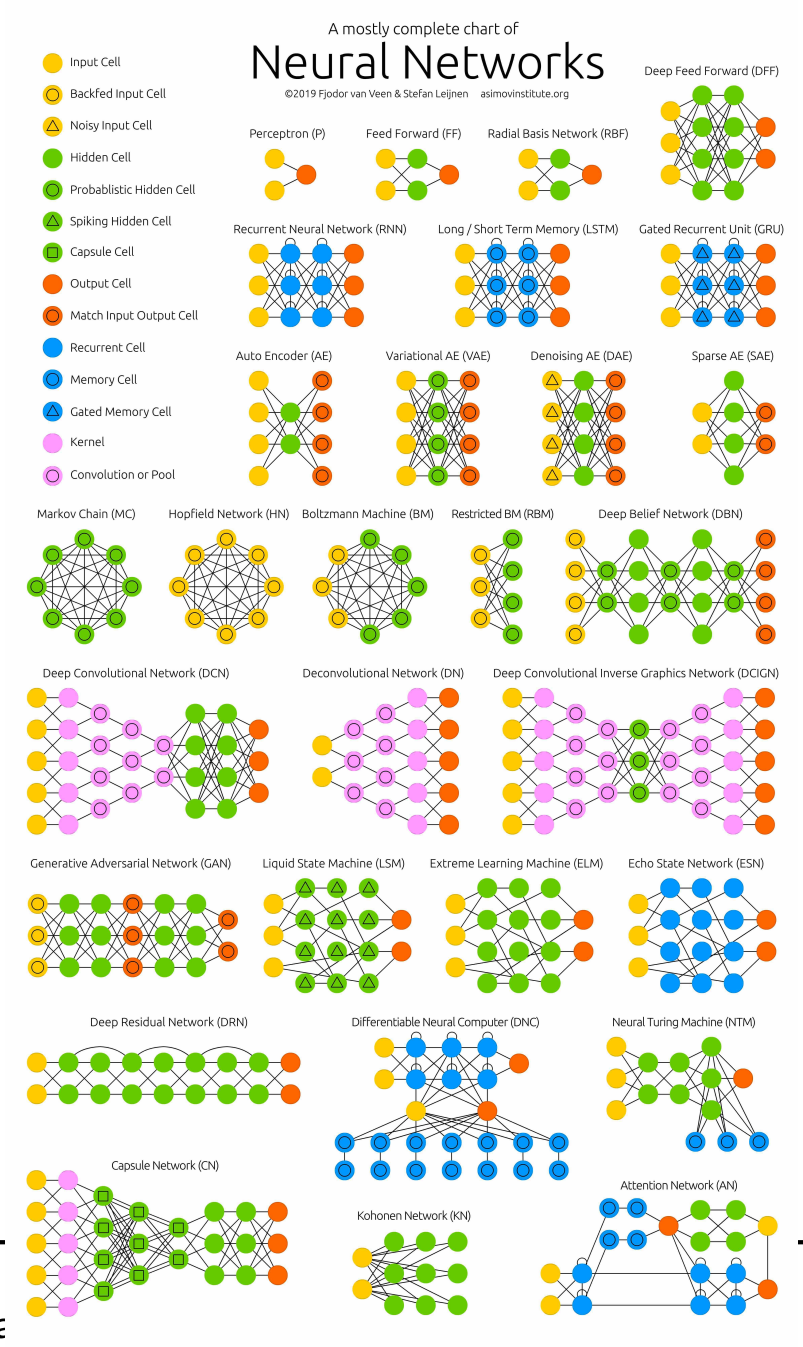


Figure 28: *Left: Unit at training time that is present with probability p and is connected to units in the next layer with weight $\mathbf{w}$. Right: At test time, the unit is always present and the weights are multiplied by p. The output at test time is the same as the expected output at training time. Figure taken from Srivastava et al. (2014).*

Neural network zoology



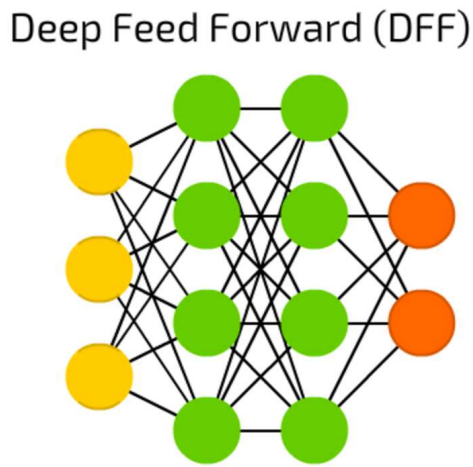
- Don't panic, we won't cover them all.
- Just focus on 2 of them
- But you can do your homework and investigate the other ones ;-)

(Deep) Feedforward neural network

A feedforward neural network is:

- fully connected, i.e., all nodes from layer ℓ are connected to that of layer $\ell + 1$;
- There is no backloops, i.e., flow of information goes straight from input to output.
- It is deep is you have more than 1 hidden layer.

It is a kind of baseline neural net model.

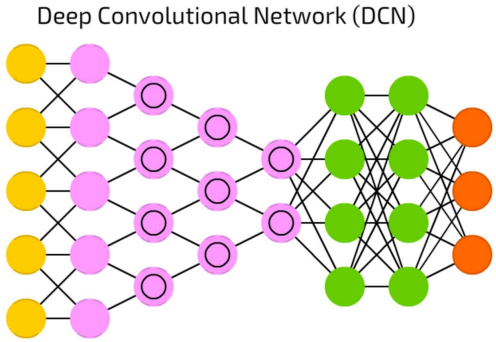


(Deep) Convolutional neural network

A convolutional neural network makes use of:

- convolution rather than matrix multiplication (in at least one layer);
- a pooling stage.

It is mostly used for image recognition.



Convolution

- A convolutional layer, say the ℓ–th layer, compute pre-activation values as follows

$$z_{\ell,i} = \sum_{j \in i + W(K)} K(j) a_{\ell-1,j}, \qquad i = 1, \ldots, s_\ell$$

  where $W(K)$ is the user defined kernel window and $K(j)$ are kernel weights that must be estimated.
- An important point is that there are only $|W(K)|$ parameters for this layer
- This is known as parameter sharing

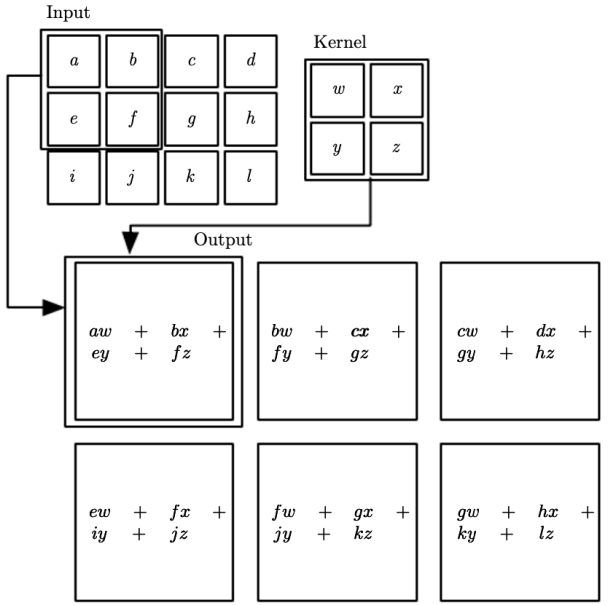


Figure 29: *An example of 2D convolution. Taken from Goodfellow et al. (2017).*

Remark. After this convolution stage we apply, as usual, an activation function φ_ℓ.

Pooling

- The last step in a convolutional layer is pooling.
- More formally the activation values a_ℓ are transformed using a mapping of the form

$$\tilde{a}_{\ell,j} = p(a_{\ell,i} : i \in \mathcal{N}(j)), \qquad j = 1, \ldots, s_\ell,$$

  where p is the pooling operator and $\mathcal{N}(j)$ a user specified neighbourhood of j.
- Typical pooling operators are maximum or (weighted) mean.

Remark. The aim of pooling the layer ℓ is to make (approximately) its output invariant to translation. Such invariance may be desirable if the goal is to detect if "something is present" rather than where this thing is.

Approximate invariance of max–pooling

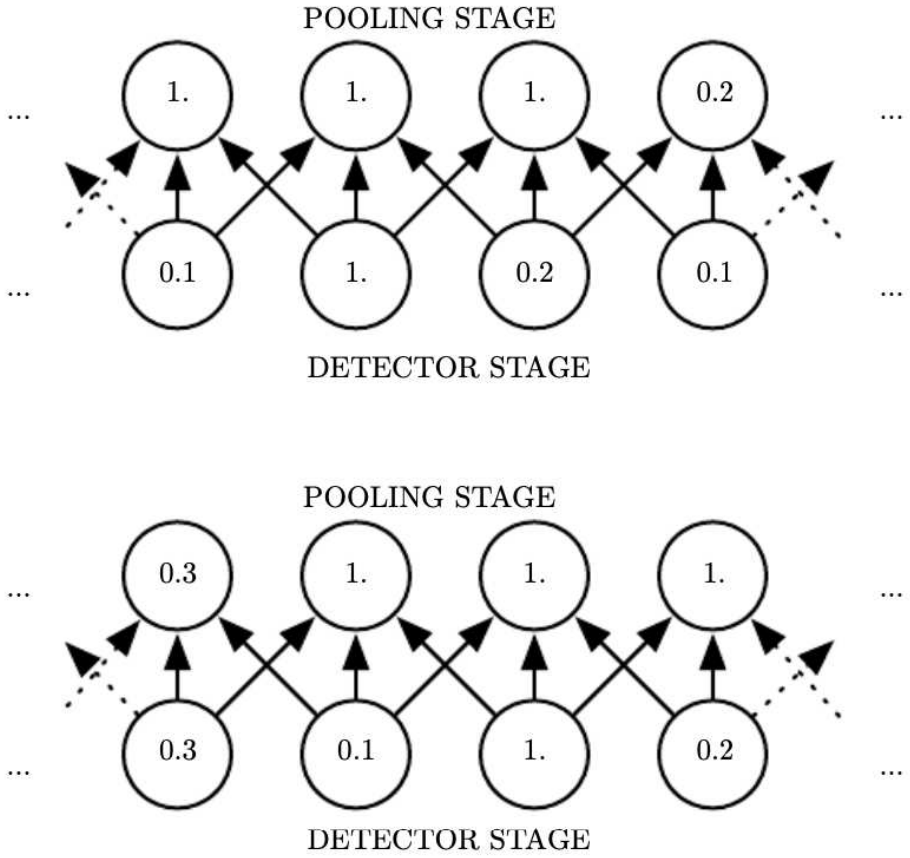


Figure 30: *Max–pooling introduces invariance. Top: A view of the middle of the output of a convolutional layer. The bottom row shows the output of the activation function. The top rows shows the output of max–pooling (pooling width: 3 pixels). Bottom: A view of the same network, after the input has been shifted to the right by 1 pixel. Although all values in the bottom row have change, only half of the values in the top row have changed. Taken from Goodfellow et al. (2017).*

Neural network in practice

- There are a few efficient framework to fit neural network, e.g.,
  - TensorFlow
  - CNTK
  - Theano
- Here we will use Keras (within R) which is capable of running on top of the aforementioned frameworks.

LET'S MOVE TO THE LAB!