NEUROINFORMATICS

LECTURE SCRIPT

WT 2015/16
Prof. Dr. Gordon Pipa
University of Osnabrück

Foreword

Welcome to Neuroinformatics!

This script will support you throughout the Neuroinformatics lecture in the winter term. It is rather new and currently not complete. It will however be completed over the course of the semester. If you find any errors, just send an e-mail to niscript@ni.uos.de. Also, if you find anything misleading or completely incomprehensible, send us an e-mail and we will try and solve it. We can't guarantee that the script is complete in terms of what will be expected in the exam. In case you find any major information left out in the script, better keep with the slides for that, but feel free to send us a note via e-mail.

Cheers,
the Neuroinformatics Team

Contributors:
Prof. Dr. Gordon Pipa
Leon Sütfeld

Contents

1 Recommended Literature **4**

2 Introduction **7**
2.1 Why do we need statistical modeling? 7
2.2 Example: Model fitting 7
2.3 How is the model constructed? 7
2.4 Example: Linear combinations of polynomials 8
2.5 Outlook: What is it useful for? 8

3 Probability theory and related topics **10**
3.1 Marginal probability 10
3.2 Conditional probability 11
3.3 Joint probability 11
3.4 Sum and product rule 11
3.5 Bayes' theorem 12
3.6 Summary: Probability theory and Bayes 13
3.7 Continuous random variables probability density functions 13
3.8 Transformation of densities 15
3.8.1 Example: Transformation of densities 16
3.9 Expectations 17

4 Graphical models **19**
4.1 Bayesian networks 19
4.2 Statistical independence of variables 21
4.2.1 Conditional independence of variables 21
4.3 D-separation: Statistical independence in graphs . . . 21
4.3.1 Head-to-tail nodes 21
4.3.2 Fixed head-to-tail nodes 23
4.4 Head-to-head nodes 23
4.5 Fixed head-to-head nodes 24
4.6 Tail-to-tail nodes 24
4.6.1 Fixed tail-to-tail nodes 25
4.7 Overview & more 25

5 Probability distributions **27**
5.1 Discrete distributions 27
5.1.1 Bernoulli distribution 27
5.1.2 Binomial distribution 28
5.1.3 Poisson distribution 28
5.2 Continuous distributions 29
5.2.1 Normal distribution 29
5.2.2 Gamma distribution 31
5.2.3 Beta distribution 32
5.2.4 χ^2 distribution 33
5.3 Summary 35

6 Model Fitting via Maximum Likelihood **36**
6.1 The Maximum Likelihood Approach - Univariate Data Sets 36
6.1.1 Rationale of The ML Approach 37
6.1.2 Setting up the likelihood function (for a single data point) 37

6.1.3 Joint Likelihood (multiple data points) 38
6.1.4 Deriving the ML Estimators 38
6.1.5 Normal Distributions: Deriving μ_{ML} 38
6.1.6 Normal Distributions: σ^2_{ML} and bias 40
6.1.7 Overview 41
6.2 Bivariate data: The linear regression 42
6.2.1 Transition from univariate data 42
6.2.2 The underlying model 43
6.2.3 Setting up our model 43
6.2.4 Setting up the likelihood function (bivariate) . 44
6.2.5 Maximizing the likelihood: The method of least squares 44
6.2.6 The Moore-Penrose-pseudoinverse 46
6.3 Examples of model fitting (bivariate) 46
6.3.1 The pen and paper method 46
6.3.2 Fitting a model to data using Matlab 48

7 Basis functions 51
7.1 Polynomial basis functions 51
7.2 Gaussian basis functions 51
7.3 Sigmoidal basis functions 52
7.4 Sinusoid basis functions 52
7.5 Bin-based basis functions 54
7.5.1 Continuity in piecewise polynomials 54
7.5.2 B-splines 55

8 Generalizability, validation and model selection 57
8.1 Training- and validation data 57
8.2 Number of basis functions, over- and underfitting . . . 57
8.3 Cross-validation 58

9 Model selection: Nested models, likelihood ratio and deviance 61
9.1 Nested and saturated models 61
9.2 Likelihood ratios 61
9.3 Saturated models 63

10 Bias-variance-decomposition 64

11 Regularization 66

12 Bayesian regression 70
12.1 Conjugate prior and MAP estimates for Bernoulli . . . 71
12.2 Sequential updates to the posterior/ prior 72
12.3 Choosing the right prior 73
12.4 Bayesian regression and regularization 73

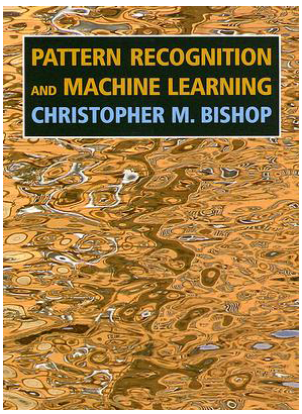
13 Model selection: Akaike Information Criterion 75
13.1 The Kullback-Leibler divergernce 75
13.2 AIC: A bias-corrected approximation of the KL divergence 77

14 Model selection: Bootstrap and EIC 78

1 Recommended Literature

Attending a lecture and using a script is never enough for learning a topic. You will find reading a book very helpful, since it provides additional information and supplemental explanations, and it will allow you to go your own speed. However, time isn't unlimited, so pick one or two.

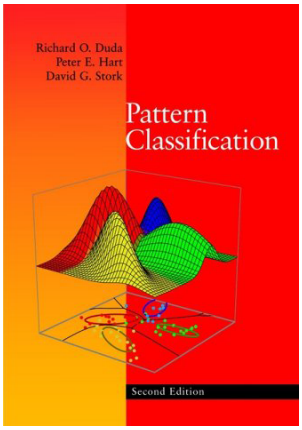
1. **Pattern Recognition and Machine Learning**
   by Christopher M. Bishop, Publisher: Springer
   http://research.microsoft.com/en-us/um/people/cmbishop/prml/



Recommendation: ++

This is the main book for this lecture. It's a good book, from easy to advanced. You can get some chapters online for free, make use of that!

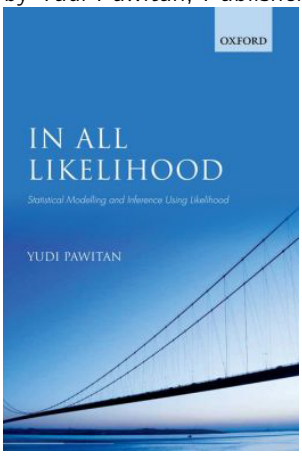
2. **Pattern Classification (2nd Edition)**
   (Student Solutions Manual also available)
   by Duda, Hart and Stork, Publisher: Wiley



Recommendation: +

This is an alternative to Bishop with emphasis on classification. From easy to advanced.

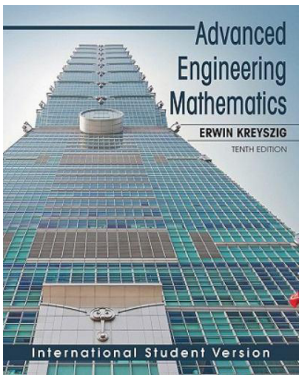
3. **In All Likelihood Statistical Modelling and Inference Using Likelihood**
by Yudi Pawitan, Publisher: Oxford University Press



Recommendation: O

Rather advanced level. Supplemental material for interested students.

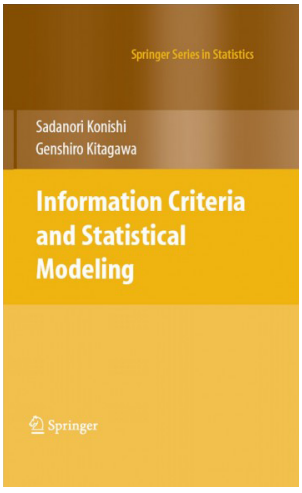
4. **Advanced Engineering Mathematics: International Student Version**
by Erwin Kreyszig



Recommendation: +

Great Math book. Gives examples and combines these with rigorous proofs.

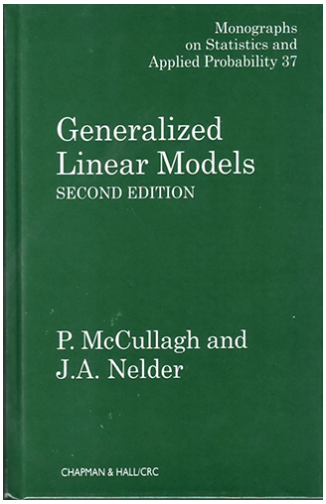
5. **Information Criteria and Statistical Modeling**
by Konishi and Kitagawa, Publisher: Springer



Recommendation: +

Advanced book, great book for people who want to know all about model selection. Also covers model selection teaching material.

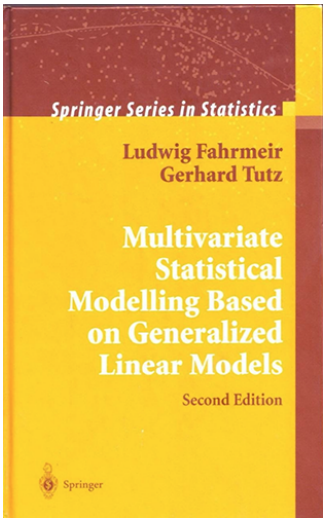
6. **Generalized Linear Models, Second Edition**
by McCullagh and Nelder, Publisher: CRC Press Inc.



Recommendation: O

Supplemental. This is an advanced book, great for people who want to know all about generalized linear models.

7. **Multivariate Statistical Modelling Based on Generalized Linear Models**
by Fahrmeir and Tutz, Publisher: Springer



Recommendation: O

Supplemental. This is an advanced book, great for people who want to know all about generalized linear models.

2 Introduction

2.1 Why do we need statistical modeling?

When collecting data in the real world - e.g. recording the spiking of a neuron - the data recorded is usually subject to random fluctuations (noise). Often the noise is intense enough to nearly completely hide the true structure of the data. We use statistical modeling to reconstruct this underlying structure from noisy data.

2.2 Example: Model fitting

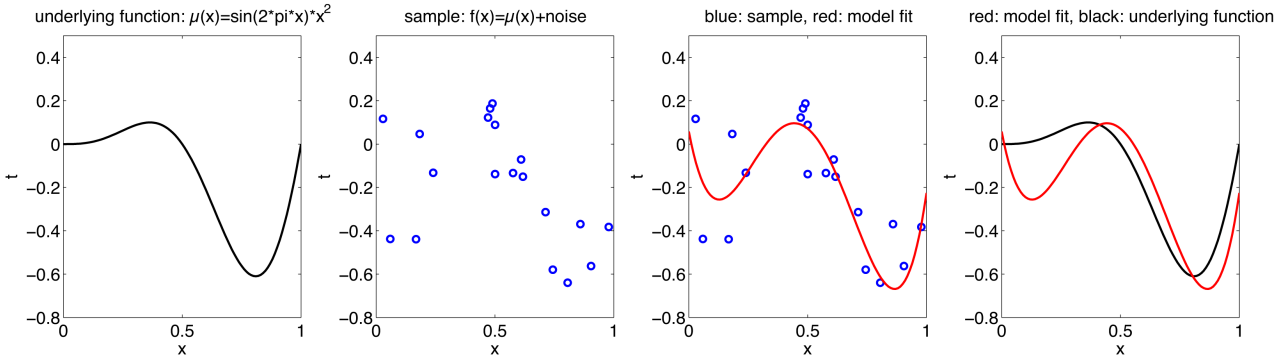


Figure: An example of statistical modeling. In the first plot, we see the original function (also called underlying function or underlying model), which represents some course of a variable. For instance, this could be voltage over time, maybe a neuron's membrane potential, but also EEG- or behavioral data. The underlying function is unknown to us in the real world. Here, all we ever have is measured data, which is subject to noise, i.e. the observations are a superposition of the original (or underlying) function and noise (second plot). In order to uncover the structure of the underlying function, we fit a flexible function to the observed data, the result is what we call our model (third plot, red line). You can see a comparison of the estimated model (red) with the underlying model (black) in the fourth plot. Bear in mind that this comparison can't be done in the real world, since we never know the exact underlying function (black).

For a better understanding, the Matlab code behind many of the figures in this script is provided with the script. Often, an algorithmic approach is easier to understand than an analytic one. Thus, if there is anything you don't get, and there's a related figure, reading the code and playing around with it is highly recommended. We aimed to keep the code simple and structured, so basic programming skills should enable you to understand the code in most cases. The Matlab file for this figure is provided in *ch2fig1.m*. In this case, you probably won't understand what the code is doing just yet. However, you might want to come back to it later.

2.3 How is the model constructed?

Our estimated model (e.g. the red curve above) is a weighted linear combination of functions - so called basis functions - and consequently a function itself. The basis functions within a model usually stem from the same family of functions, e.g. polynomials, gaussians or cubic splines. In the first part of the lecture (and the script), we will mostly deal with polynomials as basis functions. Accordingly, our n basis functions are $x^0, x^1, x^2, ..., x^n$. In principle, we are free to choose the number of basis functions ourselves. The corresponding weights are called $w_0, w_1, w_2, ..., w_n$. Thus, our polynomial model is defined as

$$y = w_0 + w_1 x + w_2 x^2 + ... + w_n x^n$$

or in a different notation

$$y(\vec{w}, x) = \sum_i (w_i \cdot x^i).$$

The weights w_i are flexible - solely by changing the weights, few basis functions can be arranged to extremely diverse curves. To determine the weights such that they optimally fit our data is what we call model fitting. The resulting function is our estimated model.

2.4 Example: Linear combinations of polynomials

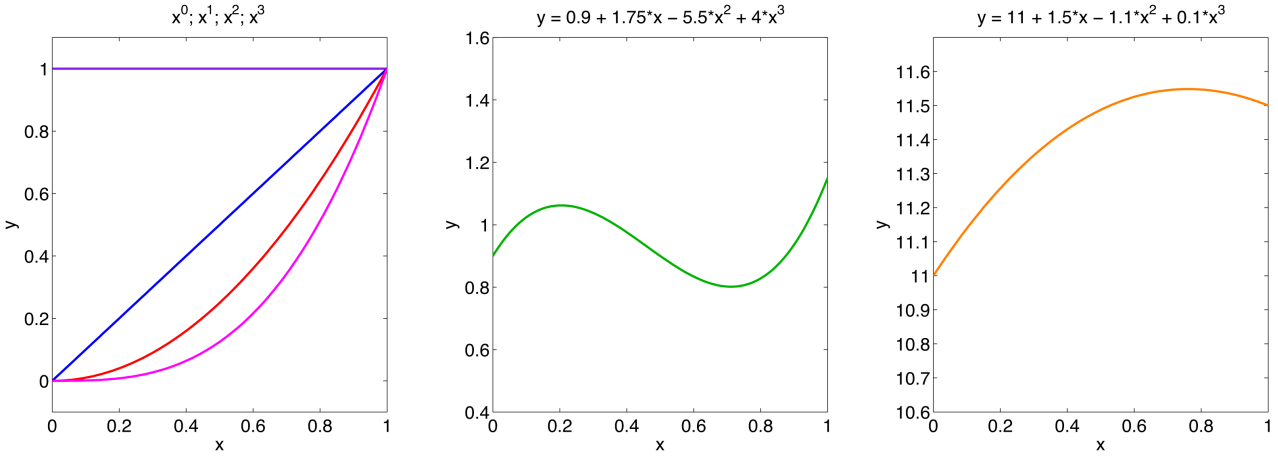


Figure: Different curves resulting from the same basis functions. The left-hand figure shows our basis functions for a 3rd order polynomial model, i.e. x^0, x^1, x^2 and x^3 in the range zero to one. The figures in the middle and on the right show two different linear combinations of these four basis functions.

The code for this figure is provided in *ch2fig2.m*. It doesn't do much, but you could play around with the weights of the polynomials just to get a feeling for it.

middle figure:

$w_0 = 0.9$
$w_1 = 1.75$
$w_2 = -5.5$
$w_3 = 4.0$
thus the model is $y = 0.9 + 1.75 \cdot x - 5.5 \cdot x^2 + 4 \cdot x^3$

right-hand figure:

$w_0 = 11.0$
$w_1 = 1.5$
$w_2 = -1.1$
$w_3 = 0.1$
thus the model is $y = 11 + 1.5 \cdot x - 1.1 \cdot x^2 + 0.1 \cdot x^3$

2.5 Outlook: What is it useful for?

The methods taught in this course, at the heart of which is the model fitting process, have a wide range of applications in the real world. Statistical modeling processes are, for instance, used in weather and climate models. More closely related to cognitive science is, for instance, the analysis of data from various psychophysical experiments. Sensory processes often cause stable patterns/ characteristics in the data, resulting from an underlying function overlain by noise. With statistical modeling, we can infer functions describing these patterns, and thus gain insight in the brain's computational principles.
Another important application for statistical modeling is the decoding

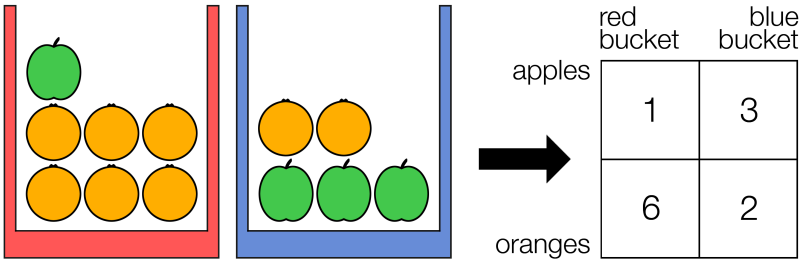
of neuronal activity. By modeling the spiking probabilities of neurons under different conditions, we are, for instance, able to classify from which of the conditions new trials have arisen. In other words: We can tell, which of two movie sequences a monkey was watching just by analyzing the output of single cortical cells.

A block course "Decoding Neuronal Activity" will be offered in the semester break.

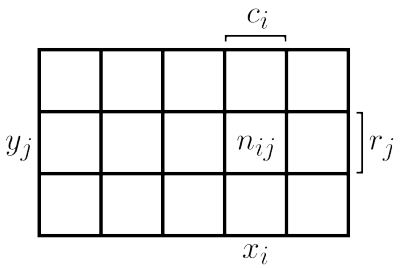
3 Probability theory and related topics

Before we get to the actual model fitting process, there are some important prerequisites to consider. First, we take a look at the basics of probability theory.
To dumb things down, let's talk about apples and oranges in colored buckets. In this case, the type of fruit is one class, the buckets are another class. Since there are two types of fruits and two buckets, this can be translated into a 2x2 grid.

Those of you familiar with probability theory and Bayes' theorem can skip this section up to the summary.



In mathematical terms, the type of fruit is labeled as y_j and the type of bucket as x_i. The set of all fruits in one bucket is $c_i = \sum_j n_{ij}$, the set of all fruits of one type is $r_j = \sum_i n_{ij}$. Finally, one type of fruit in a specific bucket is labeled n_{ij} and the set of all fruits is $N = \sum_{i,j} n_{ij}$.

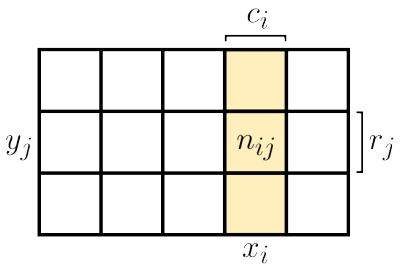


With these terms at hand we can express different statements about probabilities in formulas.

3.1 Marginal probability

The marginal probability is defined as

$$P(X = x_i) = c_i/N$$



X is a random variable, $P(X = x_i)$ is the probability that the random variable X takes the value x_i, irrespective of class y.
If you're still thinking fruits in buckets, the marginal probability is the probability that you draw a fruit from bucket x_i, irrespective of the type

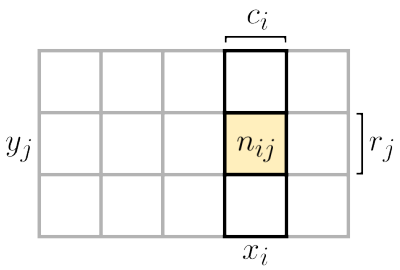
of fruit. For example, the chance that a fruit you draw is from the blue bucket is

$$p(X = \text{blue bucket}) = 5/12.$$

3.2 Conditional probability

The conditional probability is defined as

$$P(Y = y_j | X = x_i) = n_{ij}/c_i.$$



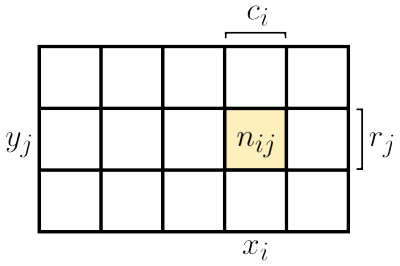
In fruit-terms, this translates to the chance of drawing a specific type of fruit y_j when reaching into a specific bucket x_i. For example, the chance of drawing an apple from the red bucket is

$$P(Y = \text{apple} | X = \text{red bucket}) = 1/7.$$

3.3 Joint probability

The joint probability is defined as

$$P(Y = y_j, X = x_i) = n_{ij}/N.$$



Fruits: The chance of any fruit being a specific type of fruit y_j, while at the same time coming from a specific bucket x_i. For example, the chance that a fruit is an orange and comes from the blue bucket is

$$P(Y = \text{orange}, X = \text{blue bucket}) = 2/12.$$

3.4 Sum and product rule

The *sum rule* goes as follows:

$$p(X) = \sum_Y p(X, Y)$$

The process of taking the sum over a variable from a joint distribution in order to kick it out of the equation is also called *marginalization*. If X is the color of the bucket and Y is the type of fruit, marginalizing over Y corresponds to no longer differentiating between different types of fruit.

In other words, the marginal probability for a specific class is the sum of all joint probabilities within this class. Looking at the grid, this means adding up the three cells of a column. For the fruits this translates to dumping the contents of both buckets into one big bucket:

$$p(Y = \text{orange}) = \sum_X p(Y = \text{orange}, X) = \frac{6}{12} + \frac{2}{12} = \frac{8}{12} = \frac{2}{3}$$

The *product rule*:

$$p(X, Y) = p(X|Y)p(Y)$$

...and since $p(X, Y) = p(Y, X)$, the following equation also holds: $p(Y|X)p(X) = p(X|Y)p(Y)$

Translated, *joint probability = conditional probability · marginal probability*. In the grid this would be picking the column, then picking a cell from this specific column.

In fruit terms: The chance for any of the fruits to be an apple from the blue bucket can be translated to the chance of being an apple in the blue bucket (conditional probability) times the chance for any fruit to be from the blue bucket (marginal probability):

$$p(Y = \text{apple}, X = \text{blue}) = p(Y = \text{apple}|X = \text{blue})p(X = \text{blue})$$

$$= \frac{3}{5} \cdot \frac{5}{12} = \frac{3}{12} = \frac{1}{4}$$

Substituting the sum rule's summand according to the product rule will give you the following equation, called the *law of total probability*:

$$p(X) = \sum_Y p(X|Y)p(Y)$$

3.5 Bayes' theorem

Bayes' theorem is now easily derived. Since $p(X, Y) = p(Y, X)$, we can substitute both sides according to the product rule, resulting in $p(Y|X)p(X) = p(X|Y)p(Y)$. Dividing both sides by $p(X)$ now gives us

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}.$$

This is *Bayes' theorem*. When using Bayes' theorem, $p(Y|X)$, $p(X|Y)$ and $p(Y)$ are typically referred to as *posterior*, *likelihood* and *prior*. The posterior is proportional to likelihood times prior:

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

This particular equation will become important later in the course, so make sure to familiarize yourselves with probability theory and Bayes' theorem.

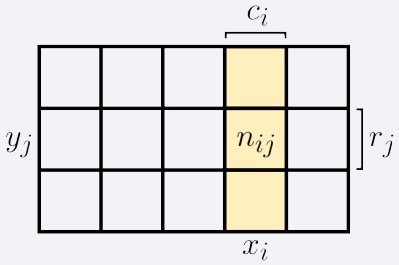
3.6 Summary: Probability theory and Bayes

marginal probability

$P(X)$

Probability that $X = x_i$ occurs.

$P(X = x_i) = c_i/N$

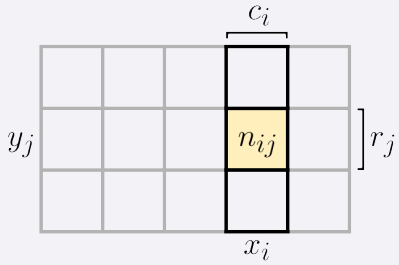


conditional probability

$P(Y|X)$

Probability that $Y = y_j$ occurs, when $X = x_i$ has already occured.

$P(Y = y_j|X = x_i) = n_{ij}/c_i$

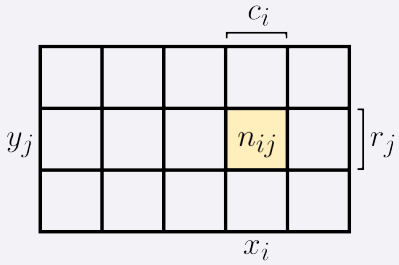


joint probability

$P(X, Y)$

Probability that $Y = y_j$ and $X = x_i$ occur at the same time.

$P(X = x_i|Y = y_j) = n_{ij}/N$



sum rule

$$p(X) = \sum_Y p(X, Y)$$

product rule

$$p(X, Y) = p(Y|X)p(X)$$

Bayes' theorem

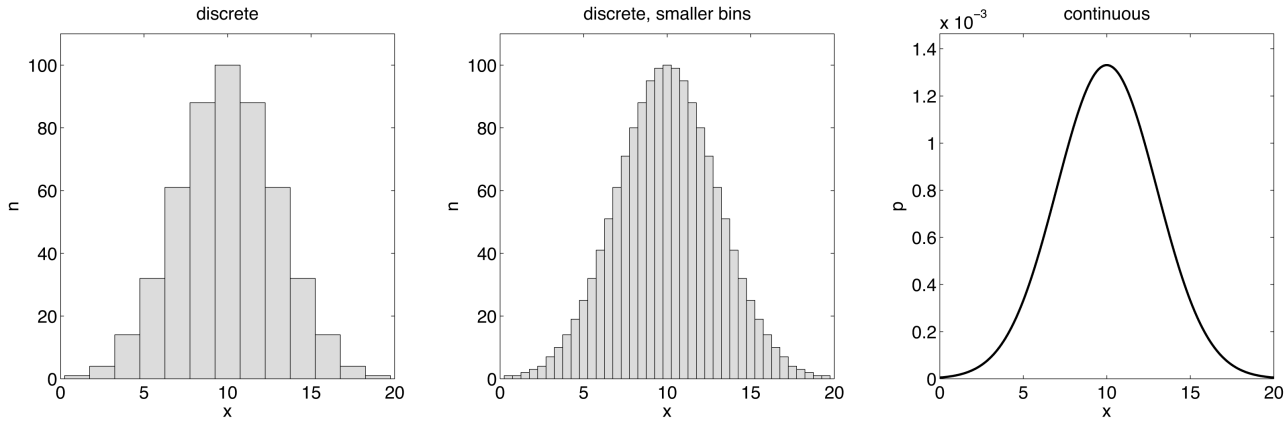
$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

Posterior $\propto$ Likelihood $\times$ Prior

3.7 Continuous random variables probability density functions

What we discussed so far were the probabilities of discrete events, i.e. the random variable X could take on precise values $x_1, x_2, ..., x_n$, but nothing in between. When we shrink the bin sizes of x_i to arbitrarily small values, there are indefinitely many possible values that the random variable can take on. In turn, the probability to take on any one specific value is indefinitely small. We then call the spectrum of values that the random variable X can take on *continuous*.

ch3fig3.m

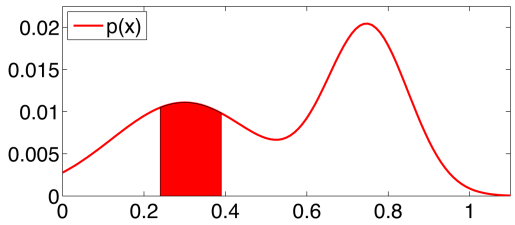


By proper definition, a random variable X is called *discrete* if the range

of X is finite or countably infinite. It is called *continuous* if the range of X is uncountably infinite. *Range* describes number of values the random variable can take on. A countably infinite range can be thought of as a string of bins ranging from $-\infty$ to ∞. It is called *countable* because for any given bounded interval within the range, you can count the number of possible outcomes, or bins. In contrast, a continuous spectrum is characterized by the non-existance of bins - for any two values of x, no matter how close, you can always find another value of x that is in between the two. Thus, even in the smallest interval, there is an infinite number of values that X can take on, hence uncountably infinite.

The curve describing the relative probability for the random variable X to take on a given value (figure above, right-hand side) is called a *probability density function (PDF)*, henceforth labeled $p(x)$. Because in a continuous spectrum the probability of a random variable X to take on any precise value is indefinitely small, the value of a probability density function at a singular point x_i has no meaning. Only the area under the curve can be interpreted. Therefore, in order to make a statement about a probability in a continuous distribution, we need to calculate the probability of X to take on a value within a certain interval (a, b).

The code for this and the following figure can be found in *ch3fig4.m*.



Here are a few more notes about probability density functions:

First, because the probability of X to take on *any* value at all is always 1, the area under the entire curve must also always be exactly 1:

$$\int_{-\infty}^{\infty} p(x)dx = 1.$$

Second, there is no such thing as a negative probability, hence

$$p(x) \geq 0.$$

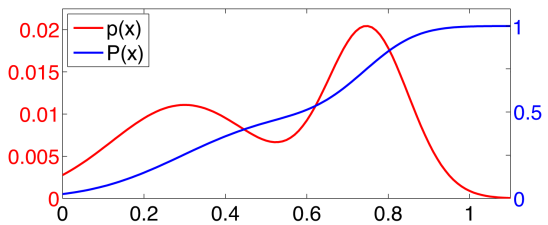
As mentioned, the probability for X to fall into a certain interval of values is given by the area under the curve within said interval:

$$p(x \in (a, b)) = \int_{a}^{b} p(x)dx$$

We can further define a *cumulative density function (CDF)*:

z here denotes a point on the x-axis. The CDF $P(x)$ displays the total area under $p(x)$ up to the point z.

$$P(z) = \int_{-\infty}^{z} p(x)dx$$



3.8 Transformation of densities

In regular functions, a change of variables is easily done. Say the functional relation is given by $x = g(y)$ and you want to express $f(x)$ as a function of y. For this, you simply substitute x by $g(y)$ and end up with $\tilde{f}(y) = f(g(y))$. However, things are a little different with probability density functions. While usually the relevant information of a function is given by its value at a certain point, the relevant information of a probability function is the area underneath it.

The following derivation isn't mathematically accurate, but it will help you to get an intuition about how to do it. Since the relevant information is the area underneath it, we transform the function area by area instead of point by point. For this, we chop the area under the curve into little stripes of width δx. For increasingly smaller values of δx, the area under the curve gets arbitrarily close to $p_x(x)\delta x$. This must correspond to $p_y(y)\delta y$:

$$p_y(y)\delta y \simeq p_x(x)\delta x$$

$$\Rightarrow p_y(y) = p_x(x) \cdot \left|\frac{\delta x}{\delta y}\right|$$

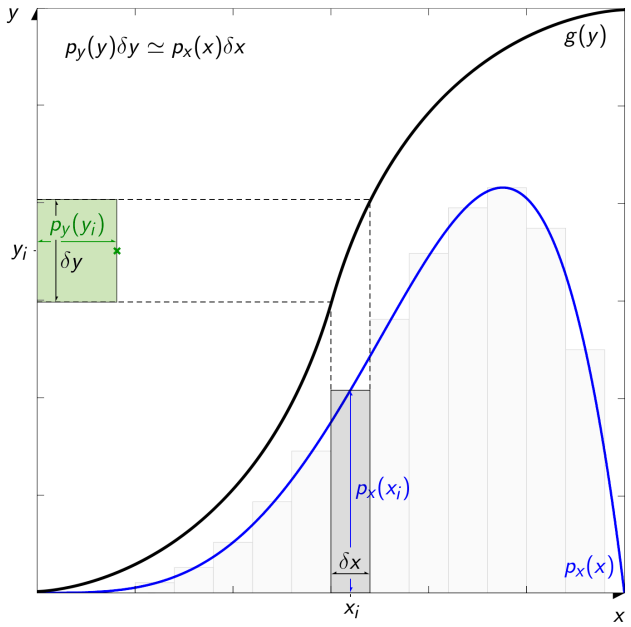
Knowing the functional relation $x = g(y)$, we can substitute $g(y)$ for x in the formula:

$$p_y(y) = p_x(g(y)) \cdot \left|\frac{\delta g(y)}{\delta y}\right|$$

And since $\left|\frac{\delta g(y)}{\delta y}\right| = |g'(y)|$, we end up with

$$p_y(y) = p_x(g(y)) \cdot |g'(y)|.$$

This concept is illustrated in the following figure:



One more practical advice: In the real world you are usually not given a function $x = g(y)$, but rather one that gives you y as a function of x.

Note that the direction of this transformation was arbitrarily chosen. If we are given a distribution $p_y(y)$ and want to derive an expression for $p_x(x)$, we need to use $p_x(x) = p_y(g(x)) \cdot |g'(x)|$.

We have a function $p_x(x)$ that we want to express in terms of a function $p_y(y)$. The functional relationship between x and y is given by $x = g(y)$. Since we are interested in the area under the curve, we chop $p_x(x)$ into little stripes of width δx. The area for each stripe is thus given by $p_x(x_i)\delta x$. We know that the area under the resulting curve $p_y(y)$ must be the same, in this example the green stripe must have the same area as the grey stripe. The green stripe has the area $p_y(y_i)\delta y$. This is how we get to $p_y(y)\delta y \simeq p_x(x)\delta x$ for the whole function - the rest follows from pushing the formulas around.

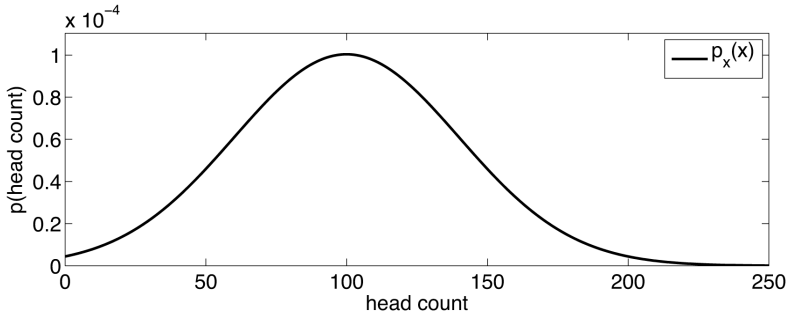
Should this be the case you will have to invert the function first, before you plug it into the formula.

3.8.1 Example: Transformation of densities

This may feel relatively abstract to you, so let's apply this in a practical example. Say we're planing a party and we want to know about the beer consumption per head. We roughly know how many students to expect and we know that people will drink more beer when the party is better, i.e. when the head count is higher.

Let's start fresh and define some variables. x will be the headcount, y will be the beer consumption. Consistently, $p_x(x)$ will be a PDF representing our expectations for the head count at the party. It could look like this:

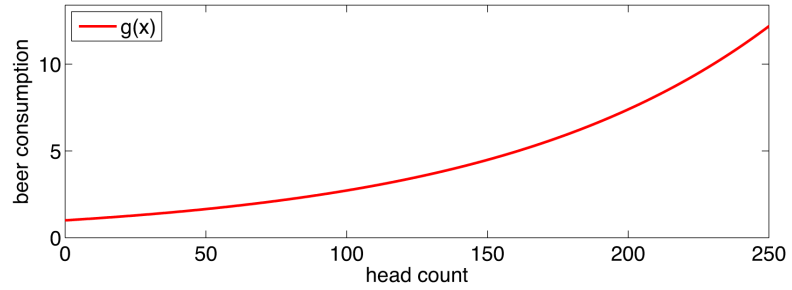
$p_x(x)$ could for instance be a normal distribution with $\mu = 100$ and some $\sigma = 40$. Don't worry though if this doesn't mean anything to you yet. You will learn about distributions in section 5 of the script.



What we want, though, is a PDF $p_y(y)$ describing the beer consumption per head. Luckily, we know how the beer consumption per head relates to the head count. This relation is given to us as a function of x:

$g : y = e^{0.01x}$ is another notation for the same thing.

$$g(x) = e^{0.01x}$$

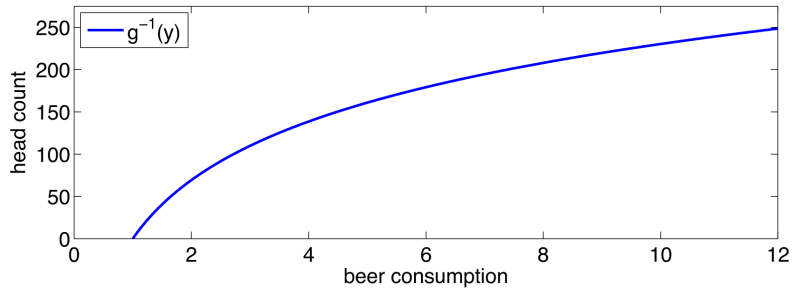


When we look at the formula we developed in the last section, we can see that we need the relational function to be a function of y and not of x, i.e. head count as a function of beer consumption instead of the other way around. Thus, we need to invert $g(x)$. The inverse will be a function of y, so it will go by the name $g^{-1}(y)$:

$p_y(y) = p_x(g(y)) \cdot |g'(y)|$

If you are already given a relational function that is a function of y (which would be called $g(y)$) - this step is not necessary. To prevent some confusion: In our example, we are *not* given any $g(y)$. The only "g" we are given is and always will be a function of x, thus "$g(x)$". Only its inverse "g^{-1}" is a function of y (and never of x), therefore called "$g^{-1}(y)$".

$$\begin{aligned} y &= e^{0.01x} \quad | \quad \ln \\ \ln y &= 0.01x \quad | \quad \cdot 100 \\ 100 \cdot \ln y &= x \end{aligned}$$



Now we have $g(y)$, next up we derive $g'(y)$:

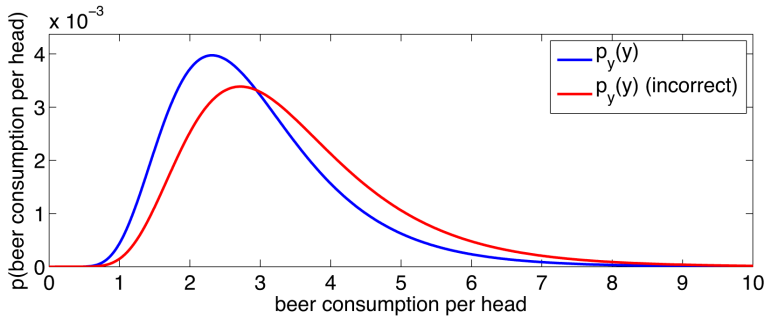
$$\begin{aligned} g(y) &= 100ln(y) \\ g'(y) &= \frac{100}{y} \end{aligned}$$

Last thing for us to do is to simply put $g(y)$ and $g'(y)$ into the formula:

$$p_y(y) = p_x(g(y)) \cdot |g'(y)|$$

$$p_y(y) = p_x(100 \ln y) \cdot \frac{100}{y}$$

The graph for $p_y(y)$ is shown in the next figure (blue). For comparison, we also included the incorrect graph, i.e. the one where the factor $|g'(y)|$ was left out (red):



For those of you who already know about the normal distribution:

$p_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

$p_x(g(y)) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(100 \ln y-\mu)^2}{2\sigma^2}}$

This should clarify how to plug a function into another function. Note that $p_x(x) \cdot g(y) \neq p_x(g(y))$.

For a better understanding of how these figures came into being, check out *ch3fig5.mat*.

3.9 Expectations

An important use of probability density functions is the generation of expectations (also called expected values) of functions. The expectation of a function $f(x)$ is its weighted average. To weight a function $f(x)$ means to multiply it by a PDF $p(x)$. For a continuous $f(x)$, the expectation is

$$\mathbb{E}[f] = \int f(x)p(x)dx.$$

For a discrete function $f(x)$, the expectation is

$$\mathbb{E}[f] = \sum_x f(x)p(x).$$

To get a better grasp of the concept, let's consider a regular die. This die can be represented by a discrete distribution with six equally probable outcomes $\vec{x} = \{1, 2, 3, 4, 5, 6\}$:

$$\mathbb{E}[\text{die}] = \frac{1}{6} \cdot 1 + \frac{1}{6} \cdot 2 + ... + \frac{1}{6} \cdot 6 = 3.5$$

Note that the expectation of a function isn't necessarily a probable or even a possible outcome of the function. For instance, rolling a 3.5 with a regular die is rather unlikely.

For a n-sided die with the numbers x_i on its faces (or alternatively any discrete uniform distribution with n possible outcomes which take the values x_i), the expectation is

$$\mathbb{E} = \frac{1}{n} \sum_i x_i.$$

Accordingly, a 6-sided die with the faces $\vec{x} = \{1, 2, 3, 4, 30, 50\}$ has the expectation

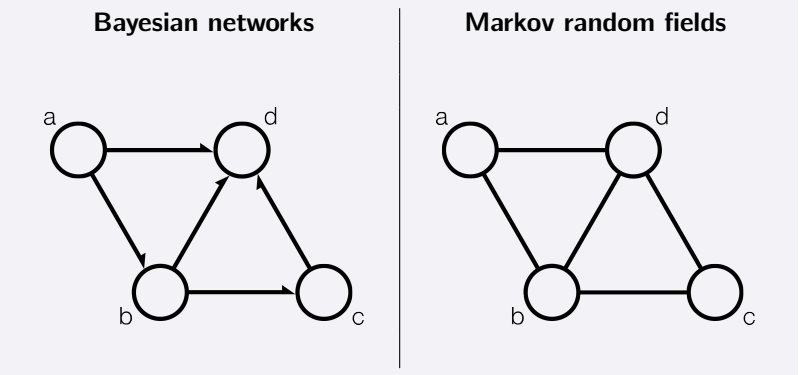
$$\mathbb{E}[\text{die}_{\text{irregular}}] = \frac{1}{6} \cdot 90 = 15.$$

While for (discrete) symmetrical distributions the expectation is always equal to the mean of all possible outcomes, this is not the case for asymmetrical distributions. It may help to think of the expectation as the mean of an infinite number of sample values drawn from the distribution.

4 Graphical models

In this chapter, we'll go slightly off the path of model fitting in order to deepen our understanding of probabilistic relations. *Graphical models* are a way of visualizing probabilistic relations between variables, and make our lives a lot easier for some operations. They are comprised of *nodes* (variables) and *links* (their probabilistic relations). The two main types of graphical models are *Bayesian networks*, in which links are directed, i.e. they indicate causal relations, and *Markov random fields*, in which links are undirected. In this lecture, we will leave the Markov random fields aside and concern ourselves with Bayesian networks only.

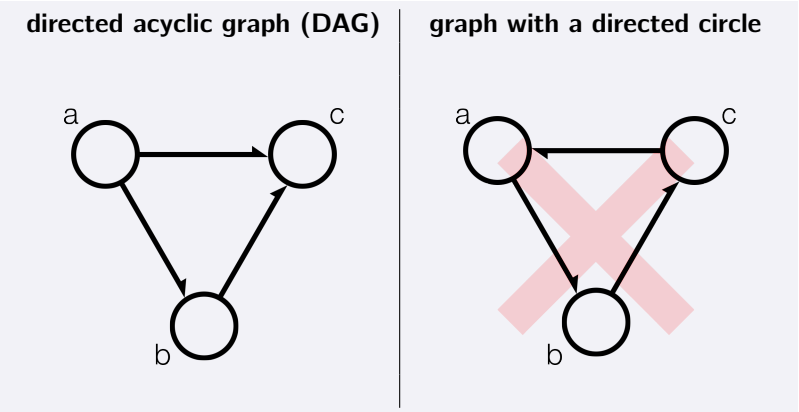
It's still relevant for the exam though.



4.1 Bayesian networks

Let's lay down some ground rules. All links in a Bayesian network are directed links. A directed link connects a *parent* node to a *child* node. A Bayesian network may or may not be *fully connected*, i.e. all nodes may or may not be connected to each other. And finally, there must be no directed circles (also called closed paths), the network must be a *directed acyclic graph (DAG)*. The latter is also equivalent to the notion that the nodes are ordered such that there is no link from a higher numbered node to a lower numbered node.

In the left-hand figure above a is a parent of both b and d. b is therefore a child of a, but also a parent of c and d and so on.



We now consider an arbitrary joint distribution $p(a, b, c)$. As it is, this joint distribution does not contain any directed relations, for all variables are created equal. However, the product rule allows us to bring some inequality into this society:

Too lazy to scroll up? Here you are: $p(X, Y) = p(Y|X)p(X)$

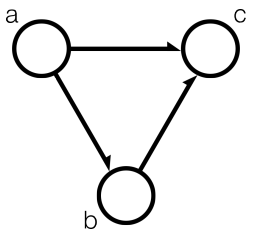
$$p(a, b, c) = p(c|a, b)p(a, b)$$

We took c out of the joint probability and replaced it with the conditional probability times the marginal probability (with respect to c). Repeating this step gets us here:

conditional probability · marginal probability = joint probability
$p(a|b) \cdot p(b) = p(a, b)$

$$p(a, b, c) = p(c|a, b)p(a, b) = p(c|a, b)p(b|a)p(a)$$

The joint probability has now been broken down (factorized) into fragments of conditional and marginal probabilities. This is a requirement for the DAG that we can now produce:



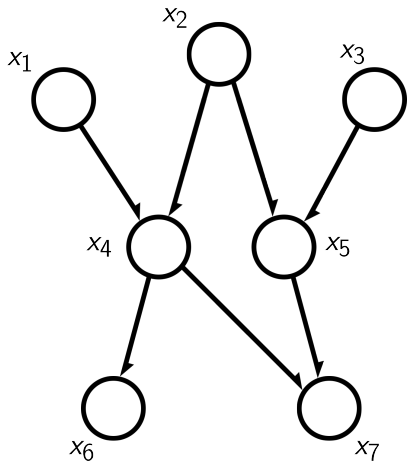
Note that we freely chose the specific ordering here. Since in the joint probability $p(a, b, c)$ all three variables are equal, we could have chosen any other hierarchy between them instead (this however would have resulted in a different graph). The more general paradigm for breaking down a joint probability with k variables looks like this:

$$p(x_1, ..., x_k) = p(x_k|x_1, ..., x_{k-1})...p(x_2|x_1)p(x_1)$$

Going back to the DAG above and its probability distribution, we can see how the graph results from the probability distribution. All fragments of the distribution follow the pattern $p(child|parents)$ and for each node there is exactly one of these conditional or marginal probability distributions. This results in the *factorization* properties of Bayesian networks:

$$p(\vec{x}) = \prod_k p(x_k|pa_k)$$

With pa_k meaning "a list of all parents of x_k". These formulas should allow you to transform an arbitrary joint distribution into a DAG and vice versa. For the sake of repetition, we look at the following graph:



This DAG is an example of a not fully connected graph (as you can see, many of the possible links are missing) and it factorizes to

$$p(\vec{x}) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2)p(x_5|x_2, x_3)p(x_6|x_4)p(x_7|x_4, x_5).$$

4.2 Statistical independence of variables

We now take a little break from the graphs to introduce the concept of *statistical independence*. Statistical independence of variables generally means that the outcome of one variable will not influence the outcome of the other variables in question. In other words, two variables are independent when their joint probability is equal to the product of their marginal probabilities:

$$p(x, y) = p(x)p(y) \text{when } x \text{ and } y \text{ are independent}$$

Of course, this can be expanded to groups of variables. The respective equation, which must be fulfilled for statistical independence of all variables is

$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2)\cdot...\cdot p(x_n)$for all x_i independent of one another.

4.2.1 Conditional independence of variables

When the outcome of a variable is *observed*, it is henceforth known and will therefore not be influenced by future outcomes of other variables. For instance, this applies to the data points t_i in polynomial curve fitting. When the physical measurements are done, the outcome of other variables in the model obviously can't influence the dataset anymore, thus the variable is *fixed*. The outcome of other variables becomes *conditioned*on the fixed variable, which can change the independence properties within the distribution. Say we have a distribution of three variables a, b and c such that the outcome of a isn't independent of the outcome of b:

$$a \not\perp\!\!\!\perp b$$

If observing c changes this, we say a is independent of b, *given* c:

$$a \perp\!\!\!\perp b \mid c$$

This is the case exactly when

$$p(a|b, c) = p(a|c),$$

i.e. when the conditioning on b is made obsolete by c. And if this is the case, the distribution $p(a, b|c)$ transforms as follows:

$$p(a, b|c) = p(a|b, c)p(b|c) = p(a|c)p(b|c)$$

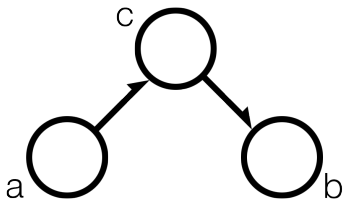
Which is exactly the expression $p(a, b) = p(a)p(b)$ (known to us as the definition of statistical independence) extended by the condition c. This leads us to the general definition of *statistical conditional independence*:

$$p(x_1, x_2, ..., x_n|y) = p(x_1|y)p(x_2|y) \cdot ... \cdot p(x_n|y)$$

4.3 D-separation: Statistical independence in graphs

4.3.1 Head-to-tail nodes

Alright, back to graphs. Would you look at this:



The node c is said to be *head-to-tail* with respect to this path because it is connected to the head of previous node and the tail of the next node on the path. One way to show that a is independent of b would be to show that the joint distribution $p(a, b, c)$ can be factorized and brought to a form that fulfills the requirement of statistical independence. To this end, we factorize the joint distribution as follows:

Statistical independence:
$p(a, b) = p(a)p(b)$

$$p(a, b, c) = p(b|c)p(c|a)p(a)$$

And *marginalize* both sides with respect to c, to get rid of it on the left side, then see if the right side can be brought to the form $p(a)p(b)$.

$$\begin{aligned}
\sum_c p(a, b, c) &= \sum_c p(b|c)p(c|a)p(a) \\
p(a, b) &= p(a) \sum_c p(b|c)p(c|a) \\
&= p(a) \sum_c p(b, c|a) \\
&= p(a)p(b|a)
\end{aligned}$$

And that's where it ends. As you can see, we failed to bring the equation to the desired form $p(a, b) = p(a)p(b)$, ergo it doesn't fulfill the requirement for statistical independence. However, this doesn't suffice to prove the opposite, i.e. that there is a dependency between a and b.

With more complex probability distributions, it becomes increasingly tricky to determine whether or not two variables are statistically independent. Performing analytical operations on the distributions can be time-consuming and prone to error. Luckily, graphs provide an easy way to determine, whether or not two variables are independent. When we speak of graphs and nodes instead of probability distributions and variables, we also speak of *d-separation* (where the d stands for *directed*) instead of statistical independence - however, the two can be regarded as synonyms.

The way to read whether or not two nodes are d-separated is to look at the path between them. This path can either be free or blocked. If it is free, the nodes are not d-separated (and the respective variables are not independent), if it is blocked, they are d-separated (and the respective variables are independent of each other).

If this seems counterintuitive to you, try to think of the path as a possible connection between the nodes. If the path is free, the connection exists and they can influence each other, if it is blocked there is no connection.

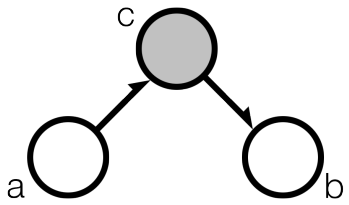
Referring to the direction of the links , the node c from the DAG above is said to be *head-to-tail*. The variables a and b are causally linked: If a changes, b changes, too. We say a and b are not d-separated (independent), the path is free.

"$|\emptyset$" (speak given the empty set) is not a necessary part of the notation and typically left out, it just clarifies that no nodes were observed/ fixed.

$$a \not\perp\!\!\!\perp b \,|\, \emptyset$$

4.3.2 Fixed head-to-tail nodes

Let us now take a look at what happens when we condition on variable c. In graphical models, observed/ fixed variables are indicated by shaded nodes.



We will check analytically for conditional independence by trying to bring the joint distribution to a form that fulfills the requirement for statistical conditional independence. To represent the fact that c is observed, the joint distribution must be divided by $p(c)$, i.e. we start with $\frac{p(a,b,c)}{p(c)}$. Just like before, we set up an equation by factorizing the joint distribution and then try to bring both sides to a form that fulfills the requirement:

Conditional independence:
$p(a, b|c) = p(a|c)p(b|c)$

$$\frac{p(a,b,c)}{p(c)} = \frac{p(a)p(c|a)p(b|c)}{p(c)}$$

$$p(a,b|c) = \frac{p(a,c)p(b|c)}{p(c)}$$

$$p(a,b|c) = p(a|c)p(b|c)$$

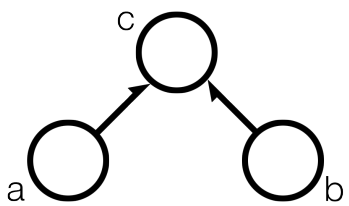
Ergo, the requirement for conditional independence is met, a is conditionally independent of b, given c:

$$a \perp\!\!\!\perp b \,|\, c$$

Fixing the node has thus flipped its impact on the d-separation property of the graph. We will see that this is a general rule. If an unobserved node caused the path to be free, it will block the path when it is observed (fixed) and vice versa.

4.4 Head-to-head nodes

We will go though the remaining types of nodes (in terms of their links' directions) a little quicker. Everything is done in exactly the same way as for the previous ones.



Analytical determination of the statistical independence of a and b:

$$p(a,b,c) = p(a)p(b)p(c|a,b) \mid \sum_c$$

$$\sum_c p(a,b,c) = \sum_c p(a)p(b)p(c|a,b)$$

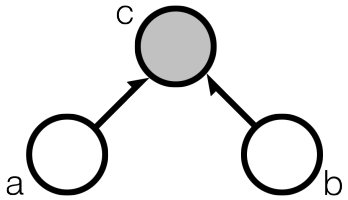
$$p(a,b) = p(a)p(b)\sum_c p(c|a,b)$$

$$p(a,b) = p(a)p(b)$$

Requirement for statistical independence fulfilled:

$$a \perp\!\!\!\perp b \mid \emptyset$$

4.5 Fixed head-to-head nodes



Analytical determination of the conditional independence of a and b:

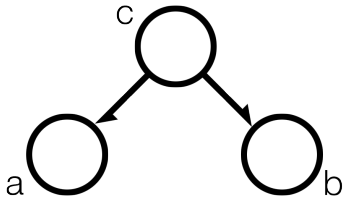
$$\frac{p(a,b,c)}{p(c)} = \frac{p(a)p(b)p(c|a,b)}{p(c)}$$

$$p(a,b|c) = \frac{p(a)p(b)p(c|a,b)}{p(c)}$$

We don't get any further here, requirement for conditional independence not fulfilled:

$$a \not\perp\!\!\!\perp b \mid c.$$

4.6 Tail-to-tail nodes



Analytical determination of the statistical independence of a and b:

$$p(a,b,c) = p(b|c)p(a|c)p(c) \mid \sum_c$$

$$\sum_c p(a,b,c) = \sum_c p(b|c)p(a|c)p(c)$$

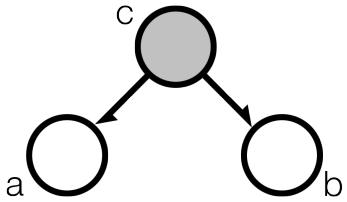
$$p(a,b) = \sum_c p(b|c)p(a,c)$$

$$p(a,b) = p(a)\sum_c p(b|c)$$

Requirement for statistical independence not fulfilled:

$$a \not\perp\!\!\!\perp b \mid \emptyset$$

4.6.1 Fixed tail-to-tail nodes



Analytical determination of the conditional independence of a and b:

$$\frac{p(a,b,c)}{p(c)} = \frac{p(a|c)p(b|c)p(c)}{p(c)}$$

$$p(a,b|c) = \frac{p(c,a)p(b|c)}{p(c)}$$

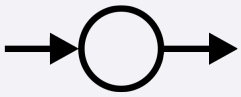
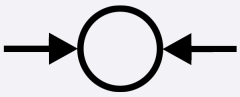
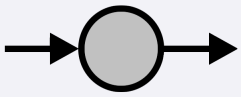
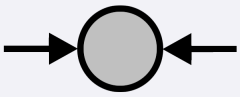
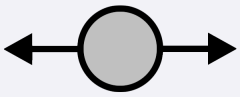
$$p(a,b|c) = p(a|c)p(b|c)$$

And we're done. Requirement for conditional independence fulfilled:

$$a \perp\!\!\!\perp b \,|\, c.$$

4.7 Overview & more

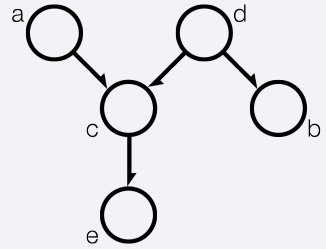
At this point, we sum up what we've learned so far about the d-separation properties of DAGs.

head-to-tail	**head-to-head**	**tail-to-tail**
		
path free	path blocked	path free
$a \not\perp\!\!\!\perp b \,\|\, \emptyset$	$a \perp\!\!\!\perp b \,\|\, \emptyset$	$a \not\perp\!\!\!\perp b \,\|\, \emptyset$
not d-separated	d-separated	not d-separated
head-to-tail (fixed)	**head-to-head (fixed)**	**tail-to-tail (fixed)**
		
path blocked	path free	path blocked
$a \perp\!\!\!\perp b \,\|\, c$	$a \not\perp\!\!\!\perp b \,\|\, c$	$a \perp\!\!\!\perp b \,\|\, c$
d-separated	not d-separated	d-separated

Of course, paths are often more than two links long. Therefore, it is important to know that a path is blocked when at least one of the nodes on the path blocks it. It may happen that in a larger graph there are several paths between two nodes. In that case, the two nodes are only d-separated (independent) if all paths between them are blocked. Let's

This makes a lot of sense if you think of the path as a connection again.

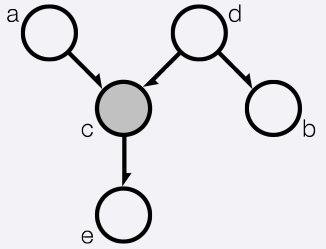
have a look at a practical example:



$a \perp\!\!\!\perp b \mid \emptyset$

"a d-separated from b given the empty set"

The path from a to b is blocked by the head-to-head connection at node c.

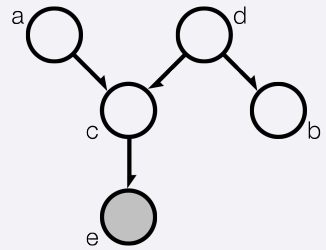


$a \not\perp\!\!\!\perp b \mid c$

"a not d-separated from b given c"

The path from a to b is free, since the head-to-head connection at node c is fixed.

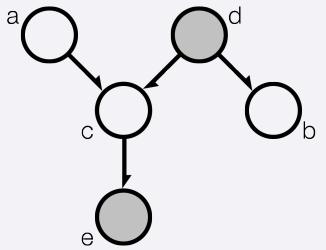
One more thing: A node y is called a *descendant* of node x, if y is a child of x or if there is a path from x to y that consists only of head-to-tail nodes. In the example graphs above, node e is a descendant of c. When a descendant of a node is fixed it has the same effect on the node as if it were fixed itself.



$a \not\perp\!\!\!\perp b \mid e$

"a not d-separated from b given e"

The path from a to b is free, since the head-to-head connection at node c is unblocked by e.



$a \perp\!\!\!\perp b \mid e, d$

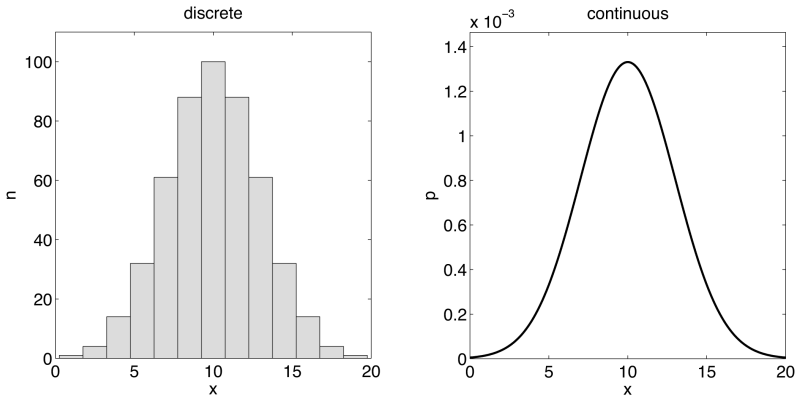
"a d-separated from b given e and d"

The path from a to b is blocked, since the tail-to-tail connection at node d is fixed.

And that's it for our little detour into the world of graphical models.

5 Probability distributions

A probability distribution assigns a probability to each possible outcome of an experiment, survey or other form data measurement. We've learned already that we distinguish between discrete and continuous distributions of random variables (section 3.7). In a discrete distribution, there is a finite (or countably infinite) number of values that the random corresponding variable can take on. In a continuous distribution there are no distinct bins. Instead, between any two values there is always another value that the random variable can take on.



5.1 Discrete distributions

We will first have a look at the three discrete distributions that are of most concern for us: The Bernoulli distribution, the binomial distribution and the Poisson distribution.

How these distributions follow from one another is currently not included in the script. Please refer to the lecture slides for that.

5.1.1 Bernoulli distribution

The Bernoulli distribution is a binary distribution. It can be used to describe trials that result in exactly one of two possible outcomes, for example flipping a coin. To this end, we consider a random variable $x \in \{0, 1\}$, where $x = 1$ could denote "heads" and $x = 0$ could denote "tails". If the coin is fair, the probability of heads is equal to the probability of tails: $p(x = 1) = p(x = 2) = 0.5$. We define

$$\mu = p(x = 1),$$

as the parameter of this distribution. Since only two outcomes are possible, the probability of one outcome is always the complementary probability of the other outcome:

$$p(x = 0) = 1 - \mu.$$

These two cases can now be put into a single equation that defines the Bernoulli distribution:

If $x = 0$, the first part of the equation will disappear. If $x = 1$, the second part will disappear.

$$p(x) = \mu^x (1 - \mu)^{1-x}$$

In this lecture, we use a slightly different notation emphasizing that this is a distribution depending on the parameter μ.

Don't be confused: $\mu = p(x = 1)$ still holds, $p(x)$ and $p(x|\mu)$ mean the same thing here.

$$p(x|\mu) = \mu^x (1 - \mu)^{1-x}$$

For every distribution we are also interested in the expected value and the variance. For the Bernoulli distributions these are

$$\mathbb{E}[x] = \mu$$

and

$$var[x] = \mu(1-\mu).$$

5.1.2 Binomial distribution

The binomial distribution is a natural extension of the Bernoulli distribution. Instead of single trials, it describes probabilities for the outcome of multiple events. To be more precise, it describes the probability of getting k positive outcomes ($x = 1$) in N trials. For example, if we toss a coin $N = 4$ times, whats the probability of getting heads $k = 3$ out of those four times? The binomial distribution is given by

$$\text{Bin}(k|N,\mu) = \binom{N}{k}\mu^k(1-\mu)^{N-k}.$$

The parameter μ still denotes the probability of a positive outcome in a single trial. As for $\binom{N}{k}$, this is defined as follows:

read "N choose k"

$$\frac{N!}{k!(N-k)!}$$

$k!$ reads *"k factorial"* and simply means $1 \cdot 2 \cdot \ldots \cdot k$:

$$\frac{N!}{k!(N-k)!} = \frac{1 \cdot 2 \cdot \ldots \cdot N}{1 \cdot 2 \cdot \ldots \cdot k \cdot 1 \cdot 2 \cdot \ldots \cdot (N-k)}$$

For our example ($N = 4$, $k = 3$, fair coin: $\mu = 0.5$), this looks as follows:

$$\begin{aligned}
\text{Bin}(k|N,\mu) &= \tbinom{N}{k}\mu^k(1-\mu)^{N-k} \\
\text{Bin}(3|4,.5) &= \tfrac{1\cdot2\cdot3\cdot4}{1\cdot2\cdot3\cdot1}0.5^3(1-0.5)^1 \\
&= \tfrac{24}{6}\cdot 0.125 \cdot 0.5 \\
&= 4 \cdot 0.0625 \\
&= 0.25
\end{aligned}$$

Expected value and variance:

$$\mathbb{E}[k] = \sum_{k=0}^{N} k \cdot \text{Bin}(k|N,\mu) = N\mu$$

$$var[k] = \sum_{k=0}^{N}(k - \mathbb{E}[k])^2 \cdot \text{Bin}(k|N,\mu) = N\mu(1-\mu)$$

5.1.3 Poisson distribution

We consider a fixed time interval in which the expected number of (positive) events is given by the parameter λ, i.e. we know that an event will occur, on averge, λ times within the interval. Of course, on some trials the number of events might be a little above λ, on some it might be a little below it. We denote $p(k|\lambda)$ the probability of k

events occurring in this interval, when the expected number of events is λ. The Poisson distribution is given by

$$p(k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}.$$

Expectation and variance:

$$\mathbb{E}[k] = \lambda$$
$$var[k] = \lambda$$

For a practical example, consider a spiking neuron: We may know from a high number of previous trials that for a certain time interval the neuron will fire $\lambda = 8$ times on average. Now we want to know the probability of $k = 6$ spikes occurring within this interval. For this, we use the Poisson distribution:

$$p(6|\lambda = 8) = \frac{8^6 \cdot e^{-8}}{6!} \approx 0.122$$

5.2 Continuous distributions

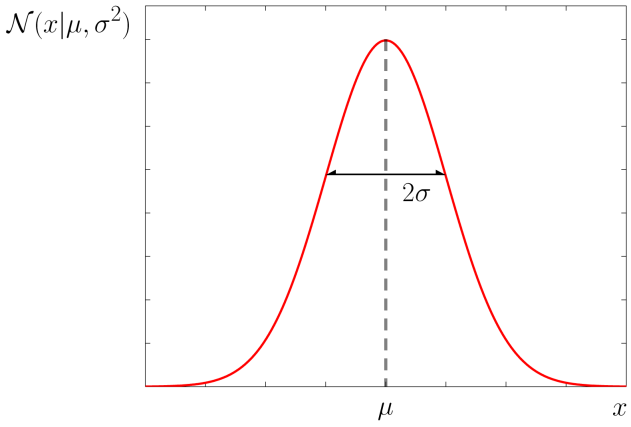
5.2.1 Normal distribution

The normal distribution (also called *Gaussian distribution*) is given by

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

This gives rise to the characteristic bell curve of the Gaussian distribution:

ch5fig2.m



The function is always positive:

$$\mathcal{N}(x|\mu, \sigma^2) > 0$$

And as mentioned earlier, the area under the curve is always 1:

$$\int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) dx = 1$$

The two parameters of the normal distribution, μ and σ^2, set the expected value and the variance:

$$\mathbb{E}[x] = \mu$$

$$var[x] = \sigma^2$$

A *standard normal distribution* is a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$.

A *z-transformation* can be used to conform any given dataset to a standard normal distribution:

$$z_i = \frac{x_i - \mu}{\sigma}$$

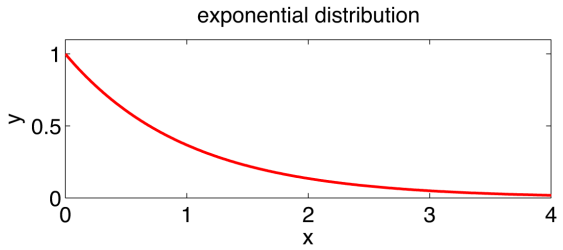
Here, μ and σ are the mean and standard deviation of the dataset. The z-transformation is a simple shift and scale algorithm, a z-transformed dataset has a mean of $\mu = 0$ and a standard deviation of $\sigma = 1$. Note that this does not change the shape of the distribution - if it wasn't a normal distribution before, it won't be one now. A z-transformed dataset is easier to handle in many statistical procedures.

What makes the normal distribution stand out in comparison to other distributions is the *central limit theorem*. The central limit theorem states that the distribution of sample means will get approximately normal with increasing sample size, regardless of the distribution from which we are sampling. In mathematical terms, the sample means

$$S_n = \frac{1}{n} \sum_i x_i$$

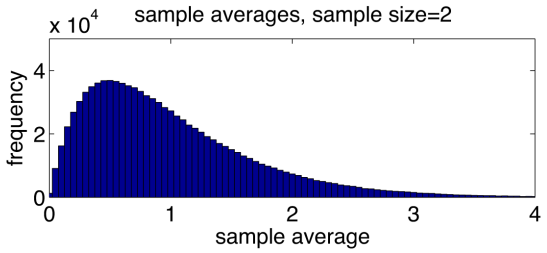
will, for large ns, approximate a normal distribution with mean μ and variance $\frac{1}{n}\sigma^2$. Let's see what this means. Say we start out with an exponential distribution:

This would work just the same with almost any other distribution.

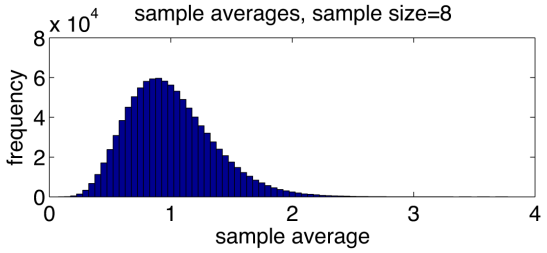


From this we draw a sample of size two, i.e. we draw two random numbers. As you can read from the graph, the probability to draw a low number close to zero is a lot higher than to draw a large number. We take the mean of the two values, and note it down as our first sample mean. We then repeat this process one million times. We sort all sample means by size and then plot a histogram showing their distribution:

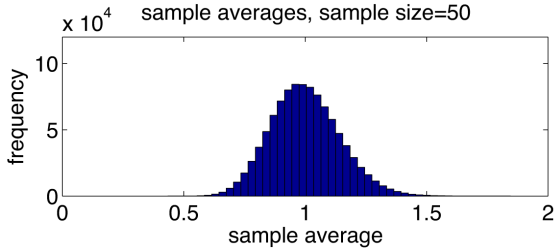
ch5fig3.m



As you can see, this still resembles the exponential distribution that we sampled from, but is already rounded off a little. Now we repeat the sampling process with an increased the sample size of eight, then again plot the distribution of the (new) sample means:



This looks more normal, but isn't quite normal yet. Let's increase the sample size to 50:



Now we're cooking - this plot is quite close to normal. As mentioned earlier, this principle works with pretty much any distribution that you will find in the real world. Nonetheless, it is possible to construct distributions whose sample means will not be normally distributed, no matter how large the sample size.

5.2.2 Gamma distribution

The gamma distribution is given by

$$p(x) = c \cdot x^{\alpha-1} e^{-\frac{x}{\beta}}$$

with x, α, $\beta > 0$ and c chosen such that $\int_{-\infty}^{\infty} p(x)dx = 1$ (normalization). To be precise, $c = \frac{1}{\Gamma(\alpha)\beta^{\alpha}}$. Note that this is independent of x and therefore constant as long as α and β are also kept constant.

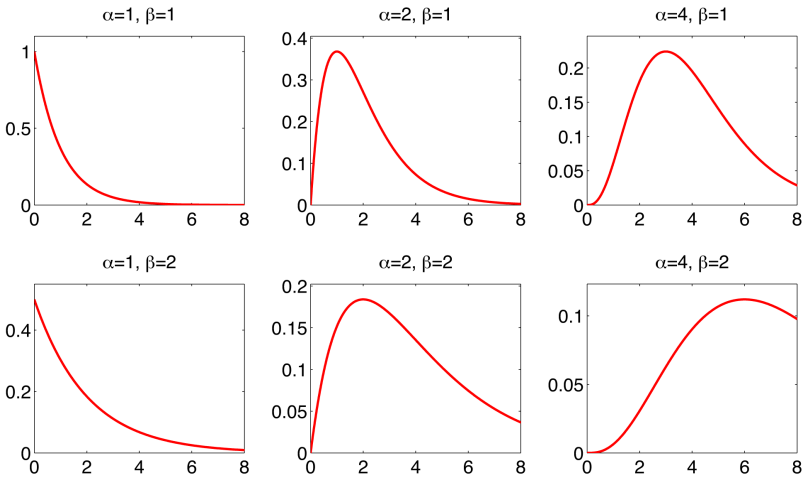
In order to infer a maximum likelihood estimate (see section 6) for the parameters α and β from the distribution, you cannot use the simplified notation, since it only holds for constant α and β. A more complete account of the gamma distribution can be found on wikipedia, where this notation of the gamma function may use k and θ instead of α and β as parameters.

The expectation and variance of the gamma distribution are given by

$$\mathbb{E}[x] = \alpha \cdot \beta$$

$$var[x] = \alpha \cdot \beta^2.$$

Here's how the distribution looks with different values for α and β:



ch5fig4.m

You can see how α determines the shape of the graph, while β acts as a stretching variable. The function peaks at $(\alpha - 1) \cdot \beta$.

5.2.3 Beta distribution

The beta distribution, defined on the interval $[0, 1]$ is given by

$$p(\mu) = c \cdot \mu^{\alpha-1}(1-\mu)^{\beta-1}$$

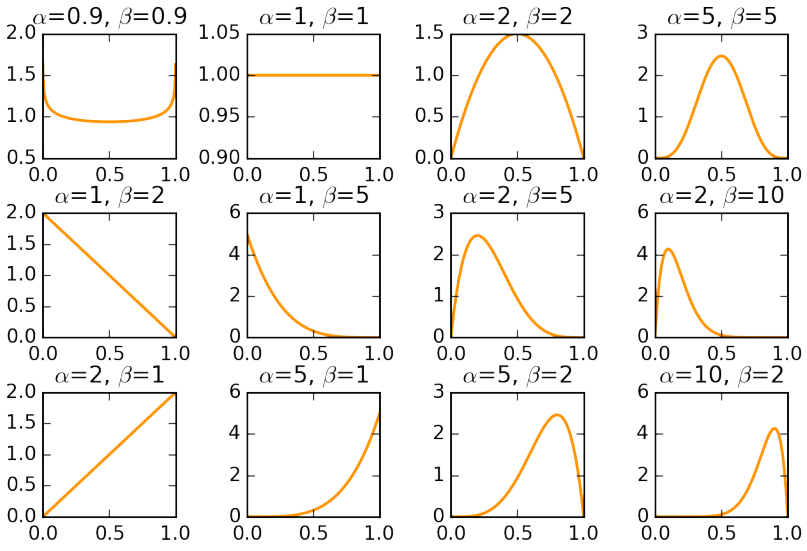
with α, $\beta > 0$ and c chosen such that $\int_0^1 p(\mu)d\mu = 1$. c is given by $\frac{1}{B(\alpha,\beta)}$ and $B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} = \int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du$. Outside of the interval $[0, 1]$ the beta distribution is replaced by $p(\mu) = 0$. A more complete account of the beta distribution can be found on wikipedia.

The expectation and variance of the beta distribution are given by

$$\mathbb{E}[\mu] = \frac{\alpha}{\alpha+\beta}$$

$$var[\mu] = \frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2}.$$

Here's how the distribution looks with different values for α and β:



ch5fig4b.py

5.2.4 χ^2 distribution

The χ^2 distribution is a special case of the gamma distribution, in which $\alpha = \frac{k}{2}$ and $\beta = 2$:

$$p(x) = c \cdot x^{\frac{k}{2}-1} e^{-\frac{x}{2}}$$

$$\mathbb{E}[x] = k$$

$$var[x] = 2k$$

with k denoting the *degrees of freedom*, usually denoted as *d.f.*. This distribution's main purpose lies in the *Pearson's chi-squared goodness of fit test*, developed by - no surprise here - Karl Pearson. The test allows us to tell, whether an observed frequency distribution differs from a theoretical, or expected, frequency distribution. The basic idea is that the bigger the difference is between the expected frequencies and the observed frequencies, the less likely it is that the observations are random samples from the expected distribution. We pick a threshold for this difference, which, when exceeded, indicates that the observations stem from a different distribution than expected, with an allowed error margin of α (usually .05). In other words: If the critical value is exceeded, the chance of the observations nevertheless being random samples from the expected distribution is below 5%. Here's how the test is conducted:

We'll come to what is menat by degrees of freedom in a bit.

We always start out by forming the null hypothesis, also called H_0. This hypothesis is basically the assumption that the observed data is not significantly different from the expected values. This hypothesis will either be accepted or rejected, depending on whether or not the gap between expectations and observations exceeds the critical value. The so-called test-statistic χ^2 serves as a measure of distance between the expected values and the observations:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

E_i is the ith expected frequency, O_i the corresponding ith observed frequency.

As mentioned, we will now compare this value to a critical value χ^2_{crit}. The critical value depends on the degrees of freedom and the chosen α-level. The degrees of freedom are given by the number of possible outcomes minus one:

$$d.f. = n - 1$$

The values for χ^2_{crit} can be found in a table, either in a statistics book or on the internet. The significance level α corresponds to the *p-value*, i.e. if $\alpha = .05$, $p \leq .05$.

The alpha level we choose ourselves. The most commonly used α-level is .05, but other values (e.g. .01) are possible, too. With the degrees of freedom and the α-level at hand, we can look up the corresponding critical value χ^2_{crit} from a table. If $\chi^2 > \chi^2_{crit}$, we reject the null hypothesis. If $\chi^2 < \chi^2_{crit}$, we accept the null hypothesis. Note though, that if we accept the null hypothesis it doesn't mean we're 95% certain that the observations stem from the expected distribution. It only means that the data we have doesn't suffice to prove the opposite. In fact, we can never verify a hypothesis, we can only falsify it. Just because a hypothesis wasn't falsified doesn't mean it is verified.

For example, a coin has two possible outcomes, ergo in a series of coin flips there is one degree of freedom. The degrees of freedom for a series of dice rolls is five, and so on.

And by "prove" we mean "show with $\geq$ 95% certainty"

Let's illustrate this using a practical example. We toss a coin 10 times and count the number of heads to be $O_1 = 6$ with the number of tails being $O_2 = 4$, accordingly. The number of possible outcomes is $n = 2$ (heads and tails), thus we have $k = 2 - 1 = 1$ degree of freedom. We

chose an α-level of .05 and look up the corresponding critical value from the table, in this case $\chi_{crit} = 3.841$. We now calculate the test-statistic (of course, we expect the coin to be fair, i.e. $E_1 = E_2 = 5$):

$$\begin{aligned}\chi^2 &= \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \\ &= \frac{(6-5)^2}{5} + \frac{(4-5)^2}{5} \\ &= \frac{1}{5} + \frac{1}{5} \\ &= \frac{2}{5} \\ &= .4\end{aligned}$$

Since $\chi^2 = .4 < 3.841 = \chi^2_{crit}$ we accept the null hypothesis: We cannot say with $> 95\%$ certainty that this coin, or the way we flipped it, wasn't fair.

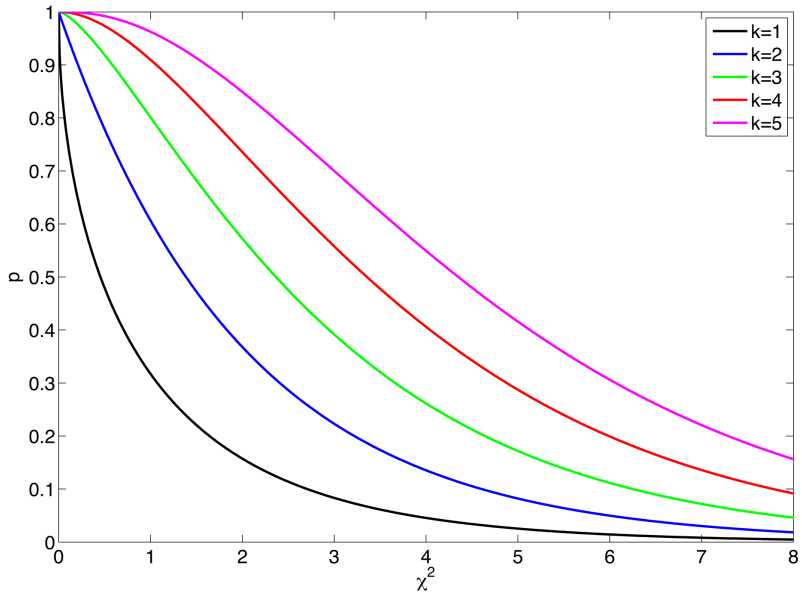
Let us now take the same coin and flip it 100 times. We still expect the coin to be fair, i.e. $E_1 = E_2 = 50$. We observe $O_1 = 60$ heads and $O_2 = 40$ tails, the same *realtive* frequencies as before.

$$\begin{aligned}\chi^2 &= \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \\ &= \frac{(60-50)^2}{50} + \frac{(40-50)^2}{50} \\ &= \frac{100}{50} + \frac{100}{50} \\ &= \frac{200}{50} \\ &= 4\end{aligned}$$

Since $\chi^2 = 4 > 3.841 = \chi^2_{crit}$ we reject the null hypothesis this time: We can now say with $> 95\%$ certainty that this coin, or the way we flipped it, wasn't fair. We obtained different results, even though in both cases there were 20% more heads than expected, and accordingly 20% less tails. The only actual difference between the experiments were the number of observations. This shows us that with more data it is "easier" to get an actual difference to go significant. In turn this should give you an intuition as to why we can never actually verify the null hypothesis. We can say that our data is not sufficient to prove a difference, but we can never prove equality.

This is where the introductory courses to statistics usually stop. However, with our knowledge of distributions we can be more precise than using fixed critical values from a table. We can actually determine the exact p-values, i.e. the level of certainty achieved, from the corresponding (in terms of d.f.) χ^2 distribution:

ch5fig5.m



5.3 Summary

To top this chapter off, let's collect the distributions that we've learned about, their expected values and variances.

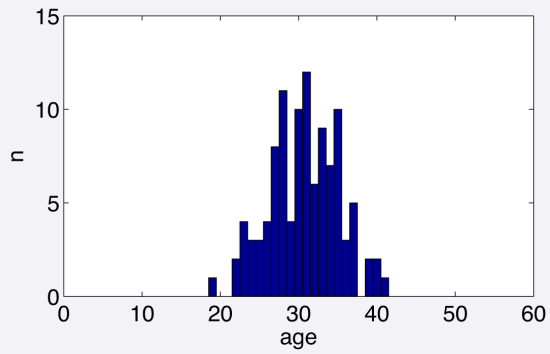
Bernoulli distribution	**Binomial distribution**	**Poisson distribution**
$p(x\|\mu) = \mu^x(1-\mu)^{1-x}$	$Bin(k\|N,\mu) = \binom{N}{k}\mu^k(1-\mu)^{N-k}$	$p(k\|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$
$\mathbb{E}[x] = \mu$	$\mathbb{E}[k] = N\mu$	$\mathbb{E}[k] = \lambda$
$var[x] = \mu(1-\mu)$	$var[k] = N\mu(1-\mu)$	$var[k] = \lambda$
Normal distribution	**Gamma distribution**	**χ^2 distribution**
$\mathcal{N}(x\|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$p(x) = c \cdot x^{\alpha-1} e^{-\frac{x}{\beta}}$	$p(x) = c \cdot x^{\frac{k}{2}-1} e^{-\frac{x}{2}}$
$\mathbb{E}[x] = \mu$	$\mathbb{E}[x] = \alpha \cdot \beta$	$\mathbb{E}[x] = k$
$var[x] = \sigma^2$	$var[x] = \alpha \cdot \beta^2$	$var[x] = 2k$

6 Model Fitting via Maximum Likelihood

Now that the groundwork is done, we can move on to the core of this course, *statistical modeling*. We will first look into models for univariate data sets, then go through the same steps for bivariate data sets. The formulas for the two cases are similar, but not identical. In order to prevent confusion we will try to separate the two cases in this script. The method that we use to model both types of data sets is the maximum likelihood approach, and the remainder of the course will be based on this approach.

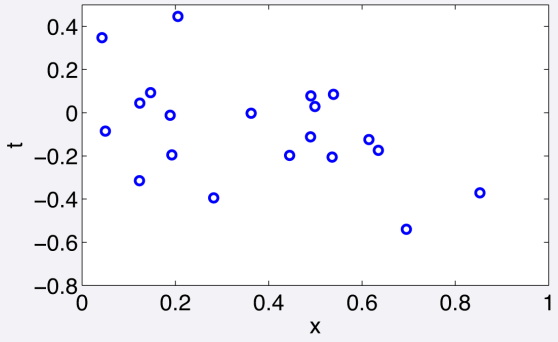
Starting here, the script may diverge somewhat from the structure of the lecture.

univariate data



In univariate datasets there is only one measured variable. This variable could for instance be "age" in a group of people. The histogram above shows the observed frequency of values.

bivariate data



In bivariate datasets, pairs of values are measured. This is often the development of a variable over time. The goal in this case is to make predictions about t for a certain indicated value of x.

6.1 The Maximum Likelihood Approach - Univariate Data Sets

ch6fig1.m, *ch6fig2.m*

Motivation: Let's say we are interested in the distribution of some variable in the real world and we'd like to describe said distribution using statistics. For example, we could be interested in the distribution of body height in the population of students at the uni. Since we can't measure all ~ 10.000 students enrolled at our uni, we randomly pick a sample of 100 students from the population and measure their height. With this data set at hand we want to draw conclusions about the population.

We consider the body height x_i of each person i to be a random sample from the corresponding random variable X_i, which is normally distributed with some defined mean μ_{true} and a defined variance σ^2_{true}. Accordingly, with $n = 100$ we have 100 identically distributed random variables X_i, and from each of these we took one sample x_i. The data set is thus given as a vector $\vec{x} = \{x_1, \ldots, x_n\}$ with $i = \{1, \ldots, n\}$.

The first thing we do now is to come up with a parametric model for the underlying distribution. A parametric model is essentially a function whose exact shape is determined by a set of parameters. In our example, μ_{true} and σ^2_{true} are the parameters defining the underlying model,

Therefore, the distributions introduced in the last chapter can in principle all serve as parametric models.

i.e. the true distribution of body height in the population (which we don't know). To estimate these parameters, we first define a "flexible" normal distribution $\mathcal{N}(\mu, \sigma^2)$, then we look for a way to find the the most probable values for μ and σ^2, given the data we observed. The method we choose to find these most probable parameter values is called the *maximum likelihood approach*, and the estimators found via the maximum likelihood approach will be called the ML estimators μ_{ML} and σ^2_{ML}. There are other ways to determine the estimators, but the ML approach is probably the most common one.

This flexible distribution is what we call *(our) model*.

6.1.1 Rationale of The ML Approach

We started out by defining a flexible parametric model and we now want to find the parameters that are most probable, given the data D we observed. Formally, we are thus looking for the particular parameter set $\vec{\theta}$, for which the distribution $p(\vec{\theta}|D)$ is maximized. Let's decompose this term according to Bayes' theorem:

$$\overbrace{p(\vec{\theta}|D)}^{\text{posterior}} = \frac{\overbrace{p(D|\vec{\theta})}^{\text{likelihood}}\overbrace{p(\vec{\theta})}^{\text{prior}}}{\underbrace{p(D)}_{\text{normalization term}}}$$

We will jump back and forth a little between general statements about the ML approach and the concrete body height example using a normal distribution. We use $\vec{\theta}$ as the name for a set of parameters that defines the shape of a distribution function. The parameter set for a normal distribution is $\vec{\theta} = \{\mu, \sigma^2\}$, that for a binomial distribution is $\vec{\theta} = \{N, \mu\}$, etc.

Here, $p(D)$ is simply a normalization term that won't qualitatively change the shape of the distribution in the numerator. If we use a constant, or "flat" prior distribution $p(\vec{\theta})$, this also won't change the shape of the distribution - and to take the suspense out: We *will* do exactly that until we reach the chapter *Bayesian regression*. What we're left with is the conclusion that, given a constant prior, the likelihood distribution is equal to the posterior distribution, up to a constant factor:

The normalization will make sure the area under the curve is exactly 1: $p(D) = \int p(D|\vec{\theta})p(\vec{\theta})$

$$p(\vec{\theta}|D) = c \cdot p(D|\vec{\theta}) \Leftrightarrow p(\vec{\theta}|D) \propto p(D|\vec{\theta})$$

In turn this means that the parameter set $\vec{\theta}_{ML}$, which maximizes the likelihood $L = p(D|\vec{\theta})$, will also maximize the posterior $p(\vec{\theta}|D)$. This is the rationale that allows us to use the likelihood instead of the posterior to determine the estimators.

For now, all we need to know is that the maximization of the likelihood is a valid approach to finding the right set of parameters. Let's now focus on *how* to do it and just keep the rationale, i.e. *why* we do it in the back of our heads for later.

6.1.2 Setting up the likelihood function (for a single data point)

So we are looking for the parameter set $\vec{\theta}_{ML}$ that maximizes the likelihood $p(D|\vec{\theta})$. The function we set up as our likelihood function must therefore be a function of $\vec{\theta}$, given the observed data:

$$L(\vec{\theta}|D) = p(D|\vec{\theta})$$

In the body height example we assume all data points to be drawn from a normal distribution, or in other words, we assume the random variables X_i to be normally distributed. The likelihood $p(D|\vec{\theta}) = p(X_i|\mu, \sigma^2)$ is therefore given by a normal distribution:

$$\begin{aligned} L_i(\mu, \sigma^2|x_i) &= p(X_i|\mu, \sigma^2) \\ &= \mathcal{N}(\mu, \sigma^2, x_i) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{-(x_i-\mu)^2}{2\sigma^2}} \end{aligned}$$

Note that this is the likelihood function for a single data point x_i under the assumption of normally distributed data. For other distributions, of course the likelihood function has to be set up accordingly.

6.1.3 Joint Likelihood (multiple data points)

If we have a set of independent data points which all stem from identical distributions (i.i.d), we can define the likelihood of the whole data set $L = p(\vec{x}|\vec{\theta})$ as the joint likelihood of all $L_i = p(x_i|\vec{\theta})$:

i.i.d. = independent and identically distributed

$$\begin{aligned} L(\vec{\theta}|\vec{x}) &= p(\vec{x}|\vec{\theta}) \\ &= p(x_1, x_2, \ldots, x_n|\vec{\theta}) \\ &= p(x_1|\vec{\theta}) \cdot p(x_2|\vec{\theta}) \cdot \cdots \cdot p(x_n|\vec{\theta}) \\ &= \prod_i p(x_i|\vec{\theta}) \end{aligned}$$

For normally distributed data this translates to

$$\begin{aligned} L(\mu, \sigma^2|\vec{x}) &= p(\vec{x}|\mu, \sigma^2) \\ &= p(x_1, x_2, \ldots, x_n|\mu, \sigma^2) \\ &= p(x_1|\mu, \sigma^2) \cdot p(x_2|\mu, \sigma^2) \cdot \cdots \cdot p(x_n|\mu, \sigma^2) \\ &= \prod_i \mathcal{N}(\mu, \sigma^2, x_i) \\ &= \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{-(x_i-\mu)^2}{2\sigma^2}} \end{aligned}$$

It should be noted that a joint likelihood distribution is not a probability distribution, since the area under the curve is no longer equal to one.

6.1.4 Deriving the ML Estimators

With our likelihood function set up - either for a single data point or for a whole data set - we can now go on to maximize it. In general, we find the maximum of a function by finding the zeros of its first derivative.

6.1.5 Normal Distributions: Deriving μ_{ML}

Let's go through the process for μ_{ML} from a normal distribution, starting with the joint likelihood that we just set up. We notice right away that we have to derive a product, which is something we really don't like to do. To resolve this inconvenience, we open up our bag of tricks and pull out: The logarithm. We logarithmize the likelihood function. The resulting function is called *log-likelihood function* and its extrema are in the exact same places as those in the original likelihood function. Since at this point we are solely interested in the maxima, we don't care that basically everything else about the function has changed. Here's the derivation step by step:

To be more precise, we use the natural logarithm ln. Despite what you may have learned in high school, the natural logarithm is often denoted as log, which we adopt in this script. Ergo: Read "log", think "ln".

$$
\begin{aligned}
LL(\mu, \sigma^2|\vec{x}) &= \log\left(\prod_{i=1}^{N}\left(\frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right)\right) \\
&\Leftrightarrow \sum_{i=1}^{N}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right) \\
&\Leftrightarrow \sum_{i=1}^{N}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)+\sum_{i=1}^{N}\log\left(e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right) \\
&\Leftrightarrow N\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)+\sum_{i=1}^{N}\frac{-(x_i-\mu)^2}{2\sigma^2} \\
&\Leftrightarrow -N\log\left(\sqrt{2\pi\sigma^2}\right)+\frac{1}{2\sigma^2}\sum_{i=1}^{N}-(x_i-\mu)^2
\end{aligned}
$$

To find μ_{ML} we derive the above equation with respect to μ and equate to zero, then we isolate μ_{ML}:

$$
\begin{aligned}
0 &= \frac{\delta}{\delta\mu}\left(-N\log\left(\sqrt{2\pi\sigma^2}\right)+\frac{1}{2\sigma^2}\sum_{i=1}^{N}-(x_i-\mu_{ML})^2\right) \\
0 &= \frac{\delta}{\delta\mu}\left(\frac{1}{2\sigma^2}\sum_{i=1}^{N}-(x_i-\mu_{ML})^2\right) \\
0 &= K\cdot\frac{\delta}{\delta\mu}\sum_{i=1}^{N}(x_i-\mu_{ML})^2 \text{ with } K=-\frac{1}{2\sigma^2} \\
0 &= K\cdot\sum_{i=1}^{N}2(x_i-\mu_{ML}) \\
\Rightarrow 0 &= \sum_{i=1}^{N}2(x_i-\mu_{ML}) \\
\Leftrightarrow \sum_{i=1}^{N}x_i &= \sum_{i=1}^{N}\mu_{ML} \\
\Leftrightarrow \sum_{i=1}^{N}x_i &= N\mu_{ML} \\
\Leftrightarrow \frac{\sum_{i=1}^{N}x_i}{N} &= \mu_{ML}
\end{aligned}
$$

This is the maximum likelihood estimator μ_{ML} that we've been looking for. What does μ_{ML} tell us? Tells us, that a normally distributed population with $\mu_{ML} = \frac{\sum_i x_i}{N}$ has the highest likelihood of being the population from which we drew the sample. This may sound trivial, but it is a matter of principle. Now that you know how to derive a maximum likelihood estimator, you can do the same for other distributions and parameters. To this end, let's quickly review what we did here:

1. Set up the the *likelihood function* (for a single data point): The likelihood is the probability distribution of our data, given the parameters. Use the density (or mass) function corresponding to $p(D|\vec{\theta})$.

2. Set up the joint likelihood function: The joint likelihood is the product of individual likelihood functions for each measured value. The data points need to be independent. *(This step obviously only applies if you are given more than one data point.)*
3. Logarithmizing the likelihood function: The *log-likelihood function* has the same extrema and is easier to work with.
4. Deriving the log-likelihood function with respect to the parameter in question. Fun fact: This derivative is also called the *score function*.
5. Equating the score function with zero in order to find the maximum: This is our ML estimator.

6.1.6 Normal Distributions: σ^2_{ML} and bias

The same principle is used to derive a formula for σ^2_{ML}, we get

$$\sigma^2_{ML} = \frac{\sum_i (x_i - \mu_{ML})^2}{N}.$$

However, this estimator is biased. The general expression for the bias of an estimator $\hat{\theta}$ is

$$\text{Bias}_\theta[\hat{\theta}] = \mathbb{E}_\theta[\hat{\theta}] - \theta_{\text{true}} = \mathbb{E}_\theta[\hat{\theta} - \theta_{\text{true}}].$$

The hat ($\hat{}$) is the most common indicator for an estimator. In the maximum likelihood approach, we use $_{ML}$ instead.

Biased means, there is a *systematic* deviation between the estimator $\hat{\theta}$ and the true value θ_{true} in the population. To be more precise, the *expected value* for $\hat{\theta}$, i.e. the average $\hat{\theta}$ from indefinitely many samples of size $n < \infty$, does not match the true parameter θ_{true}. In the case of σ^2_{ML} we therefore note:

$$\mathbb{E}[\sigma^2_{ML}] \neq \sigma^2_{\text{true}}$$

An example: We draw a couple of samples, each containing 10 measured values, from an indefinitely large population. For each of these samples we calculate μ_{ML} and σ^2_{ML}. The μ_{ML}s that we get will sometimes be a little above the population mean, sometimes a little below. Averaged over all samples, these differences disappear: If we take indefinitely many samples, their average will be exactly the population mean:

$$\mathbb{E}[\mu_{ML}] = \mu_{\text{true}}$$

μ_{ML} is therefore unbiased. With σ^2_{ML}, the story is a little different. No matter how many samples we take and average, some deviation from the actual population variance will always remain. The amount of this deviation depends on the sample size:

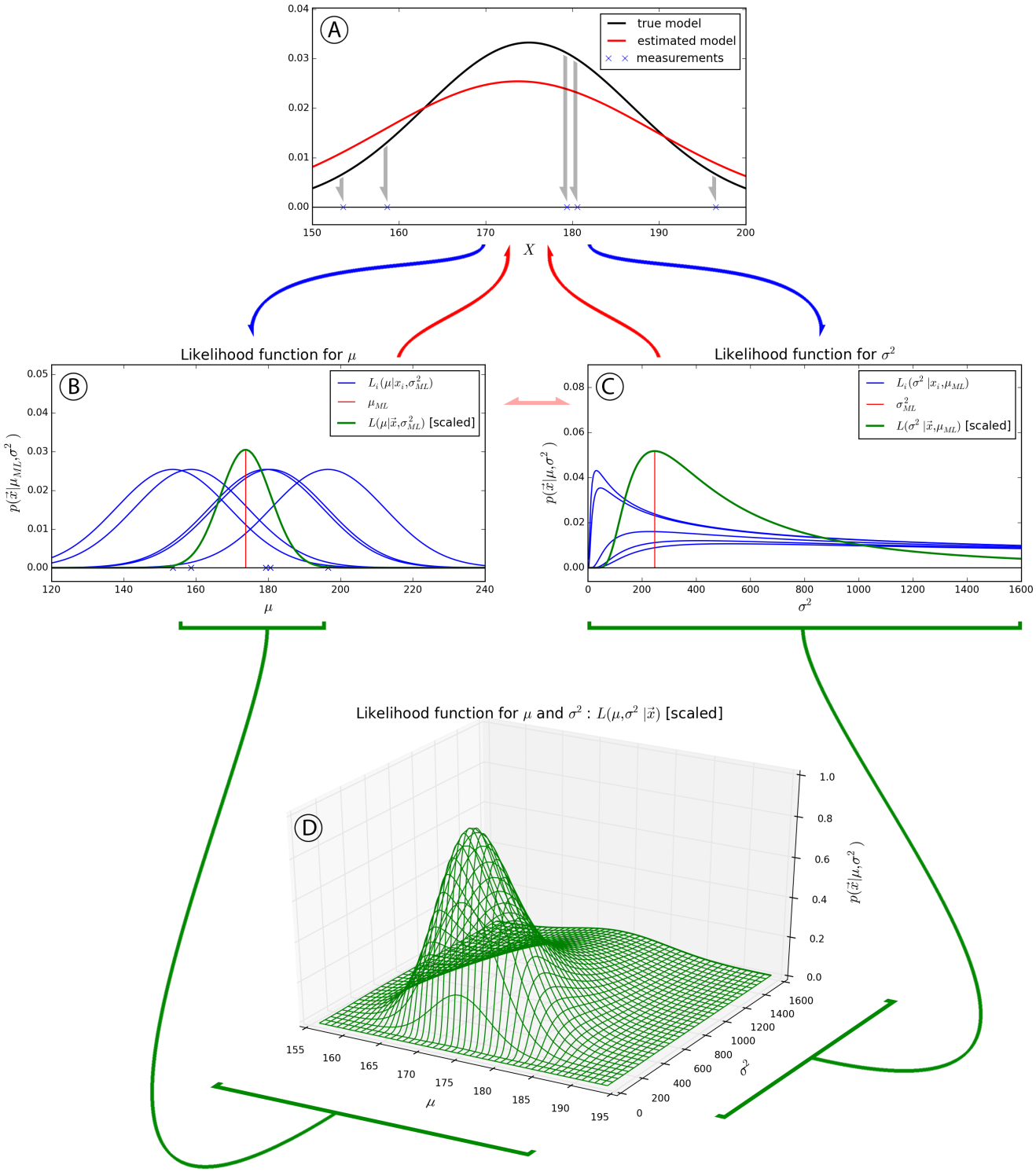
$$\mathbb{E}\left[\sigma^2_{ML} \cdot \frac{N}{N-1}\right] = \sigma^2_{\text{true}}$$

This means we can correct for the bias, using the factor $\frac{N}{N-1}$:

$$\hat{\sigma}^2_{\text{unbiased}} = \sigma^2_{ML} \cdot \frac{N}{N-1}$$

6.1.7 Overview

The following figure, roughly based on the body height example, illustrates the main concepts of this chapter:



Let's go through this figure step by step. The true distribution of our random variable X is the black function in figure **A**. From this we took five samples x_i (blue crosses):

$$\begin{aligned} x_1 &\approx 153.6 \\ x_2 &\approx 196.5 \\ x_3 &\approx 158.6 \\ x_4 &\approx 179.3 \\ x_5 &\approx 180.6 \end{aligned}$$

In figures **B** and **C**, we interpreted the normal distribution as a function of μ, (i.e. $L_i(\mu|x_i, \sigma^2)$, figure B) and of σ^2 (i.e. $L_i(\sigma^2|x_i, \mu)$, figure C), respectively. In both cases, we see the individual likelihood functions in blue, and their product - the joint likelihood - in green . The ML estimators

$$\begin{aligned} \mu_{ML} &\approx 173.7 \text{ and} \\ \sigma^2_{ML} &\approx 246.4 \end{aligned}$$

are indicated as red lines. These ML estimators are then plugged into our model, resulting in the estimated model (red curve in A).

The joint likelihood distributions (green) in B, C and D are scaled for better visibility - do not make any interpretations regarding their height.

Note that there is some mutual dependency between the plots in B and C: The width of the individual likelihood curves in B is determined by σ^2_{ML}, and by extension, so is the width and scale of the joint likelihood. The places of the likelihood functions' maxima in B - including μ_{ML} - however, are independent of σ^2. In figure C it's a little different: The shapes and maxima of the individual likelihood functions - and by extension those of the joint likelihood function - depend on μ. Shown in B is the likelihood of μ, given $\sigma^2 = \sigma^2_{ML}$, C shows the likelihood of σ^2, given $\mu = \mu_{ML}$.

From the above it should become clear that the likelihood function for a normal distribution is actually a two-dimensional function, with one dimension for μ and one for σ^2, as shown in figure **D**. Figures B and C are thus just two orthogonal slices from the two-dimensional likelihood function $L(\mu, \sigma^2|x_i)$.

Figure D should also help to understand the switch in the placement of the variables in

$$L_i(\mu, \sigma^2|x_i) = p(X_i|\mu, \sigma^2).$$

$p(X_i|\mu, \sigma^2)$ is simply the "output" of the function $L_i(\mu, \sigma^2|x_i)$, much like in $y = f(x)$, y is the "output" of $f(x)$.

In general, the likelihood function for a model with m parameters is an m-dimensional function of all model parameters, whose shape is determined by the data.

6.2 Bivariate data: The linear regression

6.2.1 Transition from univariate data

Let's now turn to bivariate datasets and model fitting as presented in the introduction chapter. The rationale of the maximum likelihood approach still applies and we still have to set up a (joint) likelihood function before deriving ML estimators for the parameters. These are now called $\vec{w}$ instead of $\vec{\theta}$, since they act as *weights* on the basis functions. The data is still generally denoted D, we therefore denote the likelihood as

Read the introduction chapter again if you don't recall what a basis function is.

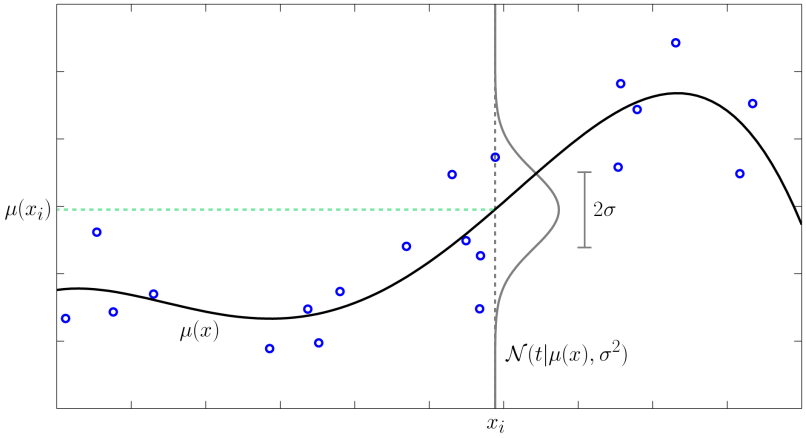
$L(\vec{w}|D) = p(D|\vec{w})$. The measured data is no longer comprised of single values but of pairs of values (x_i, t_i). The ordinate is thus denoted t for *target*. Our model shall later predict a t-value from a given x-value.

6.2.2 The underlying model

In the univariate case, the data points were drawn from a normal distribution, which extended along the x-axis and had some mean μ. In the bivariate case we assume that the x-values are error-free, so we only consider normally distributed noise along the t-axis, i.e. we have a normal distribution extending parallel to the t-axis for every place along the x-axis. The mean $\mu(x)$ of this normal distribution is now not a single value, but a function of x.

In this course we will only consider models that assume normally distributed noise. Other distributions of the noise are possible but will not play a role for bivariate data sets.

The function $\mu(x)$ results from the properties of whatever it is that we're measuring. This could be a sine wave, a simple straight line or a superposition of multiple functions, etc.



The following two expressions for the or underlying model are equivalent:

$$\begin{aligned} t(x) &= \mu(x) + \mathcal{N}(t|\mu = 0, \sigma^2) \\ t(x) &= \mathcal{N}(t|\mu(x), \sigma^2) \end{aligned}$$

We define the underlying model as being generative of the data. This means that what we call the underlying model is not defined as the deterministic function $\mu(x)$, but as the (normal) density function that has $\mu(x)$ at its center and some constant variance σ^2.

6.2.3 Setting up our model

In the univariate case we were looking for a single value μ_{ML}, which was to get as close as possible to the actual, yet unknown population mean μ_{true}. By analogy, in the bivariate case we want to find a function $y(x)$, which comes as close as possible to the unknown $\mu(x)$. To this end, we define a flexible function $y(\vec{w}, x)$, which is a composite of weighted basis functions. In case of polynomials as basis functions, it looks like this:

In the bivariate case we are - at least in this course - no longer really interested in the variance σ^2 and will thus focus on approximating $\mu(x)$.

$$y = w_0 + w_1 \cdot x + w_2 \cdot x^2 + \ldots + w_m \cdot x^m$$

Or, put a little differently:

We use n for the number of data points with control variable i, and m for the number of basis functions with control variable j.

$$y(\vec{w}, x) = \sum_j (w_j \cdot x^j)$$

In order not to have to restrict ourselves to polynomials as basis functions, we denote the mth basis function as $\varphi_m(x)$:

$$y = w_0 + w_1 \cdot \varphi_1(x) + w_2 \cdot \varphi_2(x) + ... + w_m \cdot \varphi_m(x)$$

Or, denoted the other way:

$$y(\vec{w}, x) = \sum_j (w_j \cdot \varphi_j(x))$$

Since $\varphi_m(x)$ is only a placeholder, we need to define the basis functions separately. If, for instance, we choose polynomials as basis functions, we define $\varphi_m(x) = x^m$. Independently of our choice of basis functions, $\varphi_0(x)$ is commonly defined as 1, so that w_0 is always the parameter that can shift the whole function alongside the t-axis. Our *model* is accordingly defined as

$$t(x) = \mathcal{N}(t|y(\vec{w}, x), \sigma^2) \text{ with } y(\vec{w}, x) = \sum_j w_j \varphi_j(x).$$

It is called a linear model since it is linear in w, i.e. it is a linear combination of basis functions φ_n, which can be linear or non-linear. The method of finding the best weights for a linear model - which you are about to learn - is called *linear regression*.

6.2.4 Setting up the likelihood function (bivariate)

In analogy to the univariate case, we are now looking for a suited expression for the likelihood. For an individual value t_i, this is given by

$$\begin{aligned} L_i(\vec{w}|D) &= p(D|\vec{w}) \\ &= p(t_i|x_i, \vec{w}, \sigma^2) \\ &= \mathcal{N}(t_i|y(\vec{w}, x_i), \sigma^2) \end{aligned}$$

If we have a set of data points that are i.i.d., we can again use the joint likelihood, i.e. the product of all probabilities of the individual data points (see 6.1.2). This can be written as

$$\begin{aligned} L(\vec{w}|D) &= p(\vec{t}|\vec{x}, \vec{w}, \sigma^2) \\ &= \prod_i \mathcal{N}(t_i|y(\vec{w}, x_i), \sigma^2). \end{aligned}$$

6.2.5 Maximizing the likelihood: The method of least squares

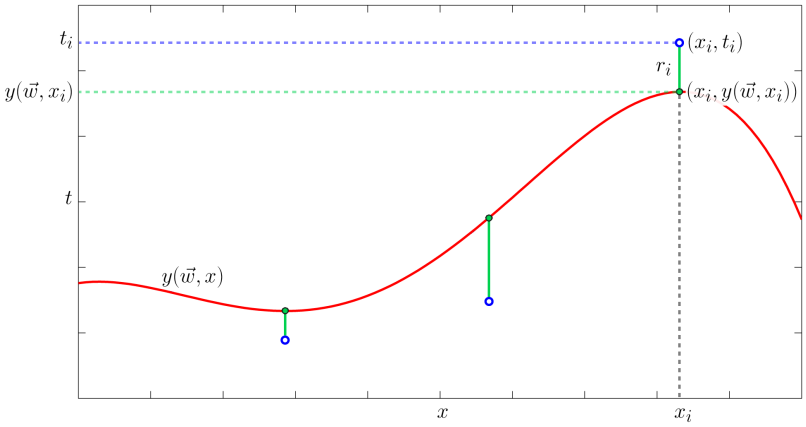
Once again we want to maximize the likelihood, i.e. find the set of model parameters $\vec{w}$ for which the likelihood function peaks, given the data. And again we have a product of individual probabilities, which we want to derive and equate with zero. And still we feel somewhat uncomfortable deriving products. So naturally, we're using the same old trick, logarithmizing the likelihood function. After some transformations we get

$$LL = \ln p(\vec{t}|\vec{x}, \vec{w}, \sigma^2) = -\frac{1}{2\sigma} \sum_i (t_i - y(\vec{w}, x_i))^2 + \frac{N}{2} \ln \frac{1}{\sigma} - \frac{N}{2} \ln(2\pi).$$

Since the last two summands in this equation don't contain any weights w_i, they are irrelevant in the search for the extrema, as is the factor $\frac{1}{2\sigma}$. We're left with

$$-\sum_i (t_i - y(\vec{w}, x_i))^2,$$

which is the negative sum of all squared errors, i.e. the distances between the data points t_i and the model prediction $y(\vec{w}, x_i)$. We can see that in order to maximize the likelihood we have to find the weights for which the sum of squared errors reaches its minimum. This is called the *method of least squares*.



Let us illustrate said method with the help of this figure. In red we see our estimated model $y(\vec{w}, x)$, the blue dots are the measured pairs of values (x_i, t_i). Green dots are the pairs of values $(x_i, y(\vec{w}, x_i))$, i.e. the models predictions for the places of measurement x_i. The gaps between the green and blue dots are called residuals, denoted as r_i:

$$r_i = t_i - y(\vec{w}, x_i)$$

In order to minimize the sum of squared errors $S = \sum_i r_i^2$, we set the gradient to zero:

$$\begin{aligned} 0 &= S\frac{\delta}{\delta w_j} \\ &= 2\sum_i r_i\left(\frac{\delta}{\delta w_j} \cdot r_i\right) = 0 \text{ with } j = 0 \ldots m (1) \end{aligned}$$

And set up a linear system of equations (LSE). We can skip the factor 2 since we set the sum to zero:

$$0 = \sum_i \left(\frac{\delta r_i}{\delta w_j} \cdot r_i\right) \text{ with } j = 0 \ldots m$$

We thus have an LSE with one row per basis function j. $\frac{\delta r_i}{\delta w_j}$ is the derivative of the residuals with respect to w_j (for the ith data point). With this at hand we can set up an LSE and get numeric values for our weights w_j. We'll use this approach to calculate our way through the model fitting process in the section 6.3.

Remember, m is the number of basis functions and i is the running index for the data points and goes from one to n.

6.2.6 The Moore-Penrose-pseudoinverse

Before we look at a practical example though, we'll have a look at an alternative approach for the model fitting process. We can show that the method of minimizing the sum of squared errors, as described in section 6.2.5, is mathematically equivalent to the following notation:

How this is done is demonstrated in the lecture. It might be added to the script at a later point.

$$\vec{w_{ML}} = \phi \dagger \vec{t}$$

In this, $\phi\dagger$ is the Moore-Penrose-pseudoinverse of the design matrix ϕ:

$$\phi\dagger = (\phi T \phi)^{-1} \phi T$$

The design matrix ϕ, in turn, is a matrix of size $n \times m$, in which m is the number of basis functions and n is the number of data points:

$$\phi = \begin{pmatrix} \varphi_0(x_1) & \varphi_1(x_1) & \dots & \varphi_{m-1}(x_1) \\ \varphi_0(x_2) & \varphi_1(x_2) & \dots & \varphi_{m-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \dots & \varphi_{m-1}(x_n) \end{pmatrix}$$

The procedure goes as follows:

1. Set up the design matrix ϕ, using the basis functions φ_i and all x_i
2. Create the Moore-Penrose-pseudoinverse $\phi\dagger$ of the design matrix ϕ
3. Calculate the weights $\vec{w_{ML}} = \phi \dagger \vec{t}$ (with $\vec{t}$ being the vector containing all t_i)

This form is, for instance, what we usually use in Python and Matlab to calculate the weights.

The pen and paper method on the other hand is basically never used in the real world. It's just in the script to help you understand what the pseudoinverse does, rather than this being a "black box method".

6.3 Examples of model fitting (bivariate)

We now have everything we need to fit a model to a given set of data. We can either choose to use an LSE as shown in section 6.2.5 (best suited for pen and paper), or to use the design matrix and its Moore-Penrose-pseudoinverse as shown in section 6.2.6. While the latter is certainly of higher practical relevance, it won't hurt your understanding of the process to go through it manually at least once. Thus, we go through both routines step by step.

6.3.1 The pen and paper method

First in line is the pen and paper method, using an LSE to find the least squared solution. In order for this not to get out of hand, we choose a model with polynomial basis functions up to the order of two and fit it to a dataset consisting of only three data points, which were randomly chosen.

Model: $y(\vec{w}, x) = w_0 + w_1 \cdot x + w_2 \cdot x^2$
Data: $P_1(1|3.2); P_2(2|2.2); P_3(3|3.0); P_4(4|3.4)$

And as a reminder:

Residuals: $r_i = t_i - y(\vec{w}, x_i)$
LSE: $0 = \sum_i (\frac{\delta r_i}{\delta w_j} \cdot r_i)$ with $j = 0 \dots m$

We're gonna go through this in a structured manner and start with the residuals. First, we simply put the x- and t-values into the residuals formula:

$$
\begin{aligned}
r_1 &= 3.2 - (w_0 + w_1 \cdot 1 + w_2 \cdot 1) \\
r_2 &= 2.2 - (w_0 + w_1 \cdot 2 + w_2 \cdot 4) \\
r_3 &= 3.0 - (w_0 + w_1 \cdot 3 + w_2 \cdot 9) \\
r_4 &= 3.4 - (w_0 + w_1 \cdot 4 + w_2 \cdot 16)
\end{aligned}
$$

In the second step, we derive these with respect to the weights w_0, w_1 and w_2:

$\frac{\delta r_i}{\delta w_0}$:	$\frac{\delta r_i}{\delta w_1}$:	$\frac{\delta r_i}{\delta w_2}$:
$r_1' = -1$	$r_1' = -1$	$r_1' = -1$
$r_2' = -1$	$r_2' = -2$	$r_2' = -4$
$r_3' = -1$	$r_3' = -3$	$r_3' = -9$

Now, we successively set up the three rows for our LSE:
For $j = 0$:

$$
\begin{aligned}
0 &= \textstyle\sum_i (\frac{\delta r_i}{\delta w_0} \cdot r_i) \\
0 &= (-1 \cdot (3.2 - w_0 - w_1 - w_2) - 1 \cdot (2.2 - w_0 - 2w_1 - 4w_2) \\
&\quad \text{- } 1 \cdot (3.0 - w_0 - 3w_1 - 9w_2) - 1 \cdot (3.4 - w_0 - 4w_1 - 16w_2)) \\
0 &= -11.8 + 4w_0 + 10w_1 + 30w_2
\end{aligned}
$$

For $j = 1$:

$$
\begin{aligned}
0 &= \textstyle\sum_i (\frac{\delta r_i}{\delta w_1} \cdot r_i) \\
0 &= (-1 \cdot (3.2 - w_0 - w_1 - w_2) - 2 \cdot (2.2 - w_0 - 2w_1 - 4w_2) \\
&\quad \text{- } 3 \cdot (3.0 - w_0 - 3w_1 - 9w_2) - 4 \cdot (3.4 - w_0 - 4w_1 - 16w_2)) \\
0 &= -30.2 + 10w_0 + 30w_1 + 100w_2
\end{aligned}
$$

For $j = 2$:

$$
\begin{aligned}
0 &= \textstyle\sum_i (\frac{\delta r_i}{\delta w_1} \cdot r_i) \\
0 &= (-1 \cdot (3.2 - w_0 - w_1 - w_2) - 4 \cdot (2.2 - w_0 - 2w_1 - 4w_2) \\
&\quad \text{- } 9 \cdot (3.0 - w_0 - 3w_1 - 9w_2) - 16 \cdot (3.4 - w_0 - 4w_1 - 16w_2)) \\
0 &= -93.4 + 30w_0 + 100w_1 + 354w_2
\end{aligned}
$$

And combine these in an LSE:

$$
\left|\begin{array}{lcccc}
11.8 & = & 4w_0 & 10w_1 & 30w_2 \\
30.2 & = & 10w_0 & 30w_1 & 100w_2 \\
93.4 & = & 30w_0 & 100w_1 & 354w_2
\end{array}\right|
$$

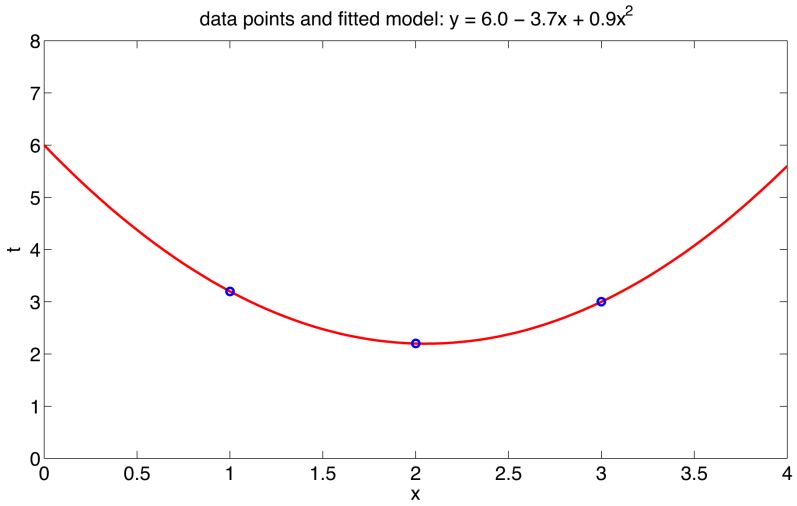
Solving this we get:

$$
\begin{aligned}
w_0 &= 4.35 \\
w_1 &= -1.16 \\
w_2 &= 0.35
\end{aligned}
$$

With this we're done - our fitted model looks like this:

$$y = 4.35 - 1.16x + 0.35x^2$$

ch6fig6.m

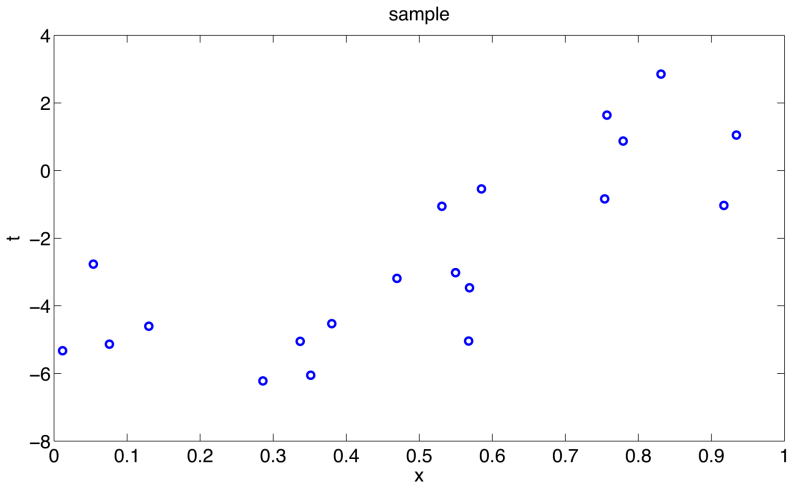


6.3.2 Fitting a model to data using Matlab

We now move over to Matlab. We start out with a sample that we measured earlier on some occasion. The sample consists of 20 pairs of data (x_i, t_i). All x_i are ordered by magnitude and denoted in vector $\vec{x}$, the associated t_is are denoted at the respective place in vector $\vec{t}$.

This section already gives you the whole code. If you want to play with it, you also find it in *ch6fig7modelfit.m*.

```
% #########
% 1) SAMPLE
% #########

x = [0.0119020695012414;0.0539501186666072;0.0758542895630636;...
     0.129906208473730;0.285839018820374;0.337122644398882;...
     0.351659507062997;0.380445846975357;0.469390641058206;...
     0.530797553008973;0.549723608291140;0.567821640725221;...
     0.568823660872193;0.585264091152724;0.753729094278495;...
     0.757200229110721;0.779167230102011;0.830828627896291;...
     0.917193663829810;0.934010684229183;];

t = [-5.32632952704015;-2.76310880554607;-5.13423201074049;...
     -4.60339861803915;-6.22174438426079;-5.05131094597675;...
     -6.05235095574945;-4.52812552605543;-3.18752129221983;...
     -1.05544755218143;-3.02106042467119;-5.04071637244840;...
     -3.46472974628074;-0.544193649813664;-0.836426551757584;...
      1.64338021520258;0.873481787155255;2.85741082659293;...
     -1.02907518849797;1.05032847625816;];

figure(1);                          % plotting
subplot(1,2,1);
plot(x,t,'o','Color','blue','LineWidth',2);
axis([0 1 -8 4]);
xlabel('x');
ylabel('t');
title('sample');
```



Next up we choose the model that we want to fit on the data and set up the corresponding design matrix. In this case we opt for polynomials up to the order of four, ergo for this model:

$$y = w_0 + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$$

The creation of the Moore-Penrose-pseudoinverse and its multiplication with $\vec{t}$ is done by Matlab, using the function *glmfit*. We feed glmfit with the (transposed) design matrix and (the transposed) vector $\vec{t}$:

```
% ################
% 2) MODEL FITTING
% ################

order = 4;                                  % order of highest polynomial
                                            % basis function

d_mat = zeros(order+1,length(x));           % setting up the design matrix
for k = 0:order
    d_mat (k+1,:) = x.^k;
end
d_mat = d_mat(2:end,:);                     % erase row for w_0 from d_mat
                                            % matlab automatically adds
                                            % this row in glmfit again


[w_ML] = glmfit(d_mat',t')                  % estimation of weights
```

In return, we receive the weights $\vec{w_{ML}}$.

```
w_ML =

   -4.4816
    3.1506
  -62.6276
  177.1496
 -115.7299
```

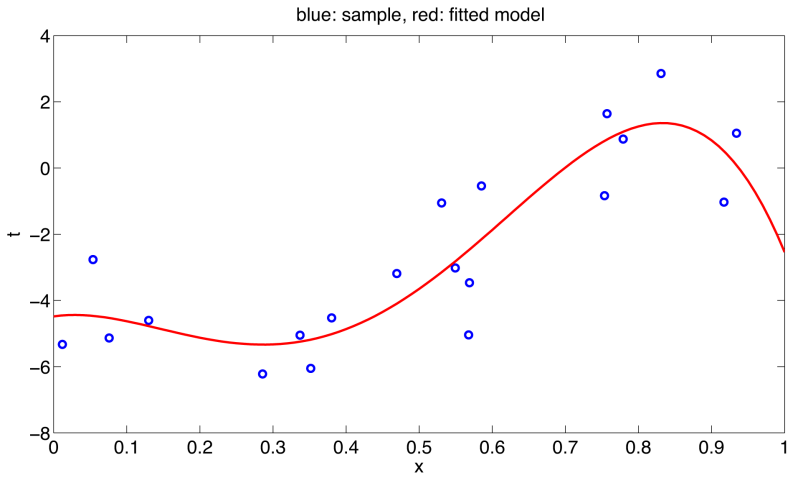
And these are our weights:

$$w_0 = -4.4816$$
$$w_1 = 3.1506$$
$$w_2 = -62.6276$$
$$w_3 = 177.1496$$
$$w_4 = -115.7299$$

With this we're effectively done. However, since we like colorful pictures, we'll just add some code for visualization purposes. For a graphical display of our estimated model we can either manually put the weights into our model, or we can use the function *glmval* to calculate the models predictions. We go with the latter, of course. For a smooth rendering we need a vector that goes from zero to one in small increments, and we call it *xreal*. We utilize this vector *xreal* to set up a new design matrix and feed that into glmval, along with our weights w_ML and the command "identity" (why we put in "identity" is not important at this point).

```
% ########################
% 3) DISPLAY FITTED MODEL
% ########################

xreal = 0:.01:1;                                % needed for...

d_mat_mod = zeros(order+1,length(xreal));       % the design matrix for smooth
for k = 0:order                                 % visual display of fitted mod.
    d_mat_mod (k+1,:) = xreal.^k;
end
d_mat_mod = d_mat_mod(2:end,:);                 % erase row for w_0 from d_m_m
                                                % matlab automatically adds
                                                % this row in glmfit again

% predictions of our model fit

ymodel = glmval(w_ML,d_mat_mod','identity');% predictions of fitted model

figure(1);                                      % plotting
subplot(1,2,2);
plot(x_i,t_i,'o','Color','blue','LineWidth',2);
hold all;
plot(xreal,ymodel,'Color','red','LineWidth',2);
axis([0 1 -8 4]);
xlabel('x');
ylabel('t');
title('blue: sample, red: fitted model');
```

In return we receive the models predictions ($y(\vec{w}, x_i)$) in the vector *ymodel*, which we plot:



7 Basis functions

While in this course, as well as in Bishop's "Pattern Recognition and Machine Learning" most of the principles about model fitting are explained using polynomial models, it is important to know that in real world scenarios other choices of basis functions are a lot more common. This chapter will give you an overview of other possible choices for basis functions. At this point, let's quickly revisit the formula describing our model:

This section and basically all following sections are concerned with models for bivariate datasets. Therefore, when we speak of a model from here on in this script, we always mean a weighted sum of basis functions.

$$y = w_0 + w_1 \cdot \varphi_1(x) + w_2 \cdot \varphi_2(x) + ... + w_m \cdot \varphi_m(x)$$

or

$$y(\vec{w}, x) = \sum_j (w_j \cdot \varphi_j(x)).$$

You may regard this chapter as a little detour. The rest of the script will mostly use polynomial basis functions for illustration purposes again. Nevertheless, this is important.

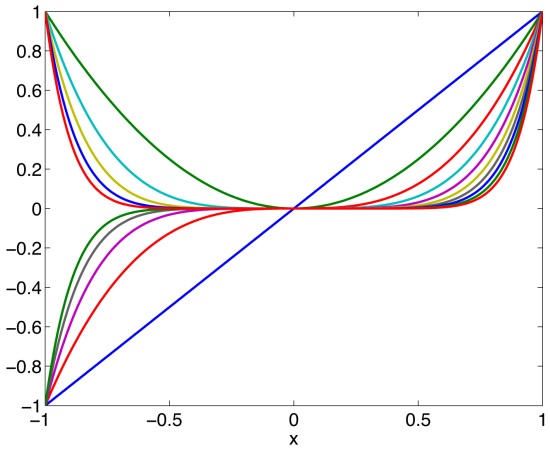
Remember that $\varphi_j(x)$ is a placeholder for our basis functions. We begin our overview of basis functions with polynomials.

7.1 Polynomial basis functions

$$\varphi_j(x) = x^j.$$

This isn't new to you. Here's the first ten polynomials in the range of -1 to 1.

ch7fig1.m



The problem with polynomial basis functions is that they are global, or non-local. This means that the height of the curve at a certain point x is defined by the sum of *all* basis functions that are used in the model at the same time. We prefer local basis functions, i.e. basis functions that influence the model only at a certain range of x and are zero or very close to zero everywhere else.

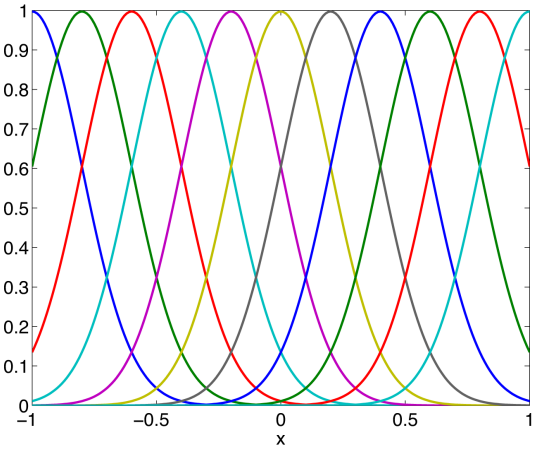
7.2 Gaussian basis functions

Gaussian functions for example are relatively local. They need to overlap in oder to cover the range of x without gaps, but they turn essentially zero at a certain distance from their peak.

$$\varphi_j(x) = e^{-\frac{(x-\mu_j)^2}{2\sigma^2}}$$

Gaussians are especially easy to use, since μ_j and σ control the location and width of the function. If you compare them to the normal distribution function you will find that the normalization coefficient is left out. The normalization coefficient is unimportant, since we multiply the function the weights w_j. Here's what they look like:

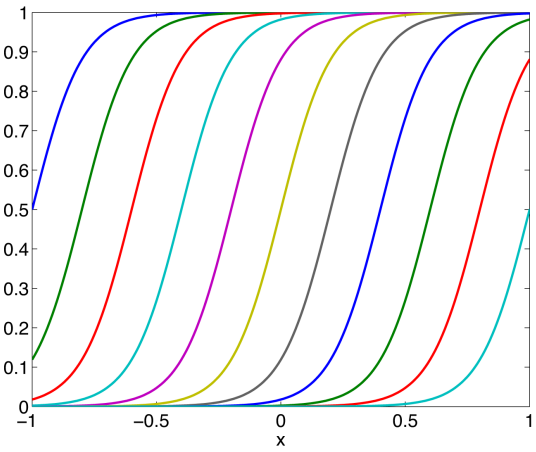
ch7fig2.m



7.3 Sigmoidal basis functions

$$\varphi_j(x) = \frac{1}{1 + e^{-\frac{(x-\mu_j)}{\sigma}}}$$

ch7fig3.m



Like Gaussians they are relatively local, and again μ_j and σ control location and slope.

7.4 Sinusoid basis functions

The criterion for a periodic function goes as follows:

$$f(x) = f(x + nk) \text{ with } k \in N$$

This simply means that the function repeats itself over and over, with some period length n. This criterion is fulfilled by sine functions, typically denoted as

$$f(x) = \sin(2\pi cx + \phi),$$

with c acting as a frequency multiplier and ϕ denoting the phase. If we plug this as a basis function into our model

$$y(\vec{w}, x) = \sum_j (w_j \cdot \varphi_j(x)),$$

we end up with

$$y(\vec{w}, x) = \sum_j (w_j \cdot \sin(2\pi j x + \phi_j)).$$

Note that we replaced the arbitrary frequency multiplier c by j in order to get rid of it as a free parameter of the model. As a consequence, the frequencies that we use are multiples of the base frequency of the sine (1 Hz). Nevertheless, we still have the additional free parameter ϕ_j in this model. Unfortunately, the sine function displayed above is not linear in ϕ_j, thus we cannot simply perform the linear regression (i.e. our regular way to determine model parameters) to determine ϕ_j as we would like to. The way out of this situation is provided by the addition formula for sine and cosine:

$$\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha)$$

Applied to our situation:

$$w_j \sin(2\pi j x + \phi_j) = w_j \sin(2\pi j x)\cos(\phi_j) + w_j \sin(\phi_j)\cos(2\pi j x)$$

This is how we free ϕ_j from its non-linearity dilemma. We can now substitute the terms $\cos(\phi_j)$ by a_j and $\sin(\phi_j)$ by b_j:

$$w_j \sin(2\pi j x + \phi_j) = w_j a_j \sin(2\pi j x) + w_j b_j \cos(2\pi j x)$$

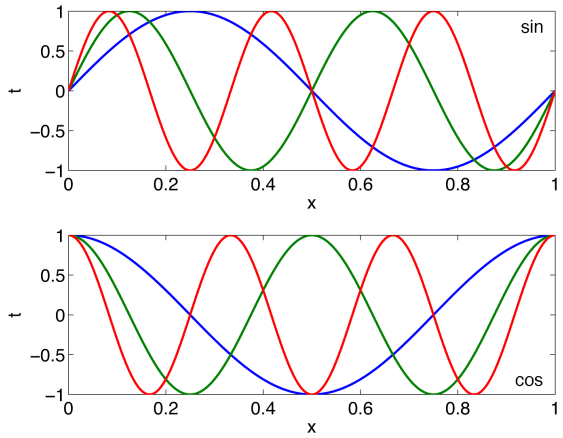
Which leads us to this expression for the model:

$$y(\vec{w}, x) = \sum_j w_j a_j \cdot \sin(2\pi j x) + \sum_j w_j b_j \cdot \cos(2\pi j x)$$

As the last step, we combine $w_j a_j$ to α_j and $w_j b_j$ to β_j:

$$y(\vec{w}, x) = \sum_j \alpha_j \cdot \sin(2\pi j x) + \sum_j \beta_j \cdot \cos(2\pi j x)$$

In this form, the model is linear in α_i and β_i. They can thus be determined via linear regression.



Here you see $\sin(2\pi j x)$ and $\cos(2\pi j x)$ for orders up to $j = 3$. Note that periodic functions are global.

The reason why sinusoids are interesting as basis functions is that, as Joseph Fourier found out, a sum of sinusoids can approximate *any* other function with arbitrary precision. This approximation process is called a *Fourier series*. For more information see Fourier series (Wikipedia).

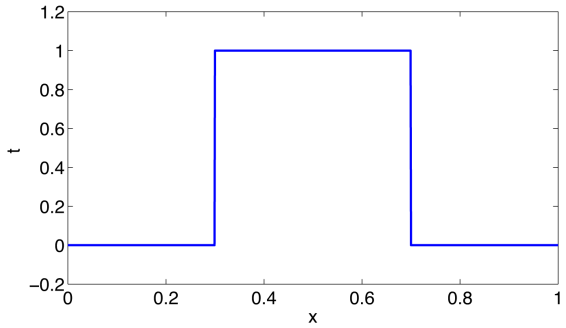
ch7fig4.m

7.5 Bin-based basis functions

Another form of basis functions are bin-based basis functions. They are strictly local, i.e. non-zero only for the range between two bounds x^l and x^r. The simplest bin-based function is a step function:

$$\varphi_j(x) = \begin{cases} 1 & x_j^l \leq x < x_j^r \\ 0 & \text{otherwise} \end{cases} .$$

ch7fig5.m

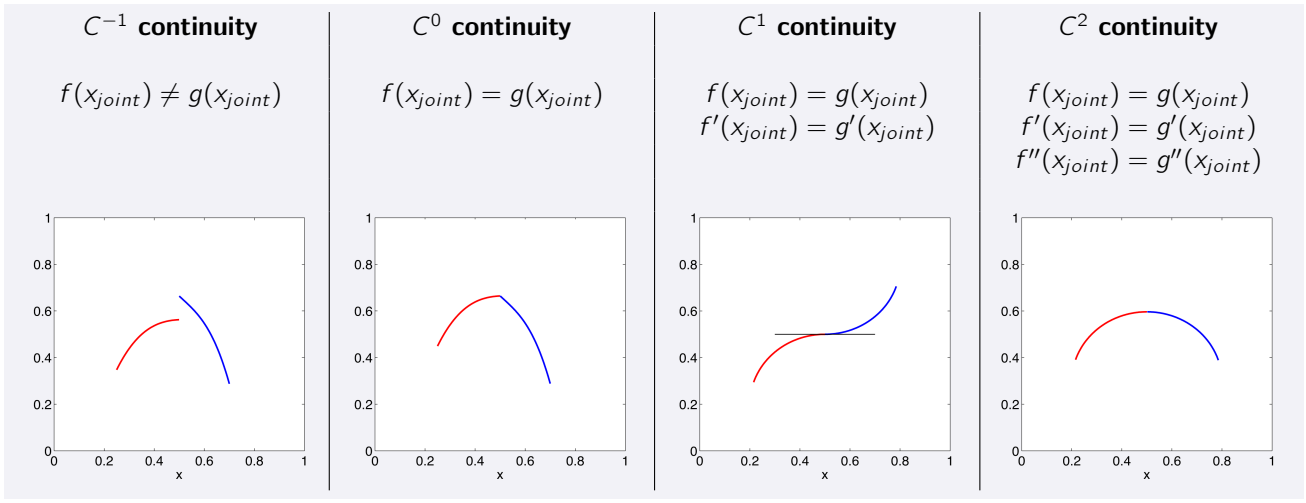


The "bin" in this case would be the range of x for which the function is non-zero. Since this isn't a smooth function, a sum of step functions can't be smooth either. In many or most situations however, a smooth function promises better results. Now, if we want a smooth but strictly local function, we can build a bin-based function from multiple polynomial functions. This is then called a piecewise polynomial function.

7.5.1 Continuity in piecewise polynomials

At the joints of piecewise polynomials, one can observe different levels of smoothness, or *continuity*. Since this principle is pretty straight forward, we'll just go with the overview:

ch7fig6.m



Remember that the first derivative $f'(x)$ gives you the slope of the function f at the point x. If the two functions have the same slope at the joint, they can be considered smooth. The second $f''(x)$ derivative gives us the direction and degree of curvature that the function f has at the point x. As a result, a joint with C^2 continuity is not only smooth, but as smooth as possible.

7.5.2 B-splines

B-splines are a special case of piecewise polynomial functions. The algorithm which helps us produce them is recursive and called the de Boor's algorithm:

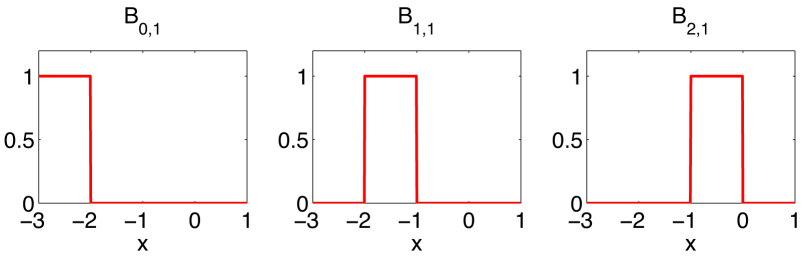
$$B_{k,d}(x) = \left(\frac{x - x_k}{x_{k+d-1} - x_k}\right) B_{k,d-1}(x) + \left(\frac{x_{k+d} - x}{x_{k+d} - x_{k+1}}\right) B_{k+1,d-1}(x)$$

It terminates with

$$B_{k,1}(x) = \begin{cases} 1 & x_k \leq x < x_{k+1} \\ 0 & \text{otherwise} \end{cases}.$$

In the algorithm, d denotes the order of recursion that is used, while k shifts the resulting function along the x-axis. In order to get a grip on this, let's have a look at the resulting functions at different orders d, starting with $d = 1$:

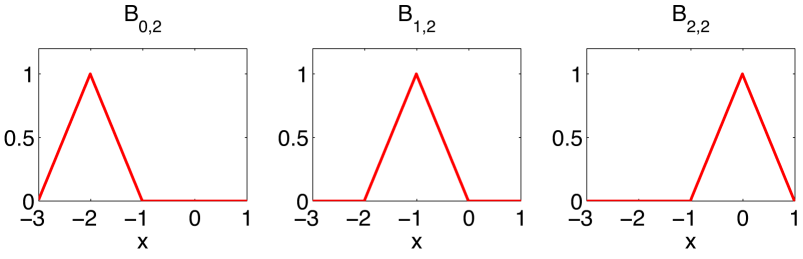
$$B_{0,1}(x) = \begin{cases} 1 & -3 \leq x < -2 \\ 0 & \text{otherwise} \end{cases}$$



As you might have guessed, the joints of a first order spline from the de Boor's algorithm have a C^{-1} continuity.

Next up, $d = 2$:

$$B_{0,2}(x) = \begin{cases} x + 3 & -3 \leq x < -2 \\ -1 - x & -2 \leq x < -1 \\ 0 & \text{otherwise} \end{cases}$$



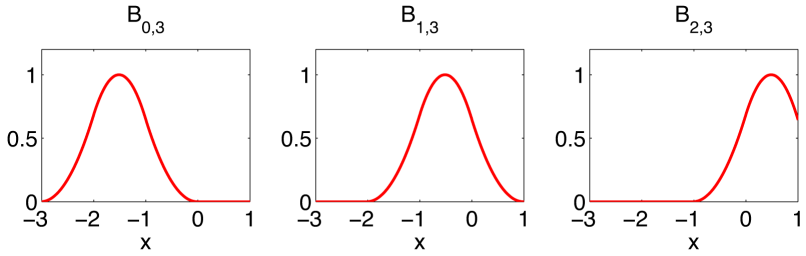
This is an example for a C^0 continuity:

Then $d = 3$:

It is not necessary to learn this algorithm by heart for this lecture, it just serves the purpose of showing you how splines (especially cubic splines) come into being.

ch7fig7.m

$$B_{0,3}(x) = \frac{1}{2}\begin{cases}(x+3)^2 & -3 \le x < -2\\ -2x^2 - 6x - 3 & -2 \le x < -1\\ x^2 & -1 \le x < 0\\ 0 & \text{otherwise}\end{cases}$$

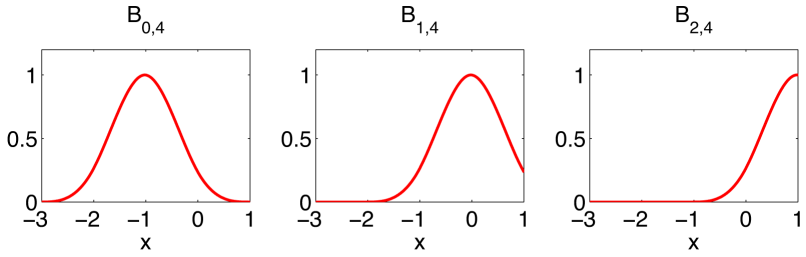


Unsurprisingly, this is a C^1 continuity.

And finally $d = 4$. These are called *cubic b-splines*, since they contain x^3 as the highest order polynom. Cubic b-splines feature C^2 continuity at the joints :

As you see, the piecewise polynomials resulting from the de Boor's algorithm go up one order of continuity for every additional order of the algorithm.

$$B_{0,4}(x) = \frac{1}{6}\begin{cases}(x+3)^3 & -3 \le x < -2\\ -3x^3 - 15x^2 - 21x - 5 & -2 \le x < -1\\ 3x^3 + 3x^2 - e3 + 1 & -1 \le x < 0\\ (1-x)^3 & 0 \le x < 1\\ 0 & \text{otherwise}\end{cases}$$



The de Boor's algorithm is not as arbitrary as it might appear. The functions it creates at the order $d = n+1$ describes the convolution of the function at $d = n$ with the step function at $d = 1$. For example, the triangularly shaped function at $d = 2$ results from a convolution of the step function at $d = 1$ with itself. The bell-shaped function at $d = 3$ results from a convolution of the functions at $d = 2$ and $d = 1$ and so on.

A convolution creates a new function from two original functions. It gives the area overlap of the two original functions while one of them is swept across the other. For more information see Convolution (Wikipedia).

About the properties of b-splines: First off, they are strictly local, i.e. outside of a defined interval they are exactly zero. Therefore, additional data points in a restricted interval of x won't affect the whole curve/ model, as it would be the case with global functions. This, in conjunction with the fact that they are put together from low order polynomials, makes for a low computational demand. In real world scenarios, b-splines are therefore a very popular choice for basis functions.

The first case is actually shown in this animation: click.

8 Generalizability, validation and model selection

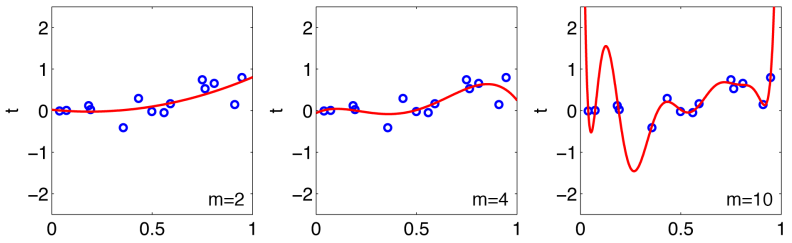
Now that we've learned about the basic model fitting process, and looked into the most popular choices for basis functions, we can dive deeper into the matter, gaining a better understanding of how statistical models are used in the real world. We are on the hunt for the best model we can get for our dataset. Therefore, we need to know what we can change about our model in order to (possibly) improve it, and once we have different models at hand, how we can test which of them is best.

8.1 Training- and validation data

The purpose of a model is to make predictions. In our case, this means predicting a t-value from an x-value, with the model representing our best guess for the functional relation between the two. At the beginning, we are provided a set of measured pairs of x- and t-values, which we use to fit our model. Once the model is fitted, it allows to predict t-values from x-values - so far so good. If we want to test our models predictions, though, we need fresh pairs of x- and t-values to compare with. Since the one dataset is all we get, we should save up a certain fraction of it right from the start for the later validation of our model. In other words, we split up the provided dataset into a *training dataset* and a *validation-* or *testing dataset*. Only the training dataset is then used to fit the model, while the testing dataset remains untouched during that process.

8.2 Number of basis functions, over- and underfitting

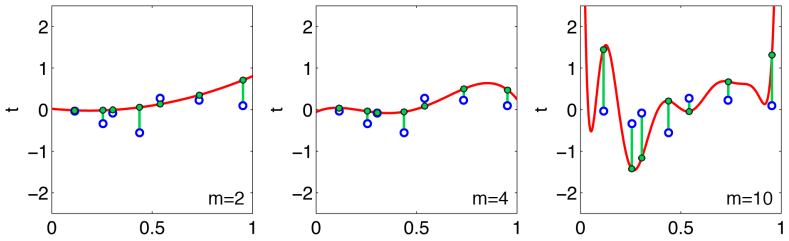
Why do we need to test our model anyway? In the previous chapters, before fitting the model, we freely picked a number of basis functions and then stuck to it. We had no way to know, which number of basis functions would get us closest to the underlying model:



m is the order of polynomials, the number of basis functions is $m + 1$, accordingly.

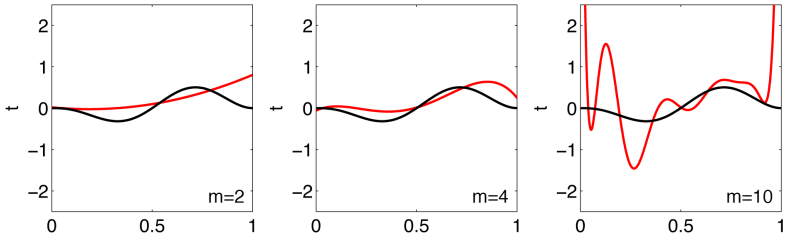
You find the code for the following figures in *ch8fig1.m*.

The key question is, how accurate would the models predictions be when compared to new data, created by the same underlying model? This is also called *generalization performance*. For us, the testing dataset will fill the role of the new data.



The green lines in this figure represent the residuals. Obviously, the polynomial model of order ten does not generalize well. If we take a look at the underlying model, the reason for this becomes obvious:

Remember that in a real world scenario we wouldn't know the underlying model.



The tenth order polynomial model is just far too complex. Instead of finding a fit that describes the underlying structure of the data, the model rather fits the noise pattern that came up just by chance. This behavior is called *overfitting*. The opposite phenomenon - *underfitting* - shows on the second order polynomial model. Here, the complexity of the model is too low for a proper reconstruction of the underlying model.

Granted, this isn't a perfect example, but you get the idea.

8.3 Cross-validation

What we just did was basically already a *cross-validation*: We split up the data into a training- and a testing set, fitted the model on the training set and evaluated its performance on the testing set. Since our models are fitted such that the likelihood is maximized, we coherently compare the models in terms of likelihood. For normally distributed data, the model with the least squared residuals (distance between model prediction $y(\vec{w}, \vec{x})$ and data $\vec{t}$) has the highest likelihood. Consequently we can simply use the sum of squared residuals

$$S = \sum_i (t_i^{test} - y(\vec{w}, x_i^{test}))^2$$

as our measure of goodness of fit.

In the example above,

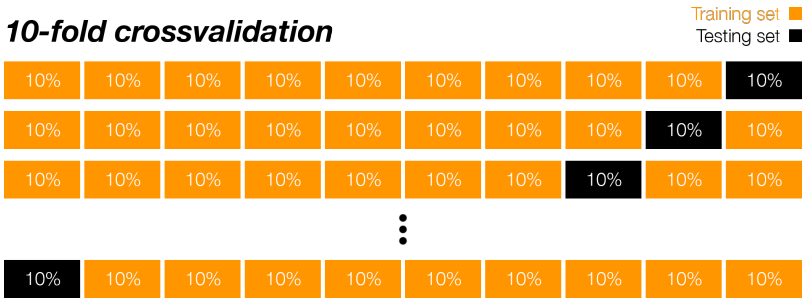
$$S_{m=2} < S_{m=4} > S_{m=10},$$

ergo the polynomial model of order $m = 4$ wins this comparison.

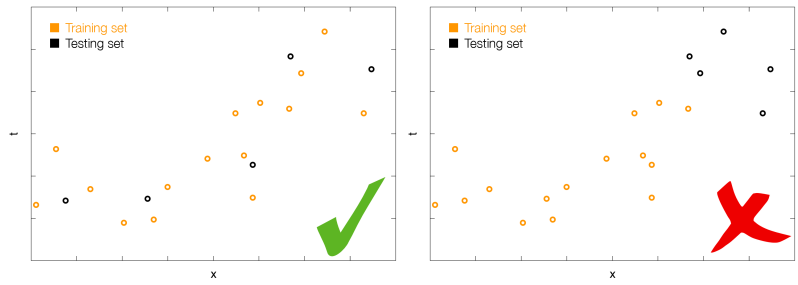
How much of the data we use for training and testing depends on the situation, there is no general rule for this. The testing set usually ranges between 10% and 50% of the complete dataset. Choosing its size is always a compromise: If the testing set is small there is a larger chance that it isn't an accurate representation of the data. It could be comprised of mostly outliers, possibly favoring models that are far from ideal. If the testing set is very large, though, we fit the model on a smaller training dataset, thus forfeiting precision of the model. However, there is a way to use the data more efficiently:

In an *n-fold cross-validation* we chop the (complete) dataset into *n* little chunks, all of the same size. We then go through the process of fitting (or "training") and testing *n* times, so that each chunk will be used as the testing set once, while the remaining chunks serve as the training set. Let's illustrate this with this scheme of a 10-fold cross-validation:

Accordingly, the example above would be a 1-fold cross-validation with a 2 : 1 split.



Note that the data pairs comprising one chunk of data are randomly chosen from the set. We do not chop the set apart in terms of intervals on x:

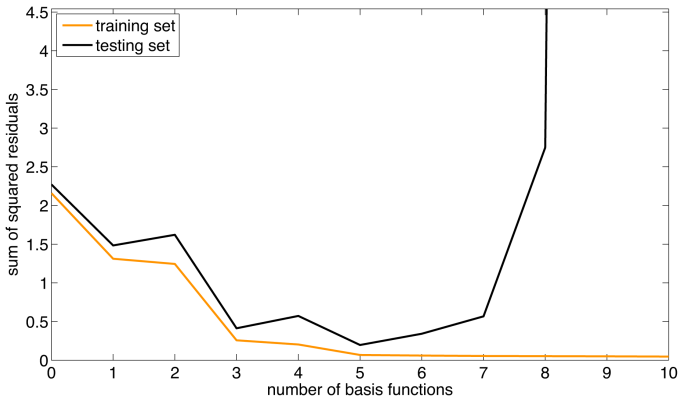


To pick a winner of an n-fold cross-validation (or "select a model"), we can use different approaches. The most common one would be to average for each model the *n* estimates of model performance and pick the model with the lowest value, or rather its number of basis functions. Once we know which number of basis functions we want to use, we don't need further testing. We can therefore run the model fitting process once again, using the complete dataset as the training set, for greater precision.

As an alterantive we could, for instance, pick the model with the lowest median.

The following figure shows a typical course of the sum of squared errors on training and testing dataset over different numbers of basis functions:

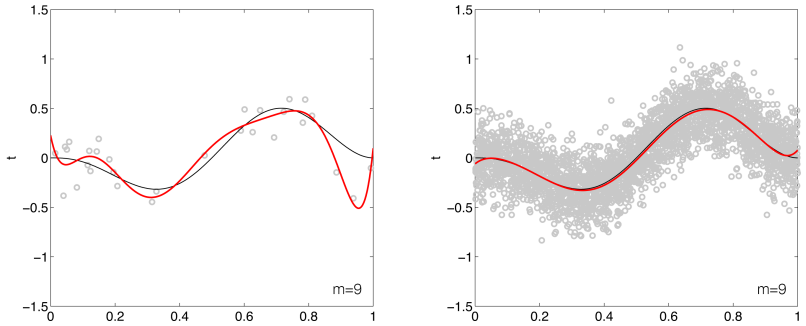
ch8fig4.m



The training set error decreases monotonically with increasing number of basis functions. This is quite plausible: Say we have two models, one of which has all the same basis functions as the other plus one more (as it is the case with polynomial models). Then it is not possible that it performs weaker than the "smaller" model: Even if the additional basis function can, for some reason, not help to improve the fit, its weight will just be determined zero, so it can't hurt the performance (on the training set!). The testing set error, though *can* be systematically weaker for a higher number of basis functions. This is the case when the models starts overfitting, as this one does approximately from $m = 8$ (one could argue the overfitting starts at 6 or 7 already).

At what number of basis function a model actually starts overfitting depends on the complexity of the underlying model, the variance of the noise and the amount of data available for the model fit. The more data we have, the higher the model complexity that we can achieve without running into overfitting:

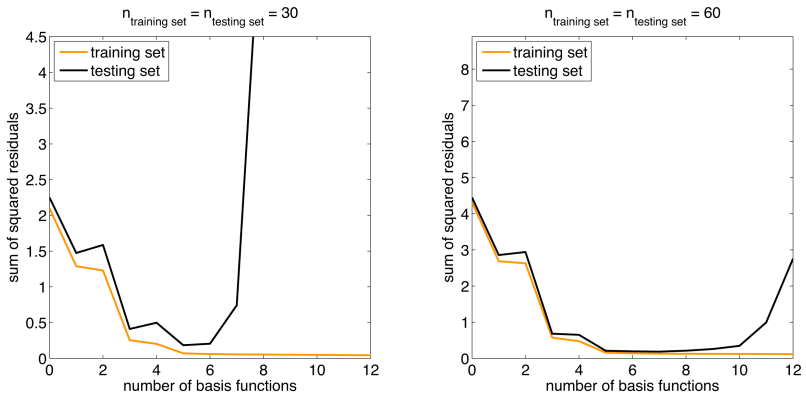
ch8fig5.m



Let's see how doubling the dataset size affects the plot from earlier:

The sum of squared residuals of equally well performing model scales linearly with the size of the datasets (because it is a sum, not a mean).

You find the code for these figures in *ch8fig6.m*.



Note that even though the model is less prone to overfitting with more data, this doesn't necessarily have a big effect on the ideal number of basis functions. In this example it lies in the range of 5 to 6 in case of fewer data, and in the range of 5 to 7 in case of more data. With an eye on the computational demand we would be likely to pick 5 as the number to go with in both cases.

9 Model selection: Nested models, likelihood ratio and deviance

After the cross-validation approach, the likelihood ratio test is the next method in model selection that we are going to introduce. For this, we first need to define the construct of nesting- and nested models as well as saturated models.

9.1 Nested and saturated models

As you know, a linear model is defined as a weighted sum of basis functions ($y = \sum_i w_i \varphi_i$). Coming from there, a *nested model* is defined as the weighted sum of any subset of the basis functions constituting the *nesting model*. Put the other way round, you can define an arbitrary model as a nesting model and create nested models by axing basis functions from it - any basis function and any number of basis functions can be canceled. In the following example, three different nested models were derived from the same nesting model. Beware that this is not a complete list, other nested models are possible.

$$\begin{aligned} \text{nesting model:} \quad & y = w_0 + w_1\varphi_1 + w_2\varphi_2 + w_3\varphi_3 \\ \text{nested model:} \quad & y = w_0 + w_1\varphi_1 + w_3 \cdot \varphi_3 \\ \text{nested model:} \quad & y = w_0 + w_1\varphi_1 + w_2 \cdot \varphi_2 \\ \text{nested model:} \quad & y = w_0 + w_2\varphi_2 \end{aligned}$$

This means that, for instance, a polynomial model of order 3 is a nested model of an order 5 polynomial model, because the nesting model ($order = 5$) contains all the basis functions from the nested model, plus additional basis functions. By extension, all polynomial models can be regarded as being nested models to the same nesting model.

A *saturated model* can be thought of a model that has as many parameters (basis functions) as there are data points. Consequently, the resulting function will go through every single point in the dataset, all residuals are exactly zero.

9.2 Likelihood ratios

Remember that in a cross-validation we compare models by comparing their likelihoods with the simple rule that higher is better (or comparing the squared error with lower being better - remember that the negative of the squared error is the crucial term in the likelihood function). Now we want to take the comparison of likelihoods one step further by comparing the likelihood ratios of two models y_1 and y_2 (with the corresponding weights $\vec{w_1}$ and $\vec{w_2}$):

$$k = \frac{L(\vec{x}, \vec{w_1})}{L(\vec{x}, \vec{w_2})}.$$

Let's discuss now, why the likelihood ratio is a favorable measure over a simple subtraction. We know from chapter 6.2.5 that the likelihood scales with the negative of the sum of squared residuals (or squared errors) $S = \sum_i (t_i - y(\vec{w}, x_i))^2$. This, in turn, is dependent on the number of data pairs and the scaling of the data (e.g. which unit is used for data notation). Now since the absolute value of the likelihood

For easier notation, we denote our data by $\vec{x}$ (disregarding the fact that it is comprised of x- and t-values) and the likelihood as a function of the data and the model parameters $\vec{w}$: $L(\vec{x}, \vec{w})$. This function will return a single numerical value when fed with both the data $\vec{x}$ and the model parameters $\vec{w}$.

depends on the amount of data points as well as the scaling of the data, the likelihood is not a normalized value - the area under the likelihood function is not equal to 1 and gets smaller for more data points. An important conclusion from this is that the likelihood is not the same as a probability. The likelihood value on its own can therefore not be interpreted. The difference of two likelihood values for two models on the same dataset will show us which one of the two performs better on that dataset, it will however not tell us the extent of that difference. That is why we use the likelihood ratio. It is a *scale-invariant* measure, i.e. it will give us a relative comparison of the two values.

This is very important to remember: **likelihood** $\neq$ **probability**.

Different example: In a performance test, participant A was five points better than participant B. In a second test, B was ten points better than A. Who is better overall? This absolute measure doesn't help us, it could be the first test only gave 10 points in total, while the second gave 100. Therefore we need a relative measure to compare the results.

A first constraint to the likelihood ratio test was already mentioned: Since the likelihood scales with the amount and scaling of data used, we can only use the likelihood ratio as a measure of comparison if in both cases the same data was used.

Let us now compare two likelihood values for two different models y_1 and y_2 using the likelihood ratio

$$k = \frac{L(\vec{x}, \vec{w_1})}{L(\vec{x}, \vec{w_2})}.$$

We further define the *log*-likelihood ratio c as

$$\begin{aligned} c &= \log(k) \\ &= \log \frac{L(\vec{x}, \vec{w_1})}{L(\vec{x}, \vec{w_2})} \\ &= \log L(\vec{x}, \vec{w_1}) - \log L(\vec{x}, \vec{w_2}). \end{aligned}$$

But why do we need a relative comparison of the two models, anyway? A relative comparison allows us to find out if the difference between the two models is statistically significant. We can test the difference between models for statistical significance with the χ^2 test, as introduced in section 5.2.3. To this end we introduce the term *deviance*, which is defined as

$$D(w_1, w_2) = 2c = 2\log(k) = 2\log \frac{L(\vec{x}, \vec{w_1})}{L(\vec{x}, \vec{w_2})} \sim \chi^2$$

$$\Leftrightarrow c = \log L(\vec{x}, \vec{w_1}) - \log L(\vec{x}, \vec{w_2}) \sim \frac{\chi^2}{2}$$

with

$$LL = -\frac{1}{2\sigma} \sum_i (t_i - y(\vec{w}, x_i))^2 + \frac{N}{2} \ln \frac{1}{\sigma} - \frac{N}{2} \ln(2\pi).$$

This is the log-likelihood formula we already saw in 6.2.5.

$\log L = LL$

Since the deviance D is approximately χ^2-distributed, we can use it as the test-statistic for a χ^2 test. The number of degrees of freedom for the χ^2-distribution is the number of parameters that are different between the two models. If $D > \chi^2_{crit}$, then the difference between the two models is statistically significant (favoring model y_1 over model y_2), i.e. we are $\geq 95\%$ certain that y_2 is less accurate than y_1. To make this process even neater, we can, instead of calculating $D(\vec{w_1}, \vec{w_2})$ and comparing it to χ^2_{crit}, simply calculate $c = \log L(\vec{x}, \vec{w_1}) - \log L(\vec{x}, \vec{w_2})$ and compare it to $c_{crit} = \frac{1}{2}\chi^2_{crit}$. The following table contains χ^2_{crit} and c_{crit} values for $p = .05$:

We will use the deviance mostly in the context of nested models, where the degrees of freedom for the χ^2-distribution are easy to determine.

d.f.	χ^2_{crit}	c_{crit}
1	3.84	1.92
2	5.99	3.00
3	7.81	3.91
4	9.49	4.74
5	11.07	5.54
6	12.59	6.30
7	14.07	7.03
8	15.51	7.75
9	16.92	8.46
10	18.31	9.15

The likelihood ratio test can be conducted using either training or testing data. If conducted on the testing data it is actually very close to the cross-validation approach introduced in chapter 8. Using the log-likelihood ratio, we add the possibility of testing the difference between two models for statistical significance.

9.3 Saturated models

The next concept we will introduce is that of *saturated models*. A saturated model is not an actual model based on basis functions, it is rather a placeholder for a perfect model the likelihood function. The likelihood of a saturated model is defined as

$$L(\vec{x}, \vec{w_{sat}}) = L(\vec{x}, \vec{x}).$$

It basically means that instead of feeding model predictions into the likelihood function, we feed the data itself in, causing the residuals to be exactly zero. We can now use the deviance to estimate an absolute value for the goodness of fit of a particular model, by comparing it to a saturated model. In other words, the deviance

$$D(\vec{x}, \vec{w}) = 2 \log \frac{L(\vec{x}, \vec{x})}{L(\vec{x}, \vec{w})}$$

can be interpreted as a measure of distance between a particular model and the saturated model.

Alternatively, the saturated model can be thought of a model with as many basis functions as there are data points, which would also result in the residuals being exactly zero.

10 Bias-variance-decomposition

In order to get a better understanding of the relationship between the expected error and the complexity of our model, we will now perform a mathematical analysis of the error term. We call this the *Bias-Variance-Decomposition*.

We dive right in with a quick recap of section 6.2: There, we defined the underlying model as a function $\mu(x)$, with the data being a combination of said underlying model and a noise term:

If the noise is normally distributed, $\epsilon = \mathcal{N}(t|\mu = 0, \sigma^2)$

$$t(x) = \mu(x) + \epsilon,$$

which, for each individual data point t_i translates to

$$t_i(x_i) = \mu(x_i) + \epsilon.$$

We further defined our model as $y(\vec{w}, x)$, with y(,x_i) being the value/ prediction at a certain place x_i. The squared error between model predictions and data is thus

$$\sum_i (t_i - y(\vec{w}, x_i))^2.$$

The mean of this squared error over all data points in the set is defined as $L2$:

$$\begin{aligned} L2 &= \frac{1}{N}\sum_{i=1}^{N}(t_i - y(\vec{w}, x_i))^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2 \mid \text{shorter notation} \end{aligned}$$

We now take the expectation of this $L2$ and perform a series of transformations on it:

The transformation itself is not included in the script. Please refer to the lecture slides for that.

$$\begin{aligned} \mathbb{E}[L2] &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[(t_i - y_i)^2] \\ &= \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{E}[\epsilon^2] + \mathbb{E}(\mu_i - \mathbb{E}[y_i])^2] + \mathbb{E}[(\mathbb{E}[y_i] - y_i)^2]\right) \end{aligned}$$

This translates to:

$$\text{expected squared error} = \text{noise} + \text{bias}^2 + \text{variance}$$

Let's explore what this means. First off, note that in this Bias-Variance-Decomposition we are analyzing the *expected* squared error. For instance, imagine you have a dataset of size $N = 50$ coming from a certain underlying model and perform a linear regression in order to fit a model on this data (minimizing the squared error ($L2$)). You repeat this for indefinitely many datasets, each of size $N = 50$, each drawn from the same underlying model, and take the average/ mean of these indefinitely many models. This is the *expectation* of the model: $\mathbb{E}[y_i]$.

To be precise, it's the expectation of the models prediction at a certain point x_i, but let's not be splitting hairs here.

Now, the systematic mismatch between said expected model and the underlying model constitutes the bias term. The bias goes down with increasing model complexity. Since the expected model is practically a model based on an infinite number of data points, this is in line with our notion that larger datasets allow for more complex models.

The (squared) difference between the expected model, i.e. the mean of all fits, and each individual fit constitutes the variance term. It describes the variability of the model across different data sets. The variance goes up with increasing model complexity. The phenomenon responsible for this is overfitting.

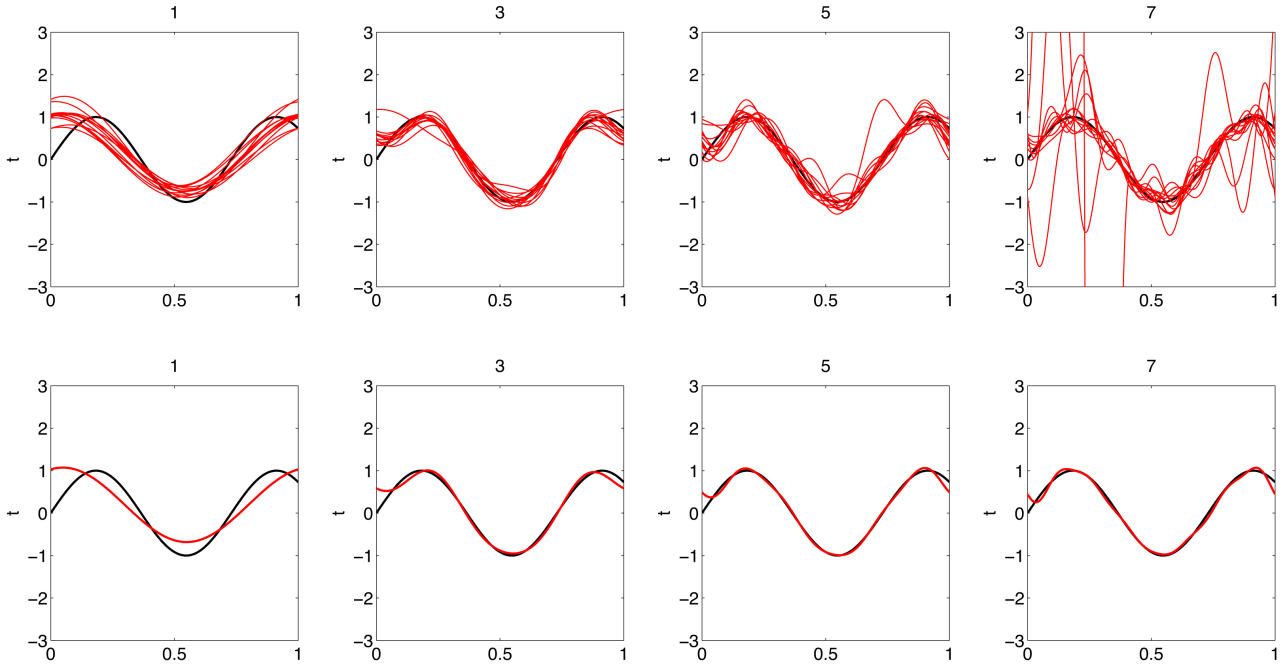
Last but not least, the noise term is independent of the model. It creates a lower bound for the expected squared error ($L2$). In other words, some discrepancy between our fit and the underlying model will always remain.

In conclusion, with increased model complexity the bias decreases while the variance increases. When making decisions about the complexity of our model, we always aim for the best compromise between bias and variance. A high variance indicates overfitting, while a high bias indicates underfitting.

This relationship is illustrated in the following figure. Model complexity increases from left to right, while the amount of data stays the same for all plots. Top row: In red you see 12 individual fits per plot and thus the increase in variability (due to overfitting) with higher model. complexity. Bottom row: Average of 2000 individual fits per plot, giving you an idea of a decreasing bias with increasing model complexity. For an infinite number of fits, this procedure would result in the *expected model* $\mathbb{E}[y]$.

Corresponding Matlab file: *ch10fig1.m*

This is an example for a model using sines as basis functions.

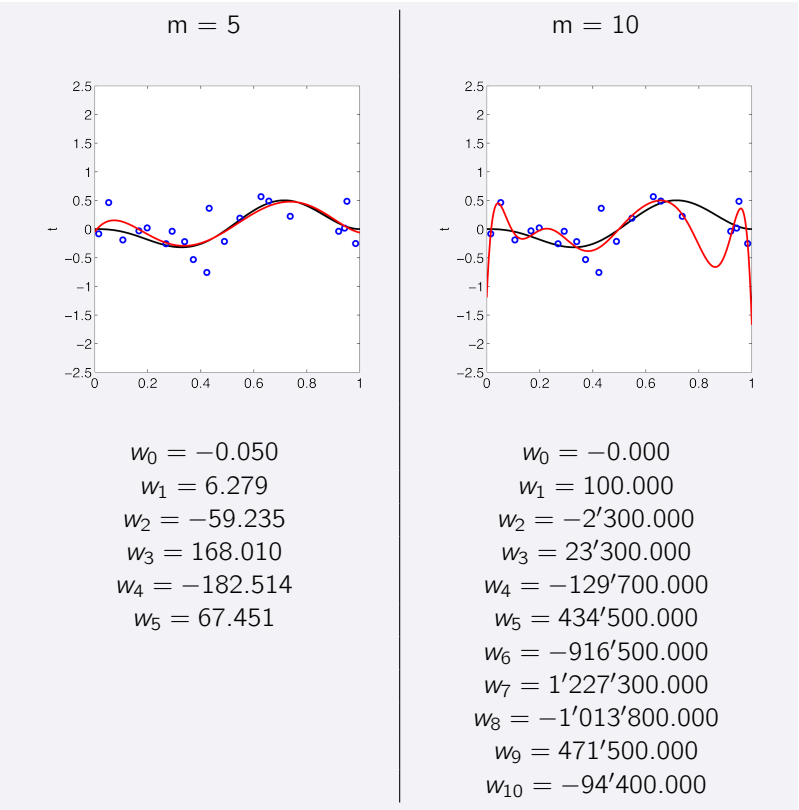


11 Regularization

By now we have a decent understanding of overfitting. So far, in order to prevent it, we reduced the model complexity. Now we will learn about an alternative approach, which can yield better results than the simple reduction of model complexity: *Regularization*.

Let's take a closer look at an overfitting model versus a non-overfitting model by comparison of the respective weights:

ch11fig1.m



$w_0 = -0.050$
$w_1 = 6.279$
$w_2 = -59.235$
$w_3 = 168.010$
$w_4 = -182.514$
$w_5 = 67.451$

$w_0 = -0.000$
$w_1 = 100.000$
$w_2 = -2'300.000$
$w_3 = 23'300.000$
$w_4 = -129'700.000$
$w_5 = 434'500.000$
$w_6 = -916'500.000$
$w_7 = 1'227'300.000$
$w_8 = -1'013'800.000$
$w_9 = 471'500.000$
$w_{10} = -94'400.000$

As you can see, the weights of the overfitting model are a lot higher than those of the non-overfitting one. This results in the high slopes and high number of changes in direction that come along with the overfitting. When using regularization, we reduce the magnitude of the slopes in the fitted model by keeping the model weights low. To this end, we put the absolute values of the weights as additional parameters into the error function:

$$E_D(w) + \lambda E_w(w)$$

$E_D(w)$ = **E**rror between model prediction and **d**ata, dependent on the model **w**eights.
$E_W(w)$ = "**E**rrorterm" of the **w**eights, dependent on the **w**eights.

We thus consider the absolute value of the weights an "error" that shall be minimized in the linear regression, while at the same time minimizing the squared error between model prediction and data (i.e. the squared residuals). Since the two terms are antagonists of one another, this means finding a compromise. λ weights the second error term, i.e. the amount of regularization that is applied. High values for λ result in stronger regularization and smaller model weights, while low values for λ attenuate the influence of the regularization.

Let's replace the abstract error terms with something a little more concrete:

$$\sum_i (t_i - \vec{w}^T \phi(x_i))^2 + \frac{\lambda}{2} \vec{w}^T \vec{w}$$

Since ϕ is the design matrix, $w^T \phi(x_i)$ is simply an expression for the model's predictions, otherwise known as $y(\vec{w}, x_i)$. $\frac{\lambda}{2} w^T w$ is called the *regulation coefficient* and results precisely in $\frac{\lambda}{2} \sum_j w_j^2$. We can therefore write this term in a more familiar form:

$$\sum_i (t_i - y(\vec{w}, x_i))^2 + \frac{\lambda}{2} \sum_j w_j^2$$

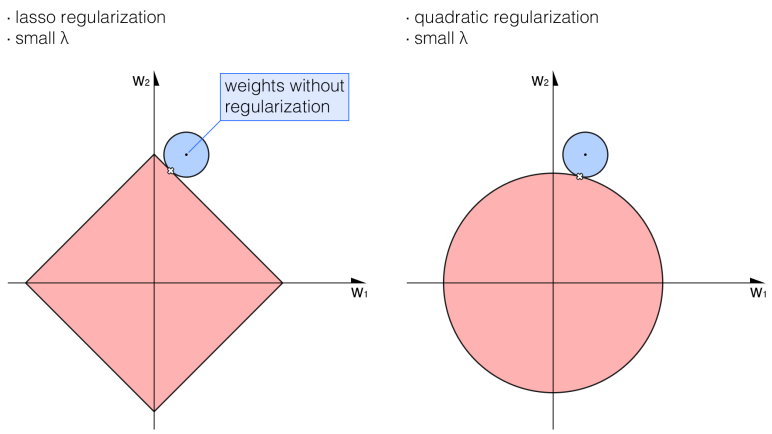
This is the error function we want to minimize in the linear regression order to get our model weights. It hasn't changed from the standard error function from chapter 6, with the exception of the added regularization coefficient. In this case, the regularization coefficient is quadratic, consequently punishing large weights harder than small weights. We can include the possibility for different types of regularization by using a more general regularization coefficient:

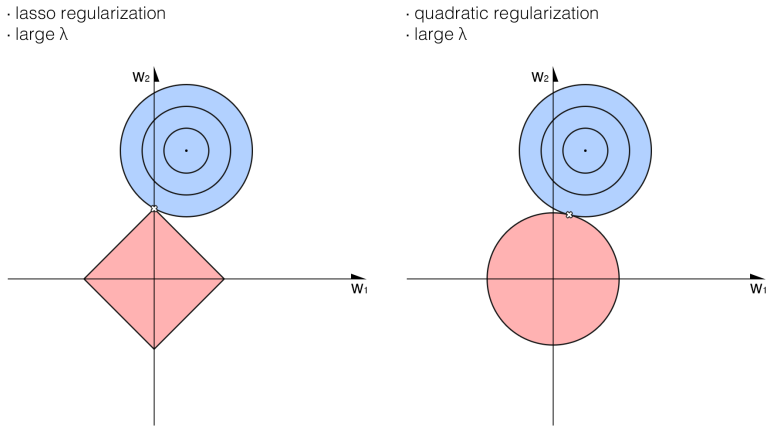
$$\sum_i (t_i - y(\vec{w}, x_i))^2 + \frac{\lambda}{2} \sum_j |w_j|^q$$

We now have two parameters that control the regularization: λ and q. While λ controls the strength of the regularization, q controls its type.

Let's go through a couple of different types of regularization: With $q = 1$ the regularization is called *lasso* regularization. Since simply the sum of all weights is minimized, the lasso type punishes small weights just as hard as large weights. As a result, many of the small weights are being cut to zero, while only the large weights survive.

The following figures show the differences between lasso and quadratic regularization, for small and large values of λ. For illustration purposes, the figures correspond to a simple model with only two basis functions and thus two weights. Without any regularization, w_1 is rather small, while w_2 is rather large, as indicated by the little dot.

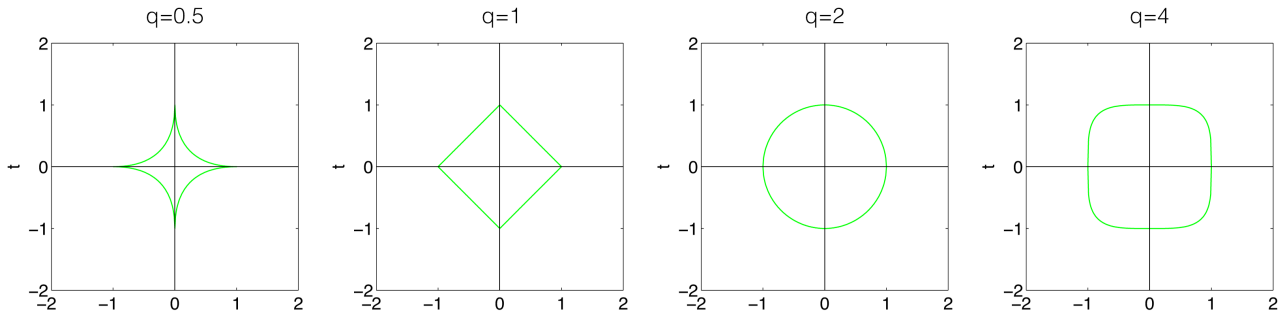




In blue you see an illustration of the first term in the error function, i.e. $E_D(w)$. The black circles in and around the blue areas are iso-lines indicating combinations of w_1 and w_2 with equal mean squared errors (MSE) between the resulting model and the data. The red shape indicates the second term of the error function (i.e. $E_W(w)$). The best compromise between the two terms is reached where the blue and red figures meet. For increasing values of λ the regularization gets stronger, thus the red shape shrinks, dragging the weights further towards zero. The difference between the two types of regularization shown here become obvious in the figures: The lasso regularization quickly drags the smaller weights (here: w_1) to zero, thus creating a sparser model with fewer basis functions in total, while the quadratic regularizer punishes larger weights (here: w_2) stronger.

In reality, the iso-lines would hardly ever go in perfect circles.

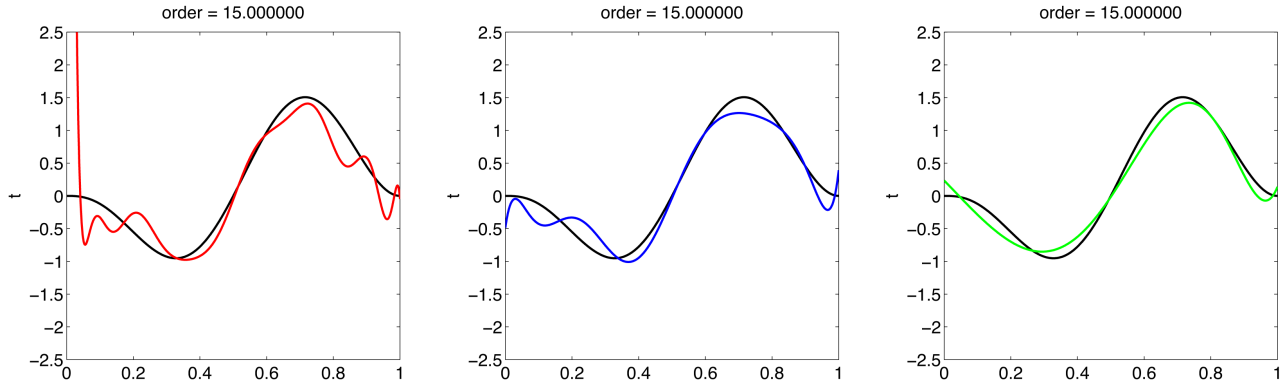
In addition to the two types of regularization, corresponding to $q = 1$ and $q = 2$, there are more values for q of interest. The following figure shows the shape of the regularizers for $q = 0.5$, $q = 1$, $q = 2$ and $q = 4$:



The regularization behavior can be inferred from the shape of these figures: For $q = 0.5$ the model will become even sparser, while for $q = 4$ large weights are being punished even harder than for $q = 2$.

Here you see a practical example of model fitting (polynomial model of order 15). Left plot is without regularization, middle plot with a soft quadratic regularization and right plot with a stronger quadratic regularization:

ch11fig4.m



Even though we use a polynomial model of a relatively high order for this task, the effective model complexity is limited by the regularization term. In other words, the task of finding the appropriate model complexity is not avoided by adding the regularization, instead it is shifted to finding an appropriate value for λ.

12 Bayesian regression

First of all, let's take a step back and look at what exactly we have been calculating in this course up to now - the maximum likelihood of model parameters. The likelihood functions we set up in order to approach this were always distribution functions of the likelihood over the value(s) of the model parameter(s) in question. But why did we consider the likelihood $p(D|\vec{w})$? Let's call up Bayes' theorem again:

$$\overbrace{p(\vec{w}|D)}^{\text{posterior}} = \frac{\overbrace{p(D|\vec{w})}^{\text{likelihood}}\overbrace{p(\vec{w})}^{\text{prior}}}{\underbrace{p(D)}_{\text{normalization term}}}$$

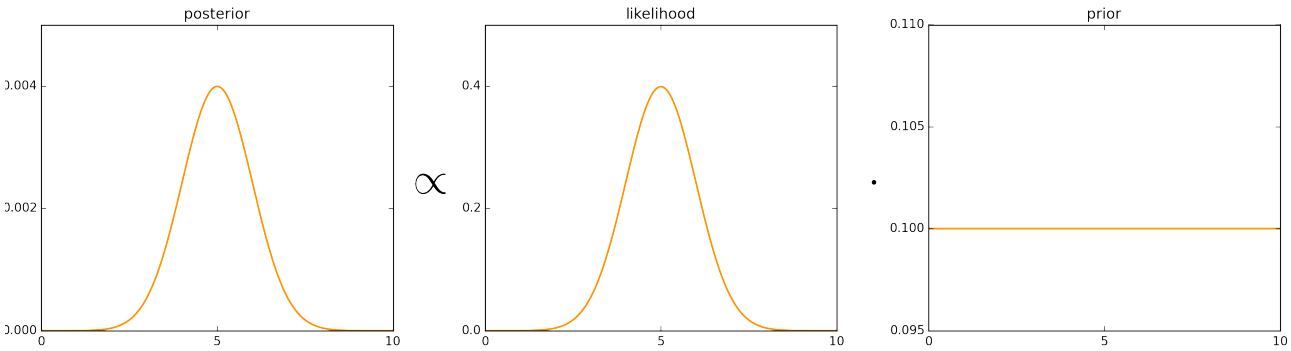
Ultimately, we are interested in the posterior $p(\vec{w}|D)$, or rather the maximum of the posterior distribution (called *MAP* estimate, from *maximum a posteriori*), i.e. the most likely set of model parameters given the data we observed. This posterior distribution is the (normalized) product of the likelihood distribution and the prior distribution. If we now assume the prior distribution to be flat (i.e. constant, non-informative), the posterior distribution will be a normalized version of the likelihood distribution (and the MAP estimates will be exactly the same as the ML estimates). Making this assumption of flat priors is precisely what we've been doing all along when we used the maximum likelihood estimates in place of the MAP estimates.

$p(D)$ does not contain - and is therefore constant with respect to - the set of model parameters $\vec{w}$, it acts only as a scaling factor. Rearranging $p(D)$ to $\int_{-\infty}^{\infty} p(D|\vec{w})p(\vec{w})dw$ shows that this scaling factor is equal to the area under the curve of the likelihood $\times$ prior distribution and therefore normalizes the area under the posterior distribution to 1.

We can formalize flat priors for discrete and continuous distributions:

$$p(\lambda) = \frac{1}{K} \text{ for discrete } \lambda\text{, K-nomial}$$

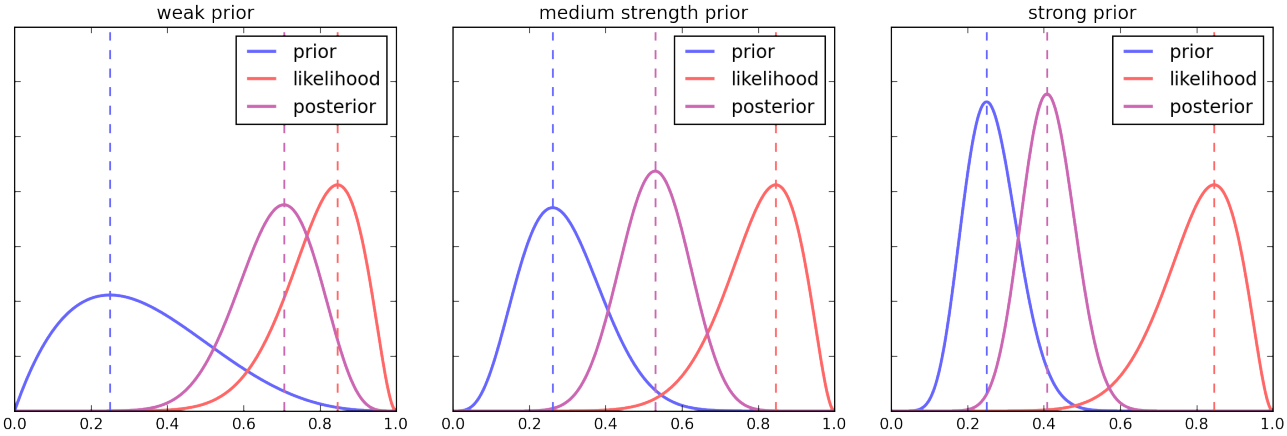
$$p(\lambda) = \frac{1}{b-a} \text{ for continuous } \lambda \in [a, b]\text{, real and bounded}$$



ch12fig1.py

To use flat priors is the norm in cases where we have no *prior* knowledge with respect to the variable in question (or do not wish to take it into consideration). If however we do have some prior knowledge (or expectation/ assumption/ belief) regarding the variable in question, we can take it into account by choosing a non-flat prior distribution $p(\vec{w})$.

With the distributions parameters we determine its shape, which implicitly also works as a weight and can therefore be used to express how certain we are about the prior assumption.



ch12fig2.py

As we can see, a bayesian regression approach (i.e. the taking into account a non-flat prior) acts as a restriction with regard to the parameter range, similar to regularization and other means to control model complexity. Note that in contrast to a regularization, where the model parameters are dragged towards zero, a bayesian prior will always drag the model parameters towards the maximum of the prior distribution. In a regularization, the added term acts as a prior centered around zero, which makes sense, since the idea of a regularization is based on the prior assumption that a complex model (i.e. one with high weights) will fit the specific noise pattern and is therefore unlikely to be a good approximation of the underlying model.

With regards to the choice of the distribution family for the prior we are basically free to choose whatever we deem ideal to formalize our prior knowledge or belief. Nevertheless, in many cases it is of advantage to choose the prior distribution from a family such that a multiplication with the likelihood will result in a posterior distribution of the same family as the prior distribution. When this is the case, prior and posterior are called *conjugate distributions*, with the prior being a *conjugate prior* to the likelihood function.

If you are interested, you can find a table that lists likelihood distributions and their conjugate priors in the Wikipedia article about conjugate priors.

12.1 Conjugate prior and MAP estimates for Bernoulli

Let's start with an example: We conduct an experiment in which we toss a thumbtack n times ($x_1, ..., x_n =: X$) and want to determine the probability for it to fall on its head (heads encoded as 1, tails = 0). We assume the data to be i.i.d. and set up the likelihood function using the Bornoulli distribution

This example is borrowed from Professor Jäkels course *Probabilistic Modelling of Perception and Cognition*

$$p(X|\mu) = \prod_{i=1}^{n} \underbrace{\mu^{x_i}(1-\mu)^{1-x_i}}_{\text{Bernoulli}} = \mu^h(1-\mu)^t$$

with h being the number of heads and t being the number of tails ($n = h + t$). The maximum likelihood estimate that we can derive

from this expression is only equal to the MAP estimate if we assume a flat prior. We may however have a certain prior belief (e.g. that this thumbtack has a μ of roughly .5), or data from previous experiments with the same thumbtack and want to take this into account for our MAP estimate. Let's recall Bayes' theorem and update it with the variable names X and μ:

$$p(\mu|X) = \frac{p(X|\mu)p(\mu)}{p(X)}$$

To express our prior belief, we opt for a conjugate prior $p(\mu)$. In the case of a Bernoulli distributed likelihood function the conjugate prior will come from a beta-distribution:

$$p(\mu) = \frac{1}{B(\alpha,\beta)} \cdot \mu^{\alpha-1}(1-\mu)^{\beta-1}$$

Since $\frac{1}{B(\alpha,\beta)}$ is independent of μ we may drop it for the purpose of setting up the posterior distribution:

$$p(\mu|X) \propto \underbrace{\mu^h(1-\mu)^t}_{p(X|\mu)} \cdot \underbrace{\mu^{\alpha-1}(1-\mu)^{\beta-1}}_{p(\mu)} = \underbrace{\mu^{h+\alpha-1}(1-\mu)^{t+\beta-1}}_{\text{new } \beta\text{-distribution}}$$

And here's the full posterior distribution:

$$p(\mu|X) = \frac{\mu^{\alpha_n-1}(1-\mu)^{\beta_n-1}}{B(\alpha_n,\beta_n)} \text{ with } \alpha_n = h+\alpha, \beta_n = t+\beta$$

We can now go on to derive a MAP estimator just like we previously derived ML estimators. This will result in

$$\mu_{MAP} = \frac{\alpha+h-1}{\alpha+\beta+h+t-2}.$$

12.2 Sequential updates to the posterior/ prior

In the section above see that the only difference between the posterior function and the likelihood function is the addition of $\alpha-1$ and $\beta-1$ to the exponents: α essentially increases to the number of observed head events, while β adds to the number of observed tail events. This becomes handy when we want to have sequential updates of the MAP estimate, i.e. when we only want to consider one data point at a time. In this case, before even measuring the first data point, we set up a prior representing our belief about the variables in question. If we have no such belief, expectation, personal bias, preference or opinion about the matter (or don't want it to influence our MAP estimates), we use a flat prior. If we choose to sequentially update our MAP estimate, i.e. after each new observation, we have two equivalent options*: 1) We keep our original prior (i.e. the parameters of the distribution) as it is and put all observations in an ever growing pool. After each new observation, the likelihood distribution is set up based on the complete set of observations and multiplied with the prior distribution. 2) We take advantage of the conjugate prior, which allows us to use the posterior distribution after the ith observation as the prior distribution for the

$i+1$th observation. What's happening is that we sum up all past observations into one distribution and update this distribution after each new observation with a relatively simple likelihood term.

(*) The two options are equivalent with respect to the resulting MAP estimates, but the second option is computationally more efficient. If for example each observation is a high resolution image and you get 25 new images per second, after a while you will have a rather large amount of data. In approach (1) you would have to save all the data and evaluate all of it 25 times a second. This requires ever-growing storage and computational power. In contrast, with approach (2) you only have a small and constant demand for storage and computational power. All you need to store is the current prior/ posterior distribution (which changes its parameter values but doesn't actually grow) and the latest image. The likelihood estimation is then based solely on the mentioned latest image, not on say a million images or more.

12.3 Choosing the right prior

Let's fill the thumbtack example with some life and numbers. We start with a visual inspection of the tack and we don't know which of the two sides it's more likely to fall on, so we go with an estimate of $\mu \approx .5$. To formalize this, we choose a beta-distribution (to get a conjugate prior to the Bernoulli likelihood) and specify $\alpha = \beta$. The exact value that we assign to α and β will determine the priors weight, i.e. the strength of our belief - and as a consequence, how many tosses it would take to convince us otherwise. The asymmetrical shape of the tack however doesn't help our trust in its "fairness", so we can only justify a weak prior with $\alpha = \beta = 3$ (for a flat prior we would set $\alpha = \beta = 1$). We now toss the thumbtack and it falls on its head. Our first MAP estimate with the chosen non-flat prior is $\mu_{MAP} = \frac{3+1-1}{3+3+1+0-2} = .6$. In comparison, had we chosen a flat prior, our first estimate would have been $\mu'_{MAP} = \frac{1+1-1}{1+1+1+0-2} = 1$. We now toss the thumbtack over and over, updating our prior after each toss. After 100 tosses we've seen 64 heads (and 36 tails) in total and our MAP estimate will be $\mu_{MAP} \approx .635$. Had we been using a flat prior from the start, our MAP estimate would now be $\mu'_{MAP} = .64$.

So what can we take from this? After a single or just a few observations, the MAP estimate will typically be dominated by the prior. This can help to even out the effects of a small sample size by keeping the MAP estimates in reasonable bounds - we get a more realistic estimate of the actual variables in question. This can prove valuable when forced to make tough decisions based on very little data. Nevertheless, we must be able to justify our prior belief if we bring it into the equation - a badly chosen (or 'overweight') prior can overshadow useful data and be very harmful with respect to the accuracy of our MAP estimates. With every new repetition the originally chosen prior distribution loses a bit of it's impact until it becomes negligible. The MAP estimates are then dominated by the observations.

For instance, imagine a game in which after every toss of a bent coin the players have to bet 10€ on their current estimate for the true probability of getting heads (μ), and each trials full payout goes to the player closest to the actual value (payout only after the game ends, 10 trials in total, the true μ is defined as the percentage of heads after 1000 tosses). Basically, you really don't want to enter this game with a flat prior.

12.4 Bayesian regression and regularization

Let's dig a little deeper into the similarities between Bayesian regression and regularization. We start out with a model for bivariate data (as defined in section 6.2.3). Let the model be comprised of basis functions ϕ_n and the data set be comprised of input values x_i and target values t_i:

$$y(\vec{w}, x) = \sum_n w_n \phi_n(x)$$

The corresponding likelihood function, assuming the target values $\vec{t}$ come from a normal distribution with a mean equal to $y(\vec{w}, x)$ and variance (noise) σ^2, is given by:

In order to keep the notation in the script consistent, some variable names differ from the slides and the Bishop book. Essentially, $s^2 = \alpha^{-1}$ and $\sigma^2 = \beta^{-1}$.

$$p(\vec{t}|\vec{x}\vec{w},\sigma^2) = \prod_i \mathcal{N}(t_i|y(\vec{w},x_i),\sigma^2)$$

The conjugate prior for a normally distributed likelihood function (with a known variance σ^2) is again a normal distribution. For the point we're trying to make, let's choose a prior distribution with mean zero. Since the prior distribution is an assumption about our model parameters $\vec{w}$, this normal distribution must have as many dimensions as we have parameters in the model. We therefore need the multivariate Gauss distribution, given by the density function

An infinitely broad normal distribution (i.e. $\sigma^2 \to \infty$ with $S = \sigma^2$) would constitute a flat prior and therefore lead to $\vec{w}_{MAP} = \vec{w}_{ML}$.

$$\mathcal{N}_p(\vec{\mu},S) = \frac{1}{(2\pi)^{\frac{p}{2}}|S|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\vec{\mu})^T S^{-1}(x-\vec{\mu})\right\}$$

in which p is the number of dimensions, $\vec{\mu}$ is the vector of means (one element per dimension) and S is the covariance matrix. $|S|$ denotes the determinant of the covariance matrix. For independent parameters $\vec{w}$, the covariance matrix is defined as $S = s^2 I$, I being the identity. Note that in this case s^2 denotes the variance in the prior distribution (defining the width/ shape and thus the strength of the prior), while σ^2 still denotes the variance in the model. In the case of $\vec{\mu} = \vec{0}$, the expression for the prior reduces to

So far you haven't been formally introduced to the multivariate Gauss distribution. If it looks a little confusing to you, compare it to the univariate normal distribution and keep in mind that the covariance matrix S corresponds to the variance σ^2. Add in some knowledge about linear algebra and you should get some intuition about how to read the density function.

$$p(\vec{w}|s^2) = \mathcal{N}(\vec{w}|\vec{0},s^2 I) = \frac{1}{2\pi s^2}^{\frac{p}{2}}.$$

The corresponding expression for our posterior is given by

$$p(\vec{w}|\vec{x},\vec{t},s^2,\sigma^2) \propto p(\vec{t}|\vec{x},\vec{w},\sigma^2)\cdot p(\vec{w}|s^2).$$

In order to maximize the posterior, we again use the log trick (log posterior = log likelihood + log prior) and end up with

$$\ln p(\vec{w}|\vec{x},\vec{t},s^2,\sigma^2) = -\frac{1}{2\sigma^2}\sum_i (t_i - \vec{w}^T\phi(x_i))^2 - \frac{1}{2s^2}\vec{w}^T\vec{w} + \text{const.},$$

which we now want to maximize with respect to $\vec{w}$ (i.e. we want $\vec{w}_{MAP}$). This is equal to minimizing the term

$$\frac{1}{2\sigma^2}\sum_i (t_i - \vec{w}^T\phi(x_i))^2 + \frac{1}{2s^2}\vec{w}^T\vec{w},$$

which in turn contains the squared residuals and the regularization coefficient ($\frac{\lambda}{2}w^T w$), with $\lambda = \frac{\sigma^2}{s^2}$) - the expression is therefore identical to a regularized least squares solution. The bottom line is that using a conjugate prior with mean $\vec{\mu} = \vec{0}$ is the same as applying a quadratic regularization. A higher variance in the prior (i.e. a broader and therefore "flatter" prior distribution) corresponds to a weaker regularization ($s^2 \propto \frac{1}{\lambda}$).

13 Model selection: Akaike Information Criterion

Another criterion for the selection of a model is the *Akaike Information Criterion* (AIC). While its derivation is rather lengthy, the AIC formula and its application are rather simple. In this chapter, we will try to get a grasp on what the AIC actually does, without going to town with it. If you are only interested in its application, jump to the last section of this chapter.

For the full derivation of the AIC please refer to *Konishi Kitagawa*'s *Information Criteria and Statistical Modeling*, chapter 3.

13.1 The Kullback-Leibler divergernce

Let's begin fresh. In statistical modeling, we typically want to come up with a simple description or model of some process that we are interested in. Our approach is to make observations and try to infer the properties of the underlying process or model from these observations. To this end, we propose some flexible model which is defined by a set of parameters and we are looking to find the most probable values for these parameters, given the data we have. Ideally, we would like to come up with a model that is identical to the underlying or true model. Failing that, we at least want it to be as close as possible.

Clarification: What we call model in this script is the a deterministic function, e.g. $\mu(x, \vec{w}) = \sum_k w_k \phi_k$. The PDF $p(y|x, \vec{w}) = \mu(x, \vec{w}) + \mathcal{N}(0, \sigma^2) = \mathcal{N}(\mu(x, \vec{w}), \sigma^2)$ is called the (likelihood) distribution corresponding to the model $\mu(x|\vec{w})$.

The *Kullback-Leibler divergence* (short KL divergence, also called KL distance, KL information, information divergence, information gain or relative entropy) is a measure of the dissimilarity of two distributions, usually an underlying distribution $g(x)$ and its approximation $f(x)$. Ideally, we would like to measure the KL divergence between the true distribution and each of the proposed estimated models' distributions, then pick the one with the smallest divergence. Looking at the definition of the KL divergence, we notice a little wrinkle with that approach, namely that it requires us to know the true distribution $g(x)$:

Yes, the devil comes in many forms...

It doesn't really matter what kind of model g and f correspond to, they just have to be distribution functions (for discrete distributions) or density functions (for continuous distributions).

$$D_{KL}(g, f) = \mathbb{E}_g(x)\left[\log \frac{g(x)}{f(x)}\right]$$

Denoted like this it describes the information lost when f is used to approximate g. It translates to

$$D_{KL}(g, f) = \sum_i g(x_i) \log \frac{g(x_i)}{f(x_i)}$$

for discrete distributions and

$$D_{KL}(g, f) = \int_{-\infty}^{\infty} g(x) \log \frac{g(x)}{f(x)} dx$$

for continuous distributions. The KL divergence never takes on negative values and is only zero if the two models are identical:

These are distributions of some random variable x. In contrast, the likelihood function $p(y|x, \vec{w})$, corresponding to the linear model defined in the box above is a distribution over y, which in turn depends on an x unrelated to the random variable in the formulas on the left.

$$D_{KL}(g, f) \geq 0$$

$$D_{KL}(g, f) = 0 \Leftrightarrow g(x) = f(x)$$

Since we don't know the true distribution $g(x)$ we cannot actually compute a value $D(g, f)$. We can however use the KL divergence to derive an expression that lets us compare multiple estimated models (or rather

their respective likelihood distributions). To this end, we first decompose the general expression as follows:

$$D_{KL}(g, f) = \mathbb{E}_g \left[\log \frac{g(x)}{f(x)} \right] = \mathbb{E}_g \left[\log g(x) \right] - \mathbb{E}_g \left[\log f(x) \right]$$

We see that the first term only depends on the true distribution $g(x)$ and is thus constant for the comparison of estimated distributions $f_j(x)$. The second term on the other hand is the expected log-likelihood of the estimated model (with the expectation based on the true distribution $g(x)$). It can also be expressed as

$$\mathbb{E}_g \left[\log f(x) \right] = \int_{-\infty}^{\infty} g(x) \log f(x) dx$$

for continuous models and as

$$\mathbb{E}_g \left[\log f(x) \right] = \sum_i g(x_i) \log f(x_i)$$

for discrete models. Since the expectation here is taken with respect to the distribution $g(x)$ that we don't know, we need to replace it with an estimator. For this estimator we make the assumption that all observed x_i have the same probability $\frac{1}{n}$ and construct an empirical distribution $\hat{g}(x_i) = \frac{1}{n}$. We then replace $g(x)$ by $\hat{g}(x)$ in the equation given above:

$$\mathbb{E}_{\hat{g}} \left[\log f(x) \right] = \sum_i \hat{g}(x_i) \log f(x_i) = \frac{1}{n} \sum_i \log f(x_i)$$

Multiplying both sides by n gives us

$$n\mathbb{E}_{\hat{g}} \left[\log f(x) \right] = \sum_i \log f(x_i).$$

Let's take a closer look at this formula: The term on the right calculates the log-likelihood value of our data given the estimated model: It takes the logarithm of the joint probability of all sample values x_i, given the parameter set $\vec{w}$ (which defines the shape of the likelihood distribution f). This is exactly the same term that we maximize in the ML approach. Now the formula tells us that this is equal to $n\mathbb{E}_{\hat{g}} \left[\log f(x) \right]$, which in turn is an estimator for the expected log-likelihood of samples of size n. Since the sample size n is constant for all estimated models $f_j(x)$, we can neglect it for the purpose of comparing models. In comprehension this means that, according to the KL divergence, the log-likelihood $\sum_i \log f(x_i)$ is essentially an estimator of the goodness of our model. Put differently, when we maximize the log-likelihood of our model, we actually maximize the goodness of our model.

Since our data set consists of a finite number of observations, $\hat{g}(x)$ is a discrete distribution. In this distribution, values other than the ones we already observed (x_i) have the probability zero. Accordingly, $g(x)$ and $\hat{g}(x)$ don't look much alike, their respective cumulative distribution functions $G(x)$ and $\hat{G}(x)$ however are of similar shape to each other.

$f(x) = p(x|\vec{w})$, if you will.

In conclusion (1): The Kullback-Leibler divergence is a tool to measure the similarity between two distributions, in our case the true distribution and the estimated distribution. For model-selection purposes we can disregard the first term in the KL divergence, which only depends on the true distribution and is thus constant for all estimated distributions. We can however use the second term to compare multiple distributions regardless of their absolute similarity to the true distribution. Since this second term still depends on both the estimated and the true distribution, we have to resort to an estimation of the true distribution based

on our data set. We see that $\frac{1}{n}LL$ is actually an approximation of the relative goodness of the model based on the KL divergence.

13.2 AIC: A bias-corrected approximation of the KL divergence

Now there's just one little problem with the story so far. By basing our estimation $\hat{g}(x)$ of the true distribution $g(x)$ on our data set, we introduced a bias to our estimation of the expected log-likelihood $\mathbb{E}_g[\log f(x)]$. In other words, the log-likelihood $\sum_i \log f(x_i)$ that we've proudly used to derive our ML estimators is biased as a criterion for finding the model that is most similar to the underlying model. The estimator (i.e. $\mathbb{E}_{\hat{g}}[\log f(x)] = \frac{1}{n}\sum_i \log f(x_i)$) is biased in so far, as it will yield higher log-likelihood values for a model with the parameter set $\vec{w}_{ML}$ than it will for a model based on the true parameter set $\vec{w}_{\text{true}}$. Yet logic dictates that the actual goodness of the model ($\mathbb{E}_g[\log f(x)]$) is maximized for models with the parameter set $\vec{w}_{\text{true}}$. In order to compensate for this reversal, we need to perform a bias evaluation. Correcting for the bias will allow us to derive a fair comparison of models, and we will call a bias-corrected log-likelihood an *information criterion*. In general, an information criterion can be constructed as follows:

$$IC(\vec{x}, \hat{g}) = -2(\text{log-likelihood of the estimated model} - \text{bias estimator})$$

The bias estimator can take various forms, depending on the relationship of the underlying and the estimated model, and the method used to construct the estimated model. The bias estimator in the Akaike Information Criterion (AIC) is suitable for models with parameters estimated with the maximum likelihood method. Since its derivation is quite lengthy and outside of the scope of the lecture, we skip right to the end and find that the bias estimator is equal to the number of free parameters in the model. The AIC criterion is thus defined as

$$AIC = -2\left(\log f(x, \vec{w}) - K\right)$$

with K being the number of free parameters in the estimated model, i.e. the dimensionality or length of the parameter vector $\vec{w}$. Note that information criteria (including the AIC) have negative signs and thus measure the relative badness of a distribution (and its corresponding model). Accordingly, a better model will be characterized by a lower AIC value. Note also that the AIC criterion is only a tool to compare different estimated models (or rather their likelihood distribution functions), since the criterion is based solely on the second term in the (decomposed) KL divergence. As such it doesn't capture the actual dissimilarity of the true and the estimated distribution, and therefore doesn't provide an absolute measure of the goodness (or badness) of a model.

In conclusion (2): The log-likelihood of the estimated model is a biased estimator for the goodness of a model. For models whose parameters were estimated with the maximum likelihood approach, the bias is approximately equal to the number of free parameters in the model. The AIC corrects for this bias and thus is, due to its negative sign, a fair estimator of the relative badness of a model.

14 Model selection: Bootstrap and EIC