Machine Learning

Questions and Answers

Daniel Zahnd

September 16, 2024 - November 14, 2024

Contents

1 Preliminaries **1**
1.1 Probability measures . . . 2
1.1.1 Random variable . . . 2
1.1.2 Expectation value . . . 2
1.1.3 Variance, standard deviation and covariance . . . 2
1.2 Preliminary questions . . . 3

2 Supervised learning **7**
2.1 Linear regression . . . 7
2.1.1 Least mean squares . . . 7
2.1.2 Probabilistic interpretation . . . 9
2.2 Classification and logistic regression . . . 10
2.3 Generalized linear models . . . 11
2.4 Generative learning and naive Bayes . . . 16
2.5 Support vector machines . . . 21
2.6 Decision trees . . . 28
2.7 Ensembling methods . . . 30
2.8 Regularization and model selection . . . 31

3 Unsupervised learning **31**
3.1 Clustering and k-means . . . 31
3.2 EM and factor analysis . . . 31
3.3 PCA and ICA . . . 32

4 Reinforcement learning **32**

1 Preliminaries

Usually, one deals with datasets containing of data $x \in \mathbb{R}^n$ associated to other data y with $n \in \mathbb{N}$. The data x can be scalar of vectorial, whereas the data y can also be a scalar or vector. The data x is either $y \in \mathbb{R}^q$ with $q \in \mathbb{R}$ or a number of a class $\{1, \ldots, K\}$ for $K \in \mathbb{N}$. In the case where e.g. $y \in \mathbb{R}$, one speaks of a regression problem, whereas in the case where $y \in \{1, \ldots, K\}$ are classes, one speaks of a classification task.

Usually, one has a dataset containing of various pairs of x and y; to distinguish an instance of data from components of a vector, one writes $(x^{(i)}, y^{(i)})$ for an instance of a dataset containing of $m \in \mathbb{N}$ instances of data. For components of an instance of data $x^{(i)} \in \mathbb{R}^n$, one writes

$$x^{(i)} = \begin{pmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix} \in \mathbb{R}^n. \tag{1.1}$$

For a whole dataset, one can write

$$\{(x^{(i)}, y^{(i)})\}_{i=1,\dots,m} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}. \tag{1.2}$$

The goal in machine learning is thus to find a function h with associated tunable parameters θ, that predicts y based on x; hence one can write

$$h_\theta(x) = y. \tag{1.3}$$

The ultimate goal usually is to determine the parameters (weights) θ; whereas for the shape or nature of the function h, one can usually make assumptions. The h is reminiscent of "hypothesis" and thus of "hypothesis" function.

1.1 Probability measures

1.1.1 Random variable

A random variable is some quantity x, which can take a random value. Those random values follow a certain probability distribution, which is determined by the underlying process constituting the random variable.If the probability distribution of a random variable is known, the probability density $p(x)$ can be written down for the continuous and discrete cases as

$$\int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1 \qquad \text{and} \qquad \sum_i p(x_i) = 1. \tag{1.4}$$

1.1.2 Expectation value

The expectation value $\mathbb{E}(x) \doteq \bar{x}$ of a random variable x for both the continuous and discrete case is defined as

$$\mathbb{E}(x) = \int_{-\infty}^{\infty} x p(x)\,\mathrm{d}x \qquad \text{and} \qquad \mathbb{E}(x) = \sum_i x_i p(x_i). \tag{1.5}$$

1.1.3 Variance, standard deviation and covariance

The variance $\mathbb{V}(x)$ of a continuous or discrete random variable x can be calculated by means of

$$\mathbb{V}(x) = \int_{-\infty}^{\infty} (x - \bar{x})^2 p(x)\,\mathrm{d}x \qquad \text{and} \qquad \mathbb{V}(x) = \sum_i (x_i - \bar{x})^2 p(x_i). \tag{1.6}$$

The standard deviation σ_x is defined as the square root of the variance, hence

$$\sigma_x = \sqrt{\mathbb{V}(x)}. \tag{1.7}$$

Let now $x_1, \dots, x_n$ be random variables with associated probability densities $p(x_j)$, $j \in \{1, \dots, n\}$ and joint probability densities $p(x_i, x_j)$. Let furthermore be $\bar{x}_i = \mathbb{E}(x_i)$. The covariance $\mathbb{K}(x_i, x_j)$ of two random variables for the continuous and discrete case is defined as

$$\mathbb{K}(x_i, x_j) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x_i - \bar{x}_i)(x_j - \bar{x}_j) p(x_i, x_j)\,\mathrm{d}x_i\,\mathrm{d}x_j \tag{1.8}$$

and

$$\mathbb{K}(x_i, x_j) = \sum_{k,l} (x_{i_k} - \bar{x}_i)(x_{j_l} - \bar{x}_j) p(x_{i_k}, x_{j_l}). \tag{1.9}$$

In the context of covariance, one usually also defines the correlation coefficient $\rho(x_i, x_j)$ between two random variables as

$$\rho(x_i, x_j) = \frac{\mathbb{K}(x_i, x_j)}{\sqrt{\mathbb{V}(x_i)\mathbb{V}(x_j)}} = \frac{\mathbb{K}(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}}. \tag{1.10}$$

1.2 Preliminary questions

1. **Q** — How is the uniform distribution defined?

A — Consider a continuous random variable x. The uniform probability distribution is defined by the probability density $p(x)$ as

$$p(x) = \mathcal{U}(x; a, b) = \begin{cases} \frac{1}{b-a} & , a \leq x \leq b \\ 0 & , \text{otherwise} \end{cases} \tag{1.11}$$

where a and b define the domain of the distribution. The expectation value and the variance of the uniform distribution are given by

$$\mathbb{E}(x) = \frac{a+b}{2}, \quad \mathbb{V}(x) = \frac{1}{12}(b-a)^2. \tag{1.12}$$

The uniform distribution is used for processes with no prior knowledge about the probabilities of events. It is furthermore used to model processes, where all outcomes are equally likely to happen.

2. **Q** — How is the standard normal distribution defined?

A — Consider a continuous random variable x. The standard normal probability distribution (Gaussian) is defined by the probability density $p(x)$ as

$$p(x) = \mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.13}$$

where μ defines the expectation value and σ the variance the distribution. The expectation value and the variance of the uniform distribution are given by

$$\mathbb{E}(x) = \mu, \quad \mathbb{V}(x) = \sigma^2. \tag{1.14}$$

The normal distribution is used for processes, which are influenced by an additive effect of a large number of different and independent influences modelled by arbitrary probability distributions. It pertains - with the uniform distribution - to the two default probability distributions to use, where little prior information about the modelled processes is available.

3. **Q** — How is the Bernoulli distribution defined?

A — Consider a discrete random variable x. The Bernoulli probability distribution is defined by the probability density $p(x)$ as

$$p(x) = \mathcal{B}(x; p) = \begin{cases} p & , x = 1 \\ 1-p & , x = 0 \end{cases}, \tag{1.15}$$

where $0 \leq p \leq 1$ is the probability of success (i.e., $x = 1$). The random variable x takes on only two possible values: 1 (success) or 0 (failure).

The expectation value and variance of the Bernoulli distribution are given by

$$\mathbb{E}(x) = p, \quad \mathbb{V}(x) = p(1-p). \tag{1.16}$$

The Bernoulli distribution models a single trial of an experiment where there are exactly two possible outcomes: success with probability p and failure with probability $1-p$. It is a fundamental building block of the binomial distribution, which models a sequence of independent Bernoulli trials.

4. **Q** — How is the binomial distribution defined?

A — Consider a discrete random variable x. The binomial probability distribution is defined by the probability density $p(x)$ as

$$p(x) = \mathcal{B}(x; n, p) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & , x \in \{1, \dots, n\} \\ 0 & , \text{otherwise} \end{cases}, \tag{1.17}$$

where n denotes the number of trials of an experiment and p is the respective probability of success or failure of the outcome. The expectation value and the variance of the binomial distribution are given by

$$\mathbb{E}(x) = np, \quad \mathbb{V}(x) = np(1-p). \tag{1.18}$$

The binomial distribution is used to model processes that model a series of identical and independent experiments with exactly two possible outcomes, success or failure.

5. **Q** — How is the Poisson distribution defined?

A — Consider a discrete random variable x. The Poisson probability distribution is defined by the probability density $p(x)$ as

$$p(x) = \mathcal{P}(x; \lambda) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda} & , x \in \mathbb{N}_0 \\ 0 & , \text{otherwise} \end{cases}, \tag{1.19}$$

where $\lambda > 0$ defines both the expectation value and the variance. The expectation value and the variance of the Poisson distribution are given by

$$\mathbb{E}(x) = \lambda, \quad \mathbb{V}(x) = \lambda. \tag{1.20}$$

The Poisson distribution is - similarly to the binomial distribution - used to model processes that model a series of identical and independent experiments with exactly two possible outcomes, success or failure, but where the probability p of success behaves as $p \to 0$ and where the number of trials n behaves as $n \to \infty$.

6. **Q** — How is the Poisson distribution defined?

A — Consider a discrete random variable x. The Poisson probability distribution is defined by the probability density $p(x)$ as

$$p(x) = \mathcal{P}(x; \lambda) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda} & , x \in \mathbb{N}_0 \\ 0 & , \text{otherwise} \end{cases}, \tag{1.21}$$

where $\lambda > 0$ defines both the expectation value and the variance. The expectation value and the variance of the Poisson distribution are given by

$$\mathbb{E}(x) = \lambda, \quad \mathbb{V}(x) = \lambda. \tag{1.22}$$

The Poisson distribution is - similarly to the binomial distribution - used to model processes that model a series of identical and independent experiments with exactly two possible outcomes, success or failure, but where the probability p of success behaves as $p \to 0$ and where the number of trials n behaves as $n \to \infty$.

7. **Q** — What is the popular definition of machine learning authored by Arthur Samuel in 1959?

A — According to Samuel, machine learning is the field of study that gives computers the ability to learn without being explicitly programmed. Explicitly programmed here means, that everything would be hard-coded, instead of rule-based programming.

8. **Q** — What is another popular definition of machine learning authored by Tom Michel in 1999?

A — According to Michel, a well-posed machine learning problem may be described as follows: A computer program is said to learn from experience E with respect to some tasks T and performance measure P, if its performance at tasks T as measured by performance P improves with experience E.

9. **Q** — Broadly speaking, which three types of machine learning algorithms are there?

A — Broadly speaking, there are three different categories of machine learning algorithms:

(1) Supervised learning: A machine learns how to make predictions about a specific target of interest, given some observations.

(2) Unsupervised learning: A machine learns how to find useful structures and patterns in given data by itself.

(3) Reinforcement learning: A machine has the ability to act and thus influence its own observations, thereby learning to make predictions to achieve a given goal.

An example of supervised learning would be a classification task; that is to say to assign data to two or more given classes of objects depending on one or more variables. One could for example do this by means of linear or polynomial regression.

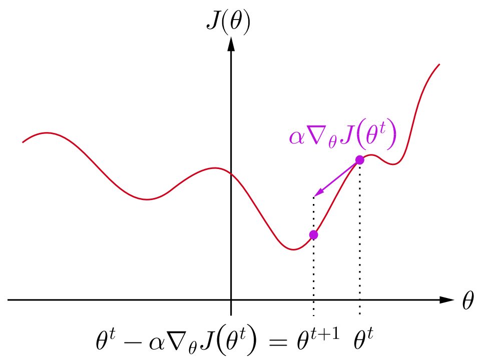
In comparison to supervised learning, unsupervised learning would for example try to separate data into two or more classes of objects, depending on what makes sense to the algorithm.

Reinforcement learning finally is about problems, where a sequence of decisions over time is required, where the basic idea is to implement a reward function as supervision to give the model a way to improve itself.

10. **Q** — How does gradient descent work; and what is the difference between stochastic gradient descent and just gradient descent?

A — Gradient descent is one of the most useful and foundational techniques in machine learning; basically, it is about minimizing a loss function $J(\theta)$ with respect to weights (parameters) θ which belong to the chosen model.

Hence, let now $J(\theta)$ be the loss function of a model, where $\theta = (\theta_1, \ldots, \theta_n)^\top$ are the parameters of the model, which need to be optimized. The loss function might take a form as given in



$J(\theta)$
$\alpha\nabla_\theta J(\theta^t)$
θ
$\theta^t - \alpha\nabla_\theta J(\theta^t) = \theta^{t+1}$
θ^t

Figure 1: Visualization of the gradient descent routine. Note, that θ is generally a vector; thus, the figure does only represent the general case in the case where θ is a scalar. If θ would consist of two components, the figure would be still visualizable in two dimensions on paper, but for higher dimensions in θ, gradient descent may not be visualizable anymore on paper.

fig. 1. Now, θ^t denotes the weights at iteration step $t \in \{1, \dots, T\}$ for a total iteration time T. Well, gradient descent updates the weights as

$$\theta^{t+1} = \theta^t - \alpha\nabla_\theta J(\theta^t), \tag{1.23}$$

where the gradient is taken with respect to the weights θ and is evaluated at θ^t, i.e.

$$\nabla_\theta = \sum_{j=1}^{n} e_j \partial_{\theta_j}. \tag{1.24}$$

In the case, where no confusion arises to which respect a gradient is taken, the subscript can also be left away. The parameter α is called the learning rate in machine learning and it has to be chosen such, that the algorithm does find the global, instead of just a local minimum of $J(\theta)$.

11. **Q** — What is the rationale for the learning rate α in gradient descent?

A — Consider fig. 2; it shows a curve $J(\theta)$ and a quadratic function $g(\theta) = \|\theta - \theta^t\|$. Now, one can Taylor expand the function $J(\theta)$ around an iteration step θ^t quite easily to first order in θ

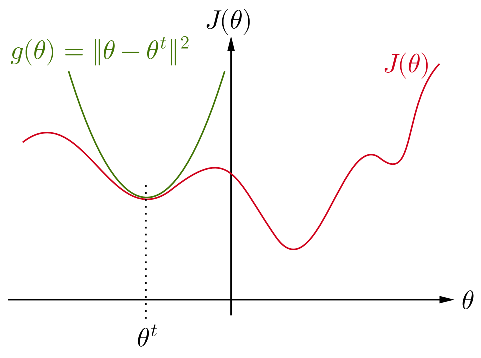


Figure 2: Rationale behind the gradient descent routine and the learning rate α.

as

$$J(\theta) \approx J(\theta^t) + \nabla_\theta J(\theta^t)^\top (\theta - \theta^t). \tag{1.25}$$

Now, the second order term would involve the Hessian, which can be quite complicated. However, one can always add a quadratic term of the form $g(\theta)$, which is scaled by some parameter ε, such that the resulting approximation of $J(\theta)$ always stays above the exact function; the added term with ε thus approximates the Hessian. One hence has

$$J(\theta) \approx J(\theta^t) + \nabla_\theta J(\theta^t)^\top (\theta - \theta^t) + \frac{1}{2\varepsilon}\|\theta - \theta^t\|^2. \tag{1.26}$$

In order to find a minimum of this expression, one has to compute the gradient of the expression with respect to θ, set it to zero and solve for θ. From this,

$$\nabla_\theta J(\theta) \approx 0 + \nabla_\theta J(\theta^t) + \frac{1}{\varepsilon}(\theta - \theta^t) \overset{!}{=} 0 \quad \Leftrightarrow \quad \theta = \theta^t - \varepsilon \nabla_\theta J(\theta^t) \tag{1.27}$$

follows. This is nothing but gradient descent, where $\varepsilon = \alpha$ is the learning rate. As one can see, the width $(2\varepsilon)^{-1}$ of the added quadratic term to the first order Taylor expansion of $J(\theta)$ is the inverse of the learning rate. If the learning rate is large, the width of the added quadratic term is very narrow.

2 Supervised learning

2.1 Linear regression

2.1.1 Least mean squares

12. **Q** — How can a most simple machine learning model performing affine regression be implemented?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \mathbb{R}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \dots, m\}$ be a training dataset. Now, for affine regression, we can suggest a hypothesis function

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n, \tag{2.1}$$

which, defining $x \doteq (1, x_1, x_2, \dots, x_n)^\top$, can be written more compactly as

$$h_\theta(x) = \theta^\top x. \tag{2.2}$$

As a loss function to minimize, it makes sense to take the Euclidean norm as a metric to measure "closeness" of the prediction $h_\theta(x^{(i)})$ to the actual value $y^{(i)}$, thus

$$J(\theta) \doteq \frac{1}{2}\sum_{i=1}^{m} \|h_\theta(x^{(i)}) - y^{(i)})\|^2 = \frac{1}{2}\sum_{i=1}^{m} \|\theta^\top x^{(i)} - y^{(i)})\|^2. \tag{2.3}$$

The update rule for this regression task would be thus given by

$$\theta^{t+1} = \theta^t - \alpha \nabla_\theta J(\theta) = \theta^t - \alpha \sum_{i=1}^{m} \left(\theta^\top x^{(i)} - y^{(i)}\right) x^{(i)}, \tag{2.4}$$

where $\alpha \in \mathbb{R}$ is a suitably chosen learning rate. The proposed algorithm is called a least mean squares algorithm. This algorithm is equivalent to stating the optimal model parameters $\hat{\theta}$ as

$$\hat{\theta} = \arg\min_\theta \left(\sum_{i=1}^{m} \left[\theta^\top x^{(i)} - y^{(i)}\right]^2 \right). \tag{2.5}$$

13. **Q** — How can the least mean squares method be given in matrix form in the general case?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \mathbb{R}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \ldots, m\}$ be a training dataset. Define a matrix X, such that $(x^{(i)})^\top$ is the i-th row and a matrix Y, such that $y^{(i)}$ is also the i-th row, namely

$$X \doteq \begin{pmatrix} 1 & (x^{(1)})^\top \\ \vdots & \vdots \\ 1 & (x^{(m)})^\top \end{pmatrix}, \qquad Y \doteq \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}. \tag{2.6}$$

Hereby, a column of ones was added to the matrix X to account for the shift parameter of an affine functino. Given a parameter vector $\theta^\top = (\theta_0, \theta_1, \ldots, \theta_n)$, one can then write down the hypothesis function $h_\theta(X)$ as

$$h_\theta(X) = X\theta \approx Y. \tag{2.7}$$

We thus write down the loss function as

$$J(\theta) = \frac{1}{2}\|X\theta - Y\|^2 = \frac{1}{2}(X\theta - Y)^\top (X\theta - Y). \tag{2.8}$$

Taking the gradient ∇_θ of this expression leads to

$$\nabla_\theta J(\theta) = \theta^\top X^\top X - (X^\top Y)^\top. \tag{2.9}$$

Setting this expression to zero yields the closed-form solution

$$\theta^\star = (X^\top X)^{-1} X^\top Y \tag{2.10}$$

to the optimization problem

$$\theta^\star = \arg\min_\theta \left[J(\theta)\right]. \tag{2.11}$$

14. **Q** — There is a closed form solution to the least mean squares problem? Why is it however not heavily used in practice?

A — The reason behind this is that the closed-form solution to the least mean squares problem involves the calculation of inverse matrices. The inverse matrix to be inverted is of size $m \times m$, where $m \in \mathbb{N}$ is the amount of data instances. Calculating the inverse of a matrix is computationally very demanding, especially for large matrices, as the calculation time goes with m^2.

Therefore, when working with large datasets, the closed-form solution is not the best option; rather, one used the gradient descent routine, which is much more efficient with large datasets, because no inverse matrices have to be calculated.

15. **Q** — How is the locally weighted affine regression method defined? What is the main difference to the normal affine regression method?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \mathbb{R}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \ldots, m\}$ be a training dataset. Now, the optimal parameters $\hat{\theta}$ for affine regression are found by means of performing

$$\hat{\theta} = \arg\min_\theta \left(\sum_{i=1}^{m} \left[\theta^\top x^{(i)} - y^{(i)}\right]^2 \right). \tag{2.12}$$

Now, the locally weighted affine regression takes as an input a query vector x and uses a weighting function $w_i(x)$ defined by the exponential

$$w_i(x) = e^{-\frac{(x - x^{(i)})^2}{2\tau^2}}, \quad \tau \in \mathbb{R} \tag{2.13}$$

to weigh the samples $x^{(i)}$ close to the query more than samples further away from it. This weighting function is then multiplied by the objective, thus one has

$$\hat{\theta} = \arg\min_{\theta} \left(\sum_{i=1}^{m} w_i(x) \left[\theta^\top x^{(i)} - y^{(i)} \right]^2 \right) \tag{2.14}$$

for the optimal parameters $\hat{\theta}$ of the locally weighted affine regression model. Thereby, also the main difference to the normal affine regression method is evident; the locally weighted method takes as an input a query vector x which requires a new optimization of the model parameters each time a new (different) query is made; the normal affine regression method however requires to find the optimal model parameters only once, independent of what the query vector x will be.

2.1.2 Probabilistic interpretation

16. **Q** — Which are three often used assumptions in machine learning?

A — The three main assumptions are:

(1) Feature data x and associated data y (related by a model $y = f(x;\theta)$) are modeled by means of a parametric probability density, that is to say, one assumes, that the data follows a probability density
$$p = p(y|x;\theta), \tag{2.15}$$
where θ are the model parameters.

(2) The data is assumed to be independent and identically distributed (IID). This assumption allows to write a joint probability distribution $p(y^{(i)}, \ldots, y^{(m)}|x^{(1)}, \ldots, x^{(m)}; \theta)$ as a simple product, namely
$$p(y^{(i)}, \ldots, y^{(m)}|x^{(1)}, \ldots, x^{(m)}; \theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta). \tag{2.16}$$

(3) The maximum likelihood approach can be used to identify the model parameters θ for the model $y = f(x;\theta)$, that is to say the optimal model parameters are found by means of
$$\hat{\theta} = \arg\max_{\theta} \left(\sum_{i=1}^{m} \ln \left[p(y^{(i)}|x^{(i)}; \theta) \right] \right). \tag{2.17}$$

17. **Q** — How and under which assumptions can it be shown, that Gaussian maximum likelihood and least mean squares yield the same result?

A — Assume, that the data y is given by a linear model of the feature data x, where Gaussian noise is added. That is to say, one has a model

$$y \approx h_\theta(x) = \theta^\top x + \eta, \quad p(\eta) = \mathcal{N}(\eta; \mu = 0, \sigma = 1). \tag{2.18}$$

The conditional probability for y given x can thus be written as

$$p(y|x;\theta,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-\theta^\top x)^2}{2\sigma^2}}, \tag{2.19}$$

where we have made the assumption that the expectation value for $y \approx h_\theta(x)$ is precisely the affine model $\theta^\top x$, thus $\mathbb{E}[h_\theta(x)] = \theta^\top x$; this is equivalent to stating, that we expect the data y to follow an affine model of x. Now, we have samples $\{(x^{(i)}, y^{(i)})\}_{i=1,\dots,m}$. The maximum likelihood approach with the given assumptions thus yields

$$p(y^{(1)}, \dots, y^{(m)}|x^{(1)}, \dots, x^{(m)}; \theta, \sigma) = \prod_{i=1}^{m} p(^{(i)}|x^{(i)}; \theta, \sigma) \tag{2.20}$$

because of the IID assumption. The optimal model parameters $\hat{\theta}$ are thus given by means of

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta,\sigma} \left(\sum_{i=1}^{m} \ln \left[p(y^{(i)}|x^{(i)}; \theta, \sigma) \right] \right) \\ &= \arg\max_{\theta,\sigma} \left(\ln \left[\frac{1}{\sqrt{2\pi}\sigma} \right] - \frac{1}{2\sigma^2} \sum_{i=1}^{m} \left[y^{(i)} - \theta^\top x^{(i)} \right]^2 \right) \\ &= \arg\min_{\theta} \left(\sum_{i=1}^{m} \left[\theta^\top x^{(i)} - y^{(i)} \right]^2 \right), \end{aligned} \tag{2.21}$$

which is exactly the equation we obtained for the affine regression model with the least squares method. Note, that in the above last step we used $\sigma = 1$.

18. **Q** — How is the maximum likelihood approach for optimization problems defined?

A — Suppose, that one has data y associated to other data x by means of a model

$$y \approx h_\theta(x), \tag{2.22}$$

that depends on some model parameters θ, which are to be found. What has to be assumed for the model $h_\theta(x)$ is that $y|x$ follows a certain probability density $p(y|x)$. Suppose now, that one has samples

$$\{(x^{(i)}, y^{(i)})\}_{i=1,\dots,m} \tag{2.23}$$

that are independently and identically distributed, one can find the model parameters θ by means of the so-called maximum likelihood approach. One calculates

$$p(y^{(1)}, \dots, y^{(m)}|x^{(1)}, \dots, x^{(m)}; \theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta). \tag{2.24}$$

Hereby, the identically and independently distributed (IID) assumption has been used to write the joint probability as a product. Now, in order to find the model parameters θ, this probability has to be maximized, since it should be highly likely to get precisely the samples $\{(x^{(i)}, y^{(i)})\}_{i=1,\dots,m}$. Therefore, the model parameters $\hat{\theta}$ are given by

$$\hat{\theta} = \arg\max_{\theta} \left(\sum_{i=1}^{m} \ln \left[p(y^{(i)}|x^{(i)}; \theta) \right] \right). \tag{2.25}$$

Taking the logarithm of the probability renders the product as a sum, without changing the argument of maximal probability in θ.

2.2 Classification and logistic regression

19. **Q** — How is the logistic regression algorithm defined?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{0, 1\}$; this is to say, we have a binary classification task at hand. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \dots, m\}$ be a training dataset. We want to find a model, which classifies the data x into either the category $y = 0$ or $y = 1$. For this purpose, one can propose the hypothesis function

$$p(y=1|x;\theta) \approx h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}, \quad p(y=0|x;\theta) \approx 1 - h_\theta(x), \tag{2.26}$$

where $\theta^\top = (\theta_0, \theta_1, \ldots, \theta_n)$ and $x = (1, x_1, \ldots, x_n)^\top$. Now, one can rewrite

$$p(y=1|x;\theta) = \phi, \qquad p(y=0|x;\theta) = 1 - \phi \tag{2.27}$$

and thus in general

$$p(y|x;\theta) = \phi^y (1-\phi)^{1-y}. \tag{2.28}$$

The maximum likelihood approach now yields for the optimal model parameters $\hat{\theta}$

$$\begin{aligned} \hat{\theta} &= \arg\max_\theta \left(\sum_{i=1}^m \ln \left[p(y^{(i)}|x^{(i)};\theta) \right] \right) \\ &= \arg\max_\theta \left(\sum_{i=1}^m \left[y^{(i)} \ln \left(\frac{1}{1+e^{-\theta^\top x^{(i)}}} \right) + \left(1 - y^{(i)}\right) \ln \left(\frac{e^{-\theta^\top x^{(i)}}}{1+e^{-\theta^\top x^{(i)}}} \right) \right] \right). \end{aligned} \tag{2.29}$$

With the resulting model, one can then query for an x, with which probability it belongs to the class $y = 1$ or $y = 0$ by means of calculating

$$p(y=1|x;\theta) = h_\theta(x), \qquad p(y=0|x;\theta) = 1 - h_\theta(x). \tag{2.30}$$

2.3 Generalized linear models

20. **Q** — Given some function $f(y)$ and a probability density of y given by $p(y|x)$, where x is some other data. What is the expectation value $\mathbb{E}[f(y)|x]$?

A — The expectation value of a continuous random variable y with probability density $p(y)$ is defined as

$$\mathbb{E}(y) = \int_{\mathbb{R}} y p(y) \, \mathrm{d}y. \tag{2.31}$$

Given some conditional probability density $p(y|x)$ and a function $f(y)$, the expectation value $\mathbb{E}[f(y)|x]$ is given by

$$\mathbb{E}[f(y)|x] = \int_{\mathbb{R}} f(y) p(y|x) \, \mathrm{d}y, \tag{2.32}$$

since

$$\int_{\mathbb{R}} p(y|x) \, \mathrm{d}y = 1 \quad \text{and} \quad \mathbb{E}[y|x] = \int_{\mathbb{R}} y p(y|x) \, \mathrm{d}y. \tag{2.33}$$

21. **Q** — What is the exponential family of probability densities and why is it important for generalized linear models?

A — A probability density $p(y|x;\eta)$ is said to belong to the exponential family, if it can be written in the form

$$p(y|x;\eta) = b(y) e^{\eta^\top T(y) - a(\eta)}, \tag{2.34}$$

where

(1) η is called the natural parameter,

(2) $T(y)$ is called the sufficient statistic (oftentimes $T(y) = y$) and where

(3) $a(\eta)$ is the logarithmic partition function, essentially playing the role of a normalizing constant.

Recall, that y is related to x by means of a hypothesis function $h_\eta(x)$; that is why x appears in the conditional probability $p(y|x;\eta)$. The parameters to optimize for the family of these distributions are η.

22. **Q** — What is the general framework of a generalized linear model?

A — There are three properties, by which a generalized linear model can be defined:

(1) The parameters η to be optimized must be of the form $\eta = \theta^\top x = \eta(\theta)$.

(2) The probability density $p(y|x;\theta)$ (where $\theta = \theta(\eta)$) used to model the data must belong to the exponential family; therefore, one has to be able to rewrite $p(y|x;\theta)$ as $p(y|x;\theta) = b(y)e^{\eta^\top T(y)-a(\eta)}$ with some $b(y)$, $T(y)$ and $a(\eta)$.

(3) The hypothesis function $h_\theta(x)$ modeling y must be of the form $h_\theta(x) = \mathbb{E}(T(y)|x)$.

23. **Q** — How is affine regression derived from a generalized linear model?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \mathbb{R}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \dots, m\}$ be a training dataset.

In order to derive a generalized linear model, we can perform the three key steps. First, we try to choose a probability density $p(y|x)$, which is suitable for the problem and check, if the chosen density belongs to the exponential family. For the sake of simplicity, $x \in \mathbb{R}$ and $y \in \mathbb{R}$ and $\sigma = 1$ for the moment. The normal distribution reads in this case as

$$p(y|x;\mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2}}. \tag{2.35}$$

Rearranging terms and defining $\mu \doteq \eta$, $T(y) \doteq y$, $a(\eta) \doteq \eta^2 = \mu^2$ and $b(y) \doteq (2\pi)^{-1/2}e^{-y^2/2}$, one can rewrite the above probability density as

$$p(y|x;\eta) = b(y)e^{\eta^\top T(y)-a(\eta)}, \tag{2.36}$$

which proves that the normal distribution belongs to the exponential family. Thus, requirement (2) generalized linear (GLMs) is satisfied. Now, the third requirement of GLMs is that the hypothesis function is given by

$$h_\theta(x) = \mathbb{E}(T(y)|x). \tag{2.37}$$

In our case we have $T(y) = y$ and hence

$$h_\theta(x) = \mathbb{E}(T(y)|x) = \mathbb{E}(y|x) = \int_{\mathbb{R}} y p(y|x;\eta)\,\mathrm{d}y = \eta = \mu. \tag{2.38}$$

Requirement (1) of GLMs now requires us to choose $\eta = \theta^\top x$, which gives us with $h_\theta(x) = \eta$ the hypothesis function

$$h_\theta(x) = \theta^\top x \tag{2.39}$$

to implement, which is exactly the hypothesis function for affine regression.

24. **Q** — How is logistic regression derived from a generalized linear model?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{0, 1\}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \dots, m\}$ be a training dataset.

In order to derive a generalized linear model, we can perform the three key steps. First, we try to choose a probability density $p(y|x)$, which is suitable for the problem and check, if the chosen density belongs to the exponential family. We choose a Bernoulli distribution of the form

$$p(y|x;\phi) = \phi^y(1-\phi)^{1-y}. \tag{2.40}$$

Rearranging terms and defining $b(y) = 1$ (unity matrix), $\eta \doteq \ln\left(\frac{\phi}{1-\phi}\right)$ and $a(\eta) \doteq -\ln(1-\phi) = \ln(1+e^\eta)$ and $T(y) \doteq y$, one can rewrite the above probability density as

$$p(y|x;\eta) = b(y)e^{\eta^\top T(y)-a(\eta)}, \tag{2.41}$$

which proves that the Bernoulli distribution belongs to the exponential family. Thus, requirement (2) generalized linear (GLMs) is satisfied. Note, that η relates to ϕ as

$$\phi(\eta) = \frac{1}{1+e^{-\eta}}. \tag{2.42}$$

Now, the third requirement of GLMs is that the hypothesis function is given by

$$h_\theta(x) = \mathbb{E}(T(y)|x). \tag{2.43}$$

In our case we have $T(y) = y$ and hence

$$h_\theta(x) = \mathbb{E}(T(y)|x) = \mathbb{E}(y|x) = \sum_{y\in\{0,1\}} yp(y|x;\eta) = \phi. \tag{2.44}$$

Requirement (1) of GLMs now requires us to choose $\eta = \theta^\top x$, which gives us with $h_\theta(x) = \phi$ the hypothesis function

$$h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}} \tag{2.45}$$

to implement, which is exactly the hypothesis function for logistic regression.

25. **Q** — How does the perceptron connect to generalized linear models?

A — The perceptron is nothing but a generalized linear model, if an appropriate activation function $f : \mathbb{R}^{d+1} \to \mathbb{R}$ has been chosen. Consider fig. 3 for a visualization. Let $x \in \mathbb{R}^d$ be

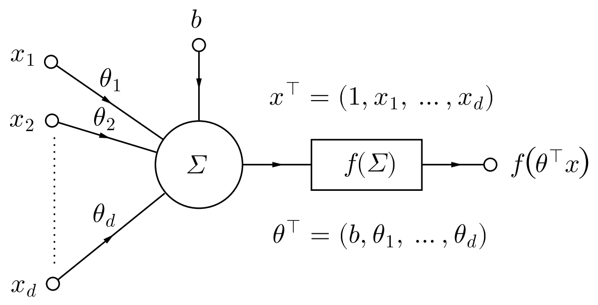


Figure 3: Visualization of a basic perceptron.

a feature vector and $y \in \{0,1\}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1,\ldots,m\}$ be a training dataset. If the activation function f is chosen to be the sigmoid function

$$f(\theta^\top x) = \frac{1}{1+e^{-\theta^\top x}}, \tag{2.46}$$

then, the perceptron reduces to logistic regression, as the hypothesis function in this case is

$$h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}. \tag{2.47}$$

and hence is a form of a generalized linear model. Therefore, the perceptron with the sigmoid activation function can be trained using gradient descent as in logistic regression. This finding also generalizes; if the activation function f belongs to the exponential family, every perceptron is a basic unit of a generalized linear model!

26. **Q** — How is logistic regression for $k \in \mathbb{N}$ classes rather than just for two classes defined?

A — Logistic regression for multiple classes is also known as multinomial logistic regression or softmax regression. This is a widely used technique in machine learning to cluster data into multiple classes.

Softmax regression can be derived based on the framework of GLMs. Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{1, \dots, k\}$, where each index for y refers to a different class. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \dots, m\}$ be a training dataset. Now, the probability density for the multinomial case can be generalized from the binomial case, which is in that case the Bernoulli probability density. Generalized to the multinomial case, the probability density reads as

$$p(y|x;\theta) = \phi_1^{\mathbb{I}\{y=1\}} \cdot \dots \cdot \phi_k^{\mathbb{I}\{y=k\}}, \tag{2.48}$$

where

$$\mathbb{I}\{y=i\} = \begin{cases} 1, & y = i \\ 0, & \text{otherwise} \end{cases}, \tag{2.49}$$

is the indicator function and where $\phi_i = \phi_i(\theta)$ are functions of the model parameters θ. One can hence write the individual probability densities ϕ_i as

$$\phi_i = p(y = i|x;\theta). \tag{2.50}$$

Now, concerning the ϕ_i, one can state that they need to add up to 1, since this is a necessary condition on probability densities; hence we have

$$\sum_{i=1}^{k} \phi_i = 1, \qquad \phi_k = 1 - \sum_{i=1}^{k-1} \phi_i. \tag{2.51}$$

Since logistic regression is also in its multinomial form a linear model, one needs to verify the three key conditions for GLMs. First, we check, if the above proposed probability density is part of the exponential family. Towards this end, we define the function $T(y)$ as a $k-1$ dimensional vector, that for $y = i$ has a one in row i but zeros in all other rows, hence

$$[T(y)]_i = \mathbb{I}\{y=i\}, \quad i \in \{1, \dots, k-1\} \quad \Leftrightarrow \quad T(y) = \begin{pmatrix} \mathbb{I}\{y=1\} \\ \vdots \\ \mathbb{I}\{y=k-1\} \end{pmatrix}. \tag{2.52}$$

With this function, we can write

$$\begin{aligned} p(y|x;\theta) &= \phi_1^{\mathbb{I}\{y=1\}} \cdot \dots \cdot \phi_k^{\mathbb{I}\{y=k\}} \\ &= \phi_1^{[T(y)]_1} \cdot \dots \cdot \phi_{k-1}^{[T(y)]_{k-1}} \phi_k^{1-\sum_{j=1}^{k-1}[T(y)]_j} \\ &= \exp\left([T(y)]_1 \ln(\phi_1) + \dots + [T(y)]_{k-1} \ln(\phi_{k-1}) + \left(1 - \sum_{j=1}^{k-1} [T(y)]_j\right) \ln(\phi_k) \right) \\ &= \exp\left([T(y)]_1 \ln\left(\frac{\phi_1}{\phi_k}\right) + \dots + [T(y)]_{k-1} \ln\left(\frac{\phi_{k-1}}{\phi_k}\right) + \ln(\phi_k) \right). \end{aligned} \tag{2.53}$$

Defining a vector η of $k-1$ elements as

$$\eta^\top \doteq \left(\ln\left[\frac{\phi_1}{\phi_k}\right], \ldots, \ln\left[\frac{\phi_{k-1}}{\phi_k}\right]\right), \tag{2.54}$$

and furthermore defining the k-th element as $\eta_k \doteq 0$, we can write $p(y|x;\theta)$ finally as

$$p(y|x;\theta) = e^{\eta^\top T(y) - (-\ln[\phi_k])}. \tag{2.55}$$

If it can be shown, that $-\ln(\phi_k)$ can be written as a function $a(\eta)$ of η, it is verified that the proposed probability density $p(y|x;\theta)$ belongs to the exponential family. To this end, one calculates

$$e^{\eta_i} = \frac{\phi_i}{\phi_k} \quad \Leftrightarrow \quad \phi_i = \phi_k e^{\eta_i}, i \in \{1, \ldots, k\}. \tag{2.56}$$

Recall now, that the ϕ_i must sum up to 1; hence we have

$$1 = \sum_{j=1}^{k} \phi_j = \phi_k \sum_{j=1}^{k} e^{\eta_j} \quad \Leftrightarrow \quad \phi_k = \frac{1}{\sum_{j=1}^{k} e^{\eta_j}} \quad \Leftrightarrow \quad \phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}. \tag{2.57}$$

The last expression is called the softmax function and is heavily used in deep learning as a so-called activation function for classification tasks. Assume, that a node in a deep neural network has k inputs $\{\eta_1, \ldots, \eta_k\}$. Acting with the softmax function on these inputs renders every input to a probability $\phi_i = e^{\eta_i}\left(\sum_{j=1}^{k} e^{\eta_j}\right)^{-1}$ of belonging to the i-th class. Note now, that it has been shown that $-\ln(\phi_k)$ can be written as a function of η. Hence, if $I^\top \doteq (1, \ldots, 1)$ is a vector of $k-1$ entries which are all ones, we can define

$$a(\eta) \doteq -\ln(\phi_k) = -\ln\left(\frac{1}{\sum_{j=1}^{k} e^{\eta_j}}\right) = -\ln\left(\frac{1}{I^\top \eta}\right). \tag{2.58}$$

Furthermore defining $b(y) = b \doteq 1$, the probability density $p(y|x;\theta)$ can indeed be written in the form

$$p(y|x;\theta) = b(y) e^{\eta^\top T(y) - a(\eta)} \tag{2.59}$$

which proves, that said probability density belongs to the exponential family. So we can proceed with setting $\eta \doteq \theta x$, where θ now is a matrix

$$\theta = \begin{pmatrix} \theta_1^\top \\ \vdots \\ \theta_{k-1}^\top \end{pmatrix} \quad \text{such that} \quad \eta_i = \theta_i^\top x \tag{2.60}$$

consisting of vectors $\theta_1, \ldots, \theta_{k-1}$, each of dimension n. What remains is to calculate the hypothesis function $h_\theta(x)$. So we evaluate

$$h_\theta(x) = \mathbb{E}[T(y)|x] = \sum_{i=1}^{k} T(y=i) p(y=i|x;\theta) = \sum_{i=1}^{k} T(y=i)\phi_i = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_{k-1} \end{pmatrix}. \tag{2.61}$$

For every ϕ_i for $i \in \{1, \ldots, k-1\}$ we have

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} = \frac{e^{\theta_i^\top x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^\top x}}, \tag{2.62}$$

where $\eta_k = 0$ and where $\phi_k = 1 - \sum_{j=1}^{k-1} \phi_j$.

So finally, one can achieve the optimal model parameters $\hat{\theta}$ by the maximum likelihood approach as

$$\begin{aligned}
\hat{\theta} &= \arg\max_{\theta} \left(p(y^{(1)}, \dots, y^{(m)} | x^{(1)}, \dots, x^{(m)}; \theta)\right) \overset{\text{IID}}{=} \arg\max_{\theta} \left(\sum_{i=1}^{m} \ln \left[p(y^{(i)} | x^{(i)}; \theta) \right] \right) \\
&= \arg\max_{\theta} \left(\sum_{i=1}^{m} \ln \left[\phi_1^{\mathbb{I}\{y^{(i)}=1\}} \cdot \dots \cdot \phi_k^{\mathbb{I}\{y^{(i)}=k\}} \right] \right) \\
&= \arg\max_{\theta} \left(\sum_{i=1}^{m} \sum_{j=1}^{k} \mathbb{I}\{y^{(i)} = j\} \ln(\phi_j) \right) \\
&= \arg\max_{\theta} \left(\sum_{i=1}^{m} \left[\sum_{j=1}^{k-1} \mathbb{I}\{y^{(i)} = j\} \ln \left(\frac{e^{\theta_j^\top x^{(i)}}}{1 + \sum_{l=1}^{k-1} e^{\theta_l^\top x^{(i)}}} \right) + \mathbb{I}\{y^{(i)} = k\} \ln \left(1 - \sum_{u=1}^{k-1} \phi_u \right) \right] \right)
\end{aligned} \tag{2.63}$$

The final expression with everything plugged in reads as

$$\arg\max_{\theta} \left(\sum_{i=1}^{m} \left[\sum_{j=1}^{k-1} \mathbb{I}\{y^{(i)} = j\} \ln \left(\frac{e^{\theta_j^\top x^{(i)}}}{1 + \sum_{l=1}^{k-1} e^{\theta_l^\top x^{(i)}}} \right) + \mathbb{I}\{y^{(i)} = k\} \ln \left(1 - \sum_{u=1}^{k-1} \left[\frac{e^{\theta_u^\top x^{(i)}}}{1 + \sum_{l=1}^{k-1} e^{\theta_l^\top x^{(i)}}} \right] \right) \right] \right), \tag{2.64}$$

which can be maximized (or minimized with negative sign) with respect to the model parameters θ.

2.4 Generative learning and naive Bayes

27. **Q** — Which are two very important formulas from probability theory heavily used in generative learning?

A — The two formulas are the the Bayes theorem and second the marginalization theorem for probabilities.

Now, consider random variables x and y with probability densities $p(x)$, $p(y)$ and a joint probability density $p(x, y)$. Furthermore, conditional probabilities are given as $p(x|y)$ and $p(y|x)$. The Bayes theorem then reads

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \qquad \text{or} \qquad p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{2.65}$$

Now, the second important formula - the marginalization theorem - is given as

$$p(x) = \int_{\mathbb{R}} p(x, y)\, \mathrm{d}y \qquad \text{or} \quad p(y) = \int_{\mathbb{R}} p(x, y)\, \mathrm{d}x. \tag{2.66}$$

Now, if every event in x or y is disjoint from any other event possible, the law of total probability applies, which permits to write the joint probability $p(x, y)$ as

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x). \tag{2.67}$$

By means of this, the Bayes theorem can be expressed by only two assumed quantities, rather than three, namely by means of $p(x|y)$ and $p(y)$ as

$$p(y|x) = \frac{p(x|y)p(y)}{\int_{\mathbb{R}} p(x|y)p(y)\, \mathrm{d}y}, \tag{2.68}$$

where the integral discretizes to a sum, if the random variable(s) are discrete.

28. **Q** — What is the difference between a discriminative and generative approach in machine learning? Explain by using the Bayes theorem!

A — In machine learning, one models a probability density $p(y|x)$ of data y, given other data x. One can go about this using two different approaches; the generative and the discriminative approach.

The discriminative approach is to directly and explicitly define a probability density for $p(y|x)$; say, a general linear model like the logistic regression. So in this case, one has

$$p(y|x;\theta) = h_\theta(x), \tag{2.69}$$

where $h_\theta(x)$ is some hypothesis function like the sigmoid function, which is parametrized by model parameters θ. This approach requires an explicit definition of the probability density, which is oftentimes hard to determine.

Because many times it is easier to find trustworthy expressions for the probability densities $p(x|y)$ and $p(y)$, the generative approach where one models $p(y|x)$ indirectly by means of the Bayes theorem as

$$p(y|x) = \frac{p(x|y)p(y)}{\int_{\mathbb{R}} p(x|y)p(y)\,\mathrm{d}y} \tag{2.70}$$

is easier. This is also the difference between the two approaches; the discriminative approach models $p(y|x)$ explicitly, whereas the generative approach models $p(y|x)$ indirectly by means of modeling $p(x|y)$ and $p(y)$ and applying the Bayes theorem.

29. **Q** — What is generative learning?

A — Oftentimes, one wants to model the probability density $p(y|x)$, where $x = (x_1, \dots, x_n)^\top$ for example contain variables like the amount of rooms and the total living area of a house, and where y would denote the prize of the house. This is called discriminative learning; so one wants to infer primarily one variable based on other variables.

The other way round, where one models $p(x|y)$, is called generative learning. There, one wants to infer more than one variables based on one variable. For example, one wants to have probability densities for the amount of rooms and the total living area of a house, if one only knows the probability density for the prize.

30. **Q** — How does Gaussian discriminative analysis (GDA) work?

A — Gaussian discriminative analysis is an example of a generative learning model. Hence, we assume the probabilities $p(x|y)$ and $p(y)$ in order to calculate $p(y|x)$ via the Bayes theorem.

Let y be a variable that can take two values; either 0 or 1. Thus, the probability $p(y;\phi)$ can be given by a Bernoulli probability density as $p(y = 1;\phi) = \phi$ and $p(y = 0;\phi) = 1 - \phi$, where $\phi \in [0, 1]$. For the probabilities $p(x|y = 0;\mu_0, \Sigma)$ and $p(x|y = 1;\mu_0;\Sigma)$, we assume Gaussian probability densities; thus one can write

$$p(x|y = 0;\mu_0, \Sigma) = \mathcal{N}(x;\mu_0, \Sigma), \quad p(x|y = 1;\mu_0, \Sigma) = \mathcal{N}(x;\mu_1, \Sigma), \tag{2.71}$$

where

$$\mathcal{N}(x;\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \tag{2.72}$$

is the n-dimensional multivariate normal distribution, where Σ is the covariance matrix of the different components $x = (x_1, \dots, x_n)^\top$ and $\mu = (\mu_1, \dots, \mu_n)^\top$ is a vector of mean values for each component. The expression $|\Sigma|$ means $|\Sigma| = \det(\Sigma)$. Note, that the covariance matrix is assumed to be the same for both probability densities above, but not the mean values. In one expression, $p(x|y)$ and $p(y)$ can be written as

$$p(x|y;\mu_0,\mu_1,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \prod_{j=0}^{1} \exp\left(-\frac{1}{2}(x-\mu_j)^\top \Sigma^{-1}(x-\mu_j)\right)^{\mathbb{I}\{y=j\}} \tag{2.73}$$

and

$$p(y;\phi) = \phi^{\mathbb{I}\{y=1\}}(1-\phi)^{\mathbb{I}\{y=0\}}. \tag{2.74}$$

The model parameters θ to learn by optimization are therefore given by

$$\theta = \{\mu_0, \mu_1, \Sigma, \phi\}. \tag{2.75}$$

Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{0,1\}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1,\dots,m\}$ be a training dataset. The way to go about optimization is taking the maximum likelihood of the data; thus we want to maximize $L(\theta) \doteq \sum_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$ with respect to θ. This is equivalent to the formulation

$$\begin{aligned}\hat{\theta} = \arg\min_\theta\,[l(\theta)] &= \arg\min_\theta \left(\overbrace{-\sum_{i=1}^{m} \ln\left[p(x^{(i)}|y^{(i)};\theta)p(y^{(i);\theta})\right]}^{\doteq\, l(\theta)} \right) \\ &= \arg\min_\theta \left(-\sum_{i=1}^{m} \ln\left[p(x^{(i)}|y^{(i)};\theta)\right] - \sum_{i=1}^{m} \ln\left[p(y^{(i)};\theta)\right] \right)\end{aligned} \tag{2.76}$$

for the optimal model parameters $\hat{\theta}$. Now, explicitly inserting the probability densities, one obtains

$$\hat{\theta} = \arg\min_\theta \left(\sum_{i=1}^{m} \left[\frac{1}{2}\ln(|\Sigma|) - \mathbb{I}\{y^{(i)}=0\}\left(f(x^{(i)},\mu_0,\Sigma) + \ln(1-\phi)\right) - \mathbb{I}\{y^{(i)}=1\}\left(f(x^{(i)},\mu_1,\Sigma) + \ln(\phi)\right) \right] \right), \tag{2.77}$$

where $f(x^{(i)},\mu_j,\Sigma) \doteq \frac{1}{2}(x^{(i)}-\mu_j)^\top\Sigma^{-1}(x^{(i)}-\mu_j)$ was defined. Now, the expression $l(\theta)$ can be optimized with respect to every model parameter $\theta = \{\mu_0,\mu_1,\Sigma,\phi\}$ by means of calculating the partial derivative with respect to every parameter, setting it to zero and solving for the parameter. Thus, one obtains

$$\begin{aligned}
\phi &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\{y^{(i)}=1\}, \\
\mu_0 &= \frac{\sum_{i=1}^{m} \mathbb{I}\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{m} \mathbb{I}\{y^{(i)}=0\}}, \\
\mu_1 &= \frac{\sum_{i=1}^{m} \mathbb{I}\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^{m} \mathbb{I}\{y^{(i)}=1\}}, \\
\Sigma &= \frac{1}{m}\left(\mathbb{I}\{y^{(i)}=0\}\sum_{i=1}^{m}(x^{(i)}-\mu_0)(x^{(i)}-\mu_0)^\top + \mathbb{I}\{y^{(i)}=1\}\sum_{i=1}^{m}(x^{(i)}-\mu_1)(x^{(i)}-\mu_1)^\top \right).
\end{aligned} \tag{2.78}$$

After finding the optimal parameters $\hat{\theta} = \{\hat{\mu}_0, \hat{\mu}_1, \hat{\Sigma}, \hat{\phi}\}$, one can then make a classification of a new sample $x^{(s)}$ by means of calculating

$$\begin{aligned}p(y=1|x^{(s)};\hat{\theta}) &= p(x^{(s)}|y=1;\hat{\theta})p(y=1;\hat{\theta}) \\ &= \frac{\hat{\phi}}{(2\pi)^{n/2}|\hat{\Sigma}|^{1/2}} \exp\left(-\frac{(x^{(s)}-\hat{\mu}_1)^\top\hat{\Sigma}^{-1}(x^{(s)}-\hat{\mu}_1)}{2}\right).\end{aligned} \tag{2.79}$$

If $p(y=1|x^{(s)};\hat{\theta}) > 0.5$ for a sample $x^{(s)}$, then $x^{(s)}$ belongs to class 1, whereas otherwise $x^{(s)}$ belongs to class 0.

Note, that $p(y=1|x^{(s)};\theta)$ can be written as a sigmoid function with an appropriate θ, which is a function of the parameters μ_0,μ_1, Σ and ϕ. Therefore, the Gaussian discriminative analysis (GDA) is very similar to logistic regression, makes however stronger assumptions. Therefore, GDA is better than logistic regression, where the GDA assumptions are fullfilled, but if that is not the case, logistic regression will yield better results.

31. **Q** — How does the naive Bayes approach work?

A — Consider a problem where one has a vector $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ of zeros and ones. Such a vector could for example contain a one at x_j, if the j-th word of a dictionary containing n words is in a checked text; say, for a spam/non-spam classification. Correspondingly, one has a class label $y \in \{1, \ldots, K\}$ for K classes; for example $K=1$ could denote work-related mails, $K=2$ spam and so on. Suppose furthermore, that one has at disposal a training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1,\ldots,m}$.

In this case, a naive Bayes algorithm can be applied. It is an example of a generative approach; thus we model $p(y|x;\theta)$ based on the Bayes theorem

$$p(y|x;\theta) \propto p(x|y;\theta)p(y;\theta), \tag{2.80}$$

where θ are the model parameters of the naive Bayes classifier. Suppose, that we have a weights vector $\theta = (\phi_{1|y=1}, \ldots, \phi_{1|y=K}, \ldots, \phi_{n|y=1}, \ldots, \phi_{n|y=K}, \phi_{y=1}, \ldots, \phi_{y=K})^\top$ containing $N = nK + K = K(n+1)$ weights. If one has for example $n = 9999$ words in a dictionary and $K = 10$ classes to classify incoming emails, one would have $N = K(n+1) = 10(9999+1) = 10^5$ model parameters.

Now, the probability $p(x_i = 1|y=k)$ that an object contains element i, given that it belongs to class k, can be given by $p(x_i = 1|y=k;\theta) = \phi_{i|y=k}$. Correspondingly, the probability $p(x_i = 0|y=k;\theta)$ must be given as $p(x_i = 0|y=k;\theta) = 1 - \phi_{i|y=k}$. Furthermore, the probability $p(y=k;\theta)$ that an object belongs to class k is given by $p(y=k;\theta) = \phi_{y=k}$. Written as sole expressions, the probabilities $p(x_i|y;\theta)$ and $p(y;\theta)$ can be given as

$$p(x_i|y;\theta) = \prod_{j=1}^{K} \phi_{i|y=j}^{\mathbb{I}\{x_i=1,y=j\}}(1-\phi_{i|y=j})^{\mathbb{I}\{x_i=0,y=j\}}, \qquad p(y;\theta) = \prod_{j=1}^{K} \phi_{y=j}^{\mathbb{I}\{y=j\}}. \tag{2.81}$$

The naive Bayes assumption is that the events $x_i|y=k$ and $x_j|y=k$ are independent of one another for any $i, y \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, K\}$. This allows us to write $p(x|y;\theta) = p(x_1, \ldots, x_n|y;\theta) = \prod_{i=1}^n p(x_i|y;\theta)$. Thus, we have

$$p(y|x;\theta) \propto p(x|y;\theta)p(y;\theta) = \prod_{i=1}^{n}\prod_{j=1}^{K} \phi_{y=j}^{\mathbb{I}\{y=j\}} \phi_{i|y=j}^{\mathbb{I}\{x_i=1,y=j\}}(1-\phi_{i|y=j})^{\mathbb{I}\{x_i=0,y=j\}}. \tag{2.82}$$

The expression, one would like to maximize, is

$$L(\theta) = \sum_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta). \tag{2.83}$$

This is equivalent to minimizing the log-likelihood $l(\theta) \doteq -\sum_{k=1}^m \ln\left[p(y^{(i)}|x^{(i)};\theta)\right]$ as

$$l(\theta) = -\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{K}\left[\mathbb{I}\{y^{(k)}=j\}\ln\left(\phi_{y=j}\right) + \mathbb{I}\{x_i^{(k)}=1, y^{(k)}=j\}\ln\left(\phi_{i|y=j}\right) + \mathbb{I}\{x_i^{(k)}=0, y^{(k)}=j\}\ln\left(1-\phi_{i|y=j}\right)\right]. \tag{2.84}$$

Calculating the derivatives with respect to all model parameters contained in θ, setting the derivatives to zero and solving for the parameter, one would obtain the optimal model parameters $\hat{\theta}$.

We now consider a case, where we have only two classes $y \in \{0, 1\}$. In that case, the log-likelihood to minimize simplifies to

$$\begin{aligned} l(\theta) = &-\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=0}^{1}\left[\mathbb{I}\{x_i^{(k)} = 1, y^{(k)} = j\}\ln\left(\phi_{i|y=j}\right) + \mathbb{I}\{x_i^{(k)} = 0, y^{(k)} = j\}\ln\left(1-\phi_{i|y=j}\right)\right] \\ &-\sum_{k=1}^{m}\left[\mathbb{I}\{y^{(k)} = 0\}\ln\left(\phi_y\right) + \mathbb{I}\{y^{(k)} = 1\}\ln\left(1-\phi_y\right)\right], \end{aligned} \tag{2.85}$$

where $\phi_{y=0} \doteq \phi_y$ was redefined. Minimizing $l(\theta)$ leads to the optimal model parameters

$$\hat{\theta} = (\hat{\phi}_{1|y=0}, \ldots, \hat{\phi}_{n|y=0}, \hat{\phi}_{1|y=1}, \ldots, \hat{\phi}_{n|y=1}, \hat{\phi}_y)^\top$$

as

$$\begin{aligned} \hat{\phi}_{i|y=0} &= \frac{\sum_{k=1}^{m}\mathbb{I}\{x_i^{(k)} = 1, y^{(k)} = 0,\}}{\sum_{k=1}^{m}\mathbb{I}\{y^{(k)} = 0\}}, \\ \hat{\phi}_{i|y=1} &= \frac{\sum_{k=1}^{m}\mathbb{I}\{x_i^{(k)} = 1, y^{(k)} = 1,\}}{\sum_{k=1}^{m}\mathbb{I}\{y^{(k)} = 1\}}, \\ \hat{\phi}_{y} &= \frac{\sum_{k=1}^{m}\mathbb{I}\{y^{(k)} = 0\}}{m}. \end{aligned} \tag{2.86}$$

Now that the optimal model parameters are known, one can make a prediction on a new sample $x^{(s)}$ whether it belongs to class $y = 0$ or class $y = 1$ by means of calculating

$$p(y = 1|x^{(s)}; \hat{\theta}) = \frac{p(x^{(s)}|y = 1; \hat{\theta})p(y = 1; \hat{\theta})}{p(x^{(s)}; \hat{\theta})} = \frac{p(x^{(s)}|y = 1; \hat{\theta})p(y = 1; \hat{\theta})}{p(x^{(s)}|y = 1; \hat{\theta})p(y = 1; \hat{\theta}) + p(x^{(s)}|y = 0; \hat{\theta})p(y = 0; \hat{\theta})}. \tag{2.87}$$

If $p(y = 1|x^{(s)}; \hat{\theta}) > 0.5$, then $x^{(s)}$ belongs to class $y = 1$; if not, $x^{(s)}$ belongs to class $y = 0$.

32. **Q** — What is Laplace smoothing in the context of the naive Bayes approach?

A — Consider a random variable z taking values $\{1, \ldots, k\}$. One can parametrize this using probabilities $\phi_i = p(z = i)$. Given a set of m independent observations $\{z^{(1)}, \ldots, z^{(m)}\}$, the maximum likelihood estimates $\hat{\phi}_j$ for this case are then derived to be

$$\hat{\phi}_j = \frac{\sum_{i=1}^{m}\mathbb{I}\{z^{(i)} = j\}}{m}. \tag{2.88}$$

If the training dataset however does not contain a single instance where $z = l$ for example, the corresponding probability ϕ_l will be estimated as zero, leading to a decision problem for the model, because both nominator and denominator of the Bayes theorem to generate the posterior will be zero.

This problem can be solved by so-called Laplace smoothing. One just adds to the maximum likelihood estimators $+1$ in the nominator, and $+k$ in the denominator, where k is the number of possible values for z. Therefore, the modified maximum likelihood estimators will be given as

$$\hat{\phi}_j = \frac{\sum_{i=1}^{m}\mathbb{I}\{z^{(i)} = j\} + 1}{m + k}. \tag{2.89}$$

It can be shown, that these are still valid probailities, as

$$\sum_{j=1}^{k} \hat{\phi}_j = \sum_{j=1}^{k} \left(\frac{\sum_{i=1}^{m} \mathbb{I}\{z^{(i)} = j\} + 1}{m+k} \right) = \frac{1}{m+k} \sum_{j=1}^{k} \left(\sum_{i=1}^{m} \mathbb{I}\{z^{(i)} = j\} + 1 \right)$$
$$= \frac{1}{m+k} \left(k + \underbrace{\sum_{i=1}^{m} \mathbb{I}\{z^{(i)} = 1\} + \cdots + \sum_{i=1}^{m} \mathbb{I}\{z^{(i)} = k\}}_{=m} \right) = 1 \tag{2.90}$$

shows. Notice, that if the value $z = l$ never occurs in the training dataset, the probability ϕ_l will be estimated as $\phi_l = 1/(m+k)$; the most unlikely, but not impossible event.

2.5 Support vector machines

33. **Q** — What is the general idea behind support vector machines?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{-1, 1\}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \ldots, m\}$ be a training dataset.

Now, the idea of a support vector machine is that one tries to find a hyperplane (dimension $\mathbb{R}^{n-1}$) in the feature space ($\mathbb{R}^n$), that separates positives ($y = +1$) from negatives ($y = -1$). That means, we have a classifier

$$h_{w,b}(x) = g(w^\top x + b), \qquad g(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}, \tag{2.91}$$

where w is a vector and b a scalar; w and b comprise the model parameters of the support vector machine. The separating hyperplane between positives and negatives is given by the equation $w^\top x + b = 0$. Graphically, one can depict this a seen in fig. 4.

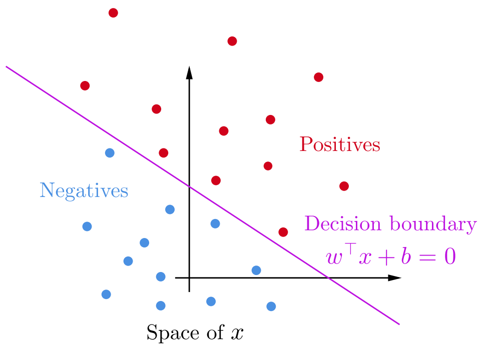


Figure 4: Schematic view of the working principle of a support vector machine.

34. **Q** — Where does the name "support vector machine" come from?

A — The term "vector" comes from the inputs x, which are to be classified to belong either to the class $y = -1$ (negatives) or to the class $y = +1$ (positives). As a matter of fact, mostly the samples $x^{(i)}$ closest to the decision boundary actually decide on the location of it; only those vectors support the decision on the decision boundary - this is where the term "support" comes from.

35. **Q** — What are the geometrical and functional margins with respect to the support vector machines (SVMs)? Why are both functional aswell as geometrical margin necessary?

A — In SVMs, the decision boundary for the SVM to decide if a new sample $x^{(j)}$ belongs to the category $y = +1$ or $y = -1$ is given by $w^\top x + b = 0$, where w is a vector and b a scalar. All vectors x that fulfill this condition live on the decision boundary (hyperplane). If the new sample $x^{(j)}$ however is not part of the decision hyperplane, then the quantity $w^\top x + b$ will either be a positive or a negative number. If it is a negative number, $x^{(j)}$ will belong to the negatives, while if it is positive, it will belong to the positives, so we have

$$w^\top x^{(j)} + b \geq 0 \quad \Rightarrow \quad y^{(j)} = 1, \qquad w^\top x^{(j)} + b < 0 \quad \Rightarrow \quad y^{(j)} = -1. \tag{2.92}$$

Thus one can make the number $w^\top x^{(j)} + b$ positive for any sample by means of defining the functional margin $\tilde{\gamma}^{(j)}$ for a particular sample as

$$\tilde{\gamma}^{(j)} = y^{(j)} \left(w^\top x^{(j)} + b \right). \tag{2.93}$$

The functional margin gives a metric of distance of a sample $x^{(j)}$ from the decision hyperplane and thus is a measure on the confidence of the decision; if the functional margin for a sample is very large, the decision to which class it belongs will be a confident one. However, the functional margin definition is insensitive to scaling factors; therefore, the definition of an alternative margin - the so-called geometrical margin - is necessary. The minimal functional margin $\tilde{\gamma}$ is given by

$$\tilde{\gamma} = \min_{j=1,\dots,m} \tilde{\gamma}^{(j)} \tag{2.94}$$

for a set of m training samples.

Consider fig. 5 for a visualization of the meaning of the geometrical margin. The weights vector w is always orthogonal to the decision boundary, as $w \cdot \left(w^\top x + b \right) = 0$ for any x on the decision boundary. Hence, one can measure the distance of a sample $x^{(i)}$ from the decision boundary along w. For any $x^{(i)}$, one can hence write $w^\top \left(x^{(i)} - \alpha^{(i)} w \right) + b = 0$, where $\alpha^{(i)}$ is a scalar needed to scale w appropriately to obtain the orthogonal distance of $x^{(i)}$ to the decision boundary. Solving for $\alpha^{(i)}$ yields

$$\alpha^{(i)} = \frac{w^\top x^{(i)} + b}{\|w\|^2}. \tag{2.95}$$

The geometric margin $\gamma^{(j)}$ for any sample $x^{(j)}$ as the positive orthogonal distance of the sample

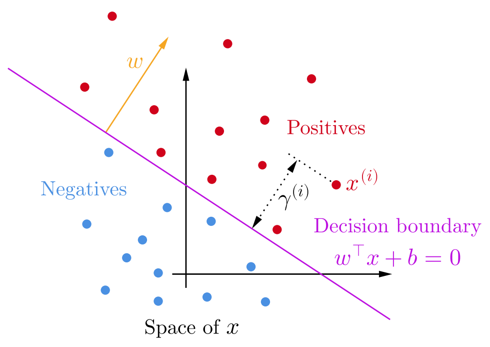


Figure 5: Visualization of the construction of the geometrical margin in SVMs.

from the boundary hyperplane can therefore be defined as

$$\gamma^{(j)} = y^{(j)} \alpha^{(j)} \|w\| = \left(\frac{w^\top x^{(j)}}{\|w\|} + \frac{b}{\|w\|} \right) y^{(j)}. \tag{2.96}$$

The minimal geometric margin γ is accordingly defined by

$$\gamma = \min_{j=1,\dots,m} \gamma^{(j)}. \tag{2.97}$$

Note, that in the case of $\|w\| = 1$, functional and geometric margin are identical.

The reason why both functional and geometric margin are defined is because only one of them is sensitive to the actual distance of a sample from the decision boundary.

36. **Q** — How is a support vector machine mathematically described?

A — Mathematically speaking, a support vector machine is all about tuning the model parameters w and b such that the minimum geometric margin is maximized for the training dataset. That is, if one has a dataset of m pairs of $x^{(i)}$ and $y^{(i)}$, then the optimal model parameters w^* and b^* need to be chosen such that

$$w^*, b^* \quad \text{such that} \quad \max_{w,b} \left(\min_{w,b,i=1,\dots,m} \left[\gamma^{(i)} \right] \right). \tag{2.98}$$

The easiest way to define the optimization problem would be to require

$$\max_{w,b} (\gamma) \quad \text{such that} \quad y^{(i)} \left(w^\top x^{(i)} + b \right) \geq \gamma \quad \forall i = 1, \dots, m, \quad \|w\| = 1. \tag{2.99}$$

Here, the geometric margin is maximized, whereas the condition $\|w\| = 1$ ensures that the geometric margin is identical to the functional margin. This ensures, that both functional and geometric margin are maximized. Because functional and geometric margin are related as $\gamma = \tilde{\gamma}/\|w\|$, one can also formulate the optimization problem as

$$\max_{w,b} \left(\frac{\tilde{\gamma}}{\|w\|} \right) \quad \text{such that} \quad y^{(i)} \left(w^\top x^{(i)} + b \right) \geq \tilde{\gamma} \quad \forall i = 1, \dots, m. \tag{2.100}$$

This still ensures, that both the functional aswell as geometric margins are maximized, but the condition $\|w\| = 1$ was dropped. Still, the problem can be simplified by setting $\tilde{\gamma} = 1$. This can be done, because one can scale the functional margin $\tilde{\gamma}$ by any constant without effect. Thus, $\tilde{\gamma}$ is scaled appropriately such that $\tilde{\gamma} = 1$. Further, we notice that maximizing $\frac{1}{\|w\|}$ will yield the same result as minimizing $\|w\|$ and thus also minimizing $\frac{1}{2}\|w\|^2$. Therefore, the final form of any SVM task is to optimize

$$w^*, b^* = \arg\min_{w,b} \left(\frac{1}{2} \|w\|^2 \right) \quad \text{such that} \quad y^{(i)} \left(w^\top x^{(i)} + b \right) \geq 1 \quad \forall i = 1, \dots, m \tag{2.101}$$

where w^* and b^* are the optimal SVM model parameters.

37. **Q** — How does the Lagrange multiplier method for minimization of convex functions with additional constraints work?

A — A constrained optimization problem of the form

$$\min_{w} f(w), \quad \text{such that} \quad g_i(w) \leq 0, \quad i = 1, \dots, k, \quad h_j(w) = 0, \quad j = 1, \dots, l, \tag{2.102}$$

can be solved using the method of Lagrange multipliers. The constraints on the $g_i(w)$ and $h_j(w)$ are known as the primal constraints. One can define the generalized Lagrangian

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{l} \beta_j h_j(w). \tag{2.103}$$

One would like to find the optimal model parameters w^*, α^* and β^* for this problem. That is to say, one would like to minimize $\mathcal{L}(w, \alpha, \beta)$ with respect to w, whereas at the same time the primal conditions need to be satisfied. This can be ensured by defining the so-called primal quantity as

$$\theta_P(w) = \max_{\alpha,\beta:\alpha_i\geq 0} [\mathcal{L}(w,\alpha,\beta)] = \max_{\alpha,\beta:\alpha_i\geq 0} \left[f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{l} \beta_j h_j(w) \right]. \tag{2.104}$$

Here, one maximizes $\mathcal{L}$ with respect to α and β, which makes the resulting quantity after the maximization only a function of w. Defining $\theta_P(w)$ helps to ensure the primal constraints in the optimization process. One can see, that

$$\theta_P(w) = \begin{cases} f(w), & \text{primal constraints satisfied} \\ \infty, & \text{primal constraints not satisfied} \end{cases}. \tag{2.105}$$

Because one wants to minimize $f(w)$ with respect to w in the end, one can now minimize $\theta_P(w)$ with respect to w, where $\theta_P(w)$ ensures that the primal constraints are satisfied; thus, a solution corresponding to the optimal value p^* of the so-called primal optimization is obtained by

$$p^* = \min_w [\theta_P(w)] = \min_w \left(\max_{\alpha,\beta:\alpha_i\geq 0} [\mathcal{L}(w,\alpha,\beta)] \right). \tag{2.106}$$

Another approach is to define the dual quantity

$$\theta_D(\alpha,\beta) = \min_w [\mathcal{L}(w,\alpha,\beta)] = \min_w \left[f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{j=1}^{l} \beta_j h_j(w) \right]. \tag{2.107}$$

In the dual approach, one first minimizes the Lagrangian with respect to w. This does ensure that one finds a global minimum (if there is one) of the objective function $f(w)$, but does not ensure yet that the primal constraints are satisfied. Hence, to satisfy the primal constraints, one must afterwards maximize with respect to α and β to obtain another solution corresponding to the optimal value d^* of the optimization problem; this solution to the so-called dual optimization problem is given by

$$d^* = \max_w [\theta_D(w)] = \max_{\alpha,\beta:\alpha_i\geq 0} \left(\min_w [\mathcal{L}(w,\alpha,\beta)] \right). \tag{2.108}$$

Now, both approaches are valid, as the relation

$$d^* \leq p^* \tag{2.109}$$

holds, because for any function $f(a,b)$, the relation $\min_a (\max_b [f(a,b)]) \leq \max_b (\min_a [f(a,b)])$ holds. If $f(w)$ and $g_i(w)$ are convex for all w and i and if all the $h_i(w)$ are affine, then it holds that the solutions to the primal and dual problems are identical; thus

$$d^* = \max_{\alpha,\beta:\alpha_i\geq 0} \left(\min_w [\mathcal{L}(w,\alpha,\beta)] \right) \overset{\text{Conditions}}{=} \min_w \left(\max_{\alpha,\beta:\alpha_i\geq 0} [\mathcal{L}(w,\alpha,\beta)] \right) = p^*. \tag{2.110}$$

Under the given conditions, there exist optimal values w^*, α^* and β^* to the optimization problem; this solution satisfies the so-called Karush-Kuhn-Tucker (KKT) conditions

$$\left.\frac{\partial \mathcal{L}(w,\alpha,\beta)}{\partial w_i}\right|_{w^*,\alpha^*,\beta^*} = 0, \quad i = 1,\dots,n, \tag{2.111}$$

$$\left.\frac{\partial \mathcal{L}(w,\alpha,\beta)}{\partial \beta_i}\right|_{w^*,\alpha^*,\beta^*} = 0, \quad i = 1,\dots,l, \tag{2.112}$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1,\dots,k, \tag{2.113}$$

$$g_i(w^*) \leq 0, \quad i = 1,\dots,k, \tag{2.114}$$

$$\alpha_i^* \geq 0, \quad i = 1,\dots,k. \tag{2.115}$$

The third KKT condition, called the dual complementary condition, is insofar interesting as it implies $g_i(w^*) = 0$ if $\alpha_i^* > 0$. In the context of $\mathcal{L}$ being the Lagrangian for support vector machines, the complementary condition implies that only the α_i^* which correspond to the functional margin $y^{(i)}\left(w^{*\top}x^{(i)} + b^*\right) = 1$ are non-zero, because for those i's, $g_i(w^*) = 0$ holds (only for those i's, it is possible for α_i^* to be non-zero, because the product $g_i(w^*)\alpha_i^*$ will be still zero).

38. **Q** — How can a basic support vector machine be implemented?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{-1, 1\}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \ldots, m\}$ be a training dataset. Following the optimization procedure for convex functions with additional constraints, one can define the Lagrangian and the constraints for a basic support vector machine as

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i g_i(w), \quad g_i(w) = 1 - y^{(i)}\left(w^\top x^{(i)} + b\right) \leq 0, \quad i = 1, \ldots, m. \tag{2.116}$$

Now, one can go about imposing the KKT conditions. First, one can calculate the derivatives with respect to w and b, set them to zero and solve for w and b. One obtains

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} y^{(i)}x^{(i)} \overset{\text{KKT}}{=} 0 \quad \Leftrightarrow \quad w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}. \tag{2.117}$$

Furthermore calculating $\nabla_b \mathcal{L}$ yields

$$\nabla_b \mathcal{L}(w, b, \alpha) = -\sum_{i=1}^{m} \alpha_i y^{(i)} \overset{\text{KKT}}{=} 0 \quad \Leftrightarrow \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \tag{2.118}$$

Inserting these findings into the original Lagrangian yields after some simplifications the simplified Lagrangian

$$\mathcal{L}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^\top x^{(j)} \doteq \theta_D(\alpha), \tag{2.119}$$

where the minimization in w and b was already performed; that is, the maximization in α remains to be carried out. The optimal solution parameters $\alpha^* = (\alpha_1^*, \ldots, \alpha_m^*)^\top$ are therefore obtained by

$$\alpha^* = \arg \max_{\substack{\alpha_i \geq 0 \\ \sum_{i=1}^m \alpha_i y^{(i)} = 0}} \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^\top x^{(j)}\right]. \tag{2.120}$$

Once, the optimal parameters α^* have been found, the vector w^* can be calculated as

$$w^* = \sum_{i=1}^{m} \alpha_i^* y^{(i)} x^{(i)}. \tag{2.121}$$

Finally, the parameter b^* is missing. From the basic definition of a support vector machine problem, it is evident that

$$w^{*\top}x^{(i)} + b \geq 1 \quad \text{for} \quad y^{(i)} = +1, \qquad w^{*\top}x^{(i)} + b \leq -1 \quad \text{for} \quad y^{(i)} = -1. \tag{2.122}$$

This means that one can write the inequality

$$\max_{i:y^{(i)}=1} \left(1 - w^{*\top}x^{(i)}\right) \leq b \leq \min_{i:y^{(i)}=-1} \left(-1 - w^{*\top}x^{(i)}\right). \tag{2.123}$$

Taking the mean value between upper and lower bound for b and making use of $\min(a) = -\min(-a)$ and $\max(a) = -\max(-a)$ for a generic parameter a, one obtains the optimal value b^* as

$$b^* = -\frac{1}{2}\left[\max_{i:y^{(i)}=-1}\left(w^{*\top}x^{(i)}\right) + \min_{i:y^{(i)}=1}\left(w^{*\top}x^{(i)}\right)\right]. \tag{2.124}$$

Having found the parameters α^*, w^* and b^*, one can now perform classifications on new samples $x^{(k)}$ by means of

$$y^{(k)} = \begin{cases} +1, & w^{*\top}x^{(k)} + b^* > 0 \\ -1, & w^{*\top}x^{(k)} + b^* < 0 \end{cases}. \tag{2.125}$$

39. **Q** — How can problems be handled with SVMs, which are not linearly separable and potentially feature outliers?

A — Let $x \in \mathbb{R}^n$ be a feature vector and $y \in \{-1, 1\}$. Let furthermore $(x^{(i)}, y^{(i)})$ with $i \in \{1, \ldots, m\}$ be a training dataset. It is however possible, that the datapoints are not linearly separable due to outliers or more general patterns, that make the problem non-linear.

Still, support vector machines can be used in this case; one however needs to make some adjustments to the equations. A first strategy is to introduce so-called regularization parameters ξ_i to allow for outliers or non-linearly separable points. This leads to the margin condition becoming $y^{(i)}\left(w^\top x^{(i)} + b\right) \geq 1 - \xi_i$ for all $i = 1, \ldots, m$. Graphically, one can picture a sample $x^{(i)}$ (outlier) not being on the "correct" side by a distance of ξ_i. This minor adjustment leads to the Lagrangian of the optimization problem becoming

$$\mathcal{L}(w, b, \alpha, \xi, r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(w^\top x^{(i)} + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{m} r_i\xi_i \tag{2.126}$$

with $\alpha_i, r_i \geq 0$ as Lagrange multipliers. One can go about solving the system using the KKT conditions, which leads to solving the problem

$$\alpha^* = \arg\max_{\substack{0 \leq \alpha_i \leq C \\ \sum_{i=1}^{m}\alpha_i y^{(i)} = 0}}\left[\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}(x^{(i)})^\top x^{(j)}\right]. \tag{2.127}$$

Having obtained a solution α^*, one can further go about calculating w^* and b^* just as before. A second strategy, which can be combined with the first strategy, one can map the features $x \mapsto \tilde{x} = \phi(x)$ to a higher dimensional space, as they might become linearly separable. This principle is visualized in fig. 6. The same optimization framework is applicable as for basic SVMs;

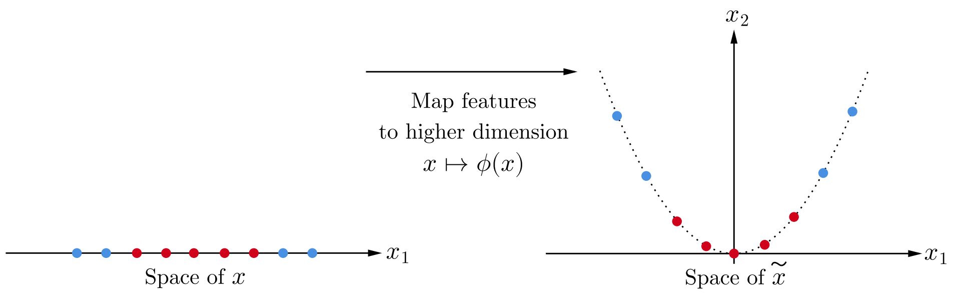


Figure 6: Principle of mapping features x to a higher dimension with an appropriate mapping function $\phi(x)$, such that the features might become separable using a hyperplane in the new space.

one just needs to replace every $x^{(i)}$ with $\phi(x^{(i)})$. Also, the prediction on a new sample $x^{(i)}$ needs to be then made with

$$y^{(k)} = \begin{cases} +1, & w^{*\top}\phi\left(x^{(k)}\right) + b^* > 0 \\ -1, & w^{*\top}\phi\left(x^{(k)}\right) + b^* < 0 \end{cases}. \tag{2.128}$$

40. **Q** — What is a kernel (matrix) in the context of SVMs and why is it useful?

A — A key quantity occurring in the equations of SVMs is the inner product $(x^{(i)})^\top x^{(j)}$ for $i, j = 1, \dots, m$ for a dataset containing m samples $(x^{(i)}, y^{(i)})$. Now, if one uses feature mapping, this inner product becomes

$$(x^{(i)})^\top x^{(j)} \mapsto \phi(x^{(i)})^\top \phi(x^{(j)}) \doteq K(x^{(i)}, x^{(j)}). \tag{2.129}$$

Because this is a re-occurring expression in the SVM equations, one defines this quantity as the so-called kernel $K(x, z)$. One can use and experiment with different kernels when using SVMs to try to separate non-linearly separable datasets. Oftentimes, it is very useful to calculate the kernel directly, instead of first calculating $\phi(x)$ and $\phi(z)$ and then taking the inner product $\phi(x)^\top \phi(z)$.

Hereby, the Mercer theorem is very useful: If a kernel $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ written as kernel matrix for input vectors $\{x^{(1)}, \dots, x^{(m)}\}$ with $m < \infty$ is positive semi-definite, then one can write the kernel in the form $K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^\top \phi(x^{(j)})$. Thus, if the Mercer theorem is fulfilled for some chosen kernel, then one has found an "allowed" feature mapping and potentially a way to calculate the inner product of the feature mappings much more efficiently than explicitly defining and calculating the feature mappings and then performing the inner product.

41. **Q** — How can the fundamental optimization problem for support vector machines be solved?

A — Given a dataset $(x^{(i)}, y^{(i)})$ of feature vectors $x \in \mathbb{R}^n$ and labels $y \in \{-1, +1\}$ with $i = 1, \dots, m$, the central problem to solve in SVM algorithms is

$$\alpha^* = \arg \max_{\alpha:\text{Constraints}} \underbrace{\left[\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^\top x^{(j)} \right]}_{\doteq W(\alpha)}. \tag{2.130}$$

Here, the constraints can e.g. be $\alpha_i \geq 0$ for all $i \in \{1, \dots, m\}$. The so-called SMO (sequential minimal optimization) algorithm can be used to solve that minimization/maximization problem.

The SMO algorithm works by means of sequentially fixing all but two quantities α_i and α_j of the function $W(\alpha)$ which is to be maximized. Fixing all parameters but α_i does not work because of the constraint $\sum_{j=1}^m y^{(j)} \alpha_j = 0$ which yields

$$\alpha_i y^{(i)} + \underbrace{\sum_{j \neq i} \alpha_j y^{(j)}}_{\doteq \gamma} = 0 \quad \Leftrightarrow \quad \alpha_i = -\gamma \tag{2.131}$$

and hence leaves no room for optimization. So all but two parameters α_i and α_j are fixed. Imposing the constraint $\sum_l \alpha_l y^{(l)} = 0$ yields

$$\alpha_i y^{(i)} + \alpha_j y^{(j)} + \sum_{k \neq i,j} \alpha_k y^{(k)} = 0 \quad \Leftrightarrow \quad \alpha_i = -\alpha_j y^{(i)} y^{(j)} - y^{(j)} \sum_{k \neq i,j} \alpha_k y^{(k)}, \tag{2.132}$$

where the parameters α_i and α_j are constrained to a box $[0, C] \times [0, C]$ by the condition $0 \leq \alpha_i \leq C$ for all $i = 1, \dots, m$, as fig. 7 shows. Now, one can substitute α_i in $W(\alpha_1, \dots, \alpha_i, \alpha_j, \dots, \alpha_m)$ with the above expression, which yields a function $W(\alpha_j)$ of only one variable α_j, because all other parameters are fixed. This means, that one now finds α_j^* by means of

$$\alpha_j^* = \arg \max_{\alpha_j} [W(\alpha_j)] \tag{2.133}$$

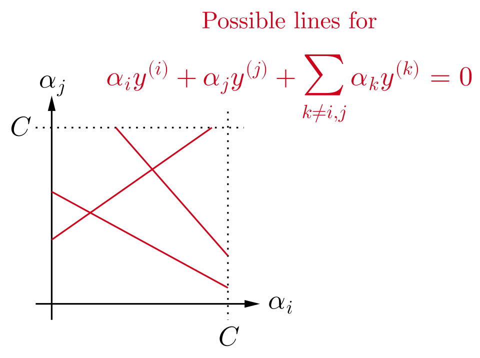

Possible lines for
$\alpha_i y^{(i)} + \alpha_j y^{(j)} + \sum_{k\neq i,j} \alpha_k y^{(k)} = 0$
α_j
C
α_i
C

Figure 7: Possible values for the parameters α_i and α_j subject to the constraint $0 \leq \alpha_i \leq C$ for all $i = 1, \ldots, m$.

using for example a gradient ascent (or descent) method. Then, one can go about calculating α_j^* using eq. (2.132). Thus, the optimization in two parameters has been carried out. The SMO algorithm then iterates pair-wise through all other parameters, leading to the optimal solution α^*.

2.6 Decision trees

42. **Q** — What is a decision tree and what is the principle of the decision tree method?

A — A decision tree is a very intuitive and useful approach for classification tasks. Consider a dataset $\{(x^{(i)}, y^{(i)})\}_{i=1,\ldots,m}$ of pairs of feature vectors $x \in \mathbb{R}^n$ and labels $y \in \{1, \ldots, K\}$. A decision tree works by the principle of concatenating binary decisions to partition a parent partition R_p into two child partitions R_1 and R_2. The goal is to finally have the samples belonging to each class separated in separate partitions.

Visually, decision trees can be explained as seen in fig. 8. Given a parent region R_p consisting

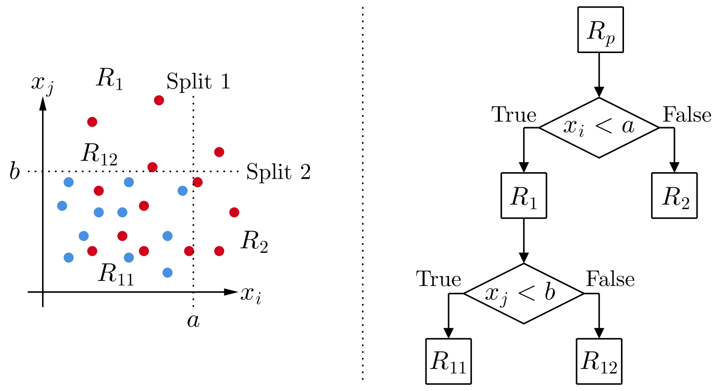


Figure 8: Left panel: Decision tree working principle shown in a plane for $y \in \{1, 2\}$ and features x_i and x_j. Right panel: The same situation as seen in the left panel shown as decision tree.

of samples x of feature vectors, the child regions R_1 and R_2 given some threshold value t applied to feature x_j are calculated by

$$\begin{aligned} R_1 &= \{x | x_j < t, x \in R_p\}, \\ R_2 &= \{x | x_j \geq t, x \in R_p\}. \end{aligned} \tag{2.134}$$

Regarding the right panel of fig. 8, every instance where a decision is made is called a leaf node (rhomboid structure). The depth of a decision tree is furthermore given by the maximal number of concatenated decisions (maximal number of concatenated rhomboid structures). Finally, we understand the cardinality of a certain leaf (square box) as leaf size.

In order to choose the splits, one requires a loss function $L(R)$, where R is a region. This loss can generally be understood as the proportion of misclassified samples in R. Therefore, making a split one would like to decrease the proportion of misclassified samples in the splits R_1 and R_2 compared to a parent region R_p. One can define the quantity $D(R_p, R_1, R_2)$ denoting the decrease in terms of loss going from a parent region R_p to two child regions R_1 and R_2 as

$$D(R_p, R_1, R_2) = L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}. \tag{2.135}$$

As a loss function, the cross-entropy loss

$$L_{ce}(R) = -\sum_c p_c(R) \log_2 \left[p_c(R)\right], \tag{2.136}$$

is used, where $p_c(R)$ is the proportion of of c-classified samples in R, namely

$$p_c(R) = \frac{\text{Number of samples belonging to class } c \text{ in } R}{\text{Number of samples in } R}. \tag{2.137}$$

Using $L_{ce}(R)$ as a loss function ensures, that one always has a decrease $D(R_p, R_1, R_2)$ by splitting a parent region into two child regions.

There are three common stopping criteria for training a decision tree:

(1) Minimum leaf size: Do not split a region R anymore, if its cardinality falls below a threshold t.

(2) Maximum depth: Do not split a region R anymore, if a threshold t of decisions have already been made in succession to reach R.

(3) Maximum number of nodes: Do not split a region R anymore, if a maximum threshold t of decisions have been made.

Now, if a decision tree has been trained, there are a finite number of leafs R_i, $i = 1, \dots, q$ and leaf nodes in the tree. The decision tree may be pictures as a function

$$f(x) = \begin{cases} 1, & x \in R_1 \\ \vdots & \vdots \\ q, & x \in R_q \end{cases} \tag{2.138}$$

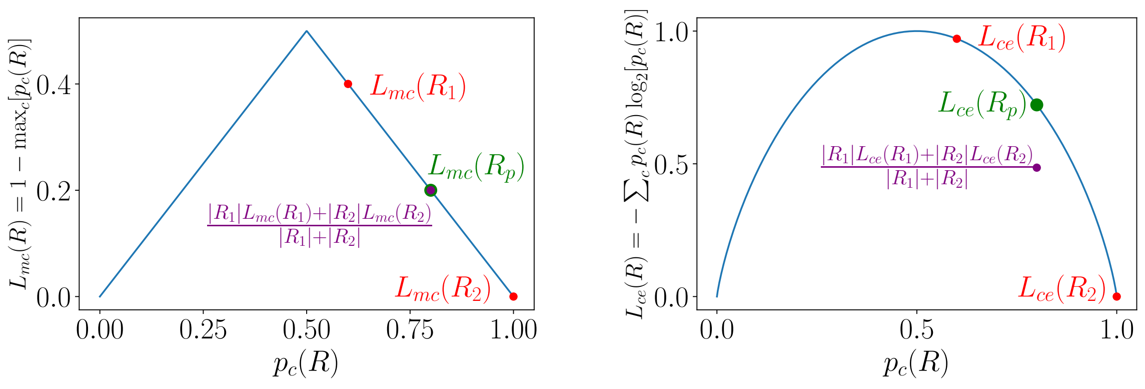
Given a new sample $x^{(l)}$, one feeds that sample to the decision tree and checks, in which leaf R_i it would end up; for example $f(x^{(l)}) = j$ for $j \in \{1, \dots, q\}$. Finally, one calculates $p_c(R_j)$ on the training data for all classes present in R_j, which will yield probabilities of the sample $x^{(l)}$ belonging to any of the classes c in R_j. The sample $x^{(l)}$ is then classified as belonging to the class of highest probability in R_j.

43. **Q** — What common two types of loss functions are there? What are the issues?

A — The two common types of classification losses for training decision trees are the misclassification loss and the cross-entropy loss. In practice however, only the cross-entropy loss is used, because the misclassification loss has the issue, that there is no guarantee that it decreases when making splits.

The misclassification loss is defined as

$$L_{mc}(R) = 1 - \max_c [p_c(R)], \tag{2.139}$$



(a) Misclassification loss for binary classification. **(b)** Cross-entropy loss for binary classification.

Figure 9: Loss functions for training of decision trees. The figure was produced by assuming $R_p = \{400+, 100-\}$, $R_1 = \{150+, 100-\}$ and $R_2 = \{250+, 0-\}$.

whereas the cross-entropy loss is given by

$$L_{ce}(R) = -\sum_c p_c(R)\log_2[p_c(R)]. \tag{2.140}$$

Both loss functions are visualized in fig. 9. As one can see from the figure, only for the cross-entropy loss, the decrease $D(R_p, R_1, R_2)$ in loss going from the parent R_p to the children R_1 and R_2 is guaranteed to decrease.

44. **Q** — What are the main advantages and disadvantages of the decision tree method?

A — The advantages of decision trees are the following:

(A1) Decision trees are easy to explain and interpretable.

(A2) Decision trees support categorical variables (features); one does not have to first convert categorical variables to numbers.

(A3) Decision trees are fast.

(A4) Decision trees can capture non-linear patterns.

The drawbacks of decision trees are the following:

(D1) Decision trees lead to a high variance in predictions; this is mainly due to the fact that decision trees tend to overfit, as they continue to split regions until each leaf is ideally pure. This leads to capture of just nose, instead of capturing the general pattern behind the data. Therefore, also predictions will vary wildly; hence they have large variance.

(D2) Decision trees cannot easily capture additive structure; that is, for example perform a split according to a linear function between two features. Many splits in a staircase-like manner are necessary to do that.

2.7 Ensembling methods

45. **Q** — What is the general idea of ensembling methods?

A — Text.

46. **Q** — What methods are there to decorrelate methods in an ensemble of methods? What is decorrelation for?

A — Text.

2.8 Regularization and model selection

47. **Q** — What is regularization?

A — Text.

48. **Q** — What are train, validation and test splits for?

A — Text.

49. **Q** — What is cross validation and how does it work?

A — Text.

50. **Q** — What is feature selection, when is it useful and how does it work?

A — Text.

51. **Q** — What is mutual information and why is it used as a score function? What happens, if the two input quantities to mutual information are independent?

A — Text.

3 Unsupervised learning

52. **Q** — What is unsupervised learning? What is the goal of unsupervised learning?

A — Text.

3.1 Clustering and k-means

53. **Q** — How does the k-means algorithm work?

A — Text.

54. **Q** — How do Gaussian mixture models work?

A — Text.

3.2 EM and factor analysis

55. **Q** — ?

A — Text.

3.3 PCA and ICA

56. **Q** — ?

A — Text.

4 Reinforcement learning

57. **Q** — ?

A — Text.