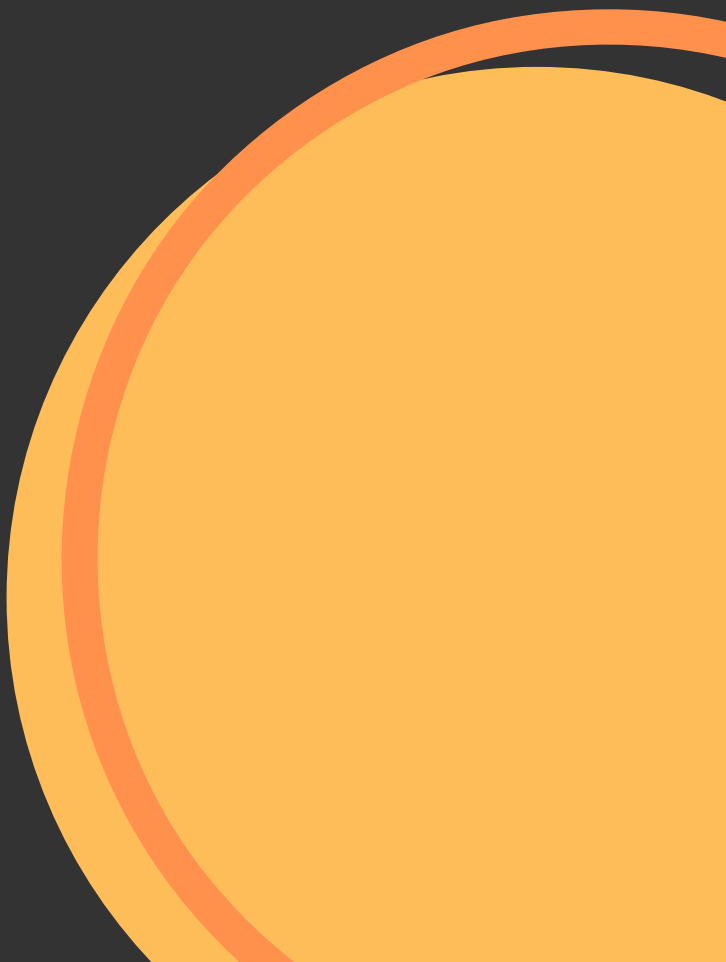
Andriy Burkov's

THE HUNDRED-PAGE MACHINE LEARNING BOOK



"All models are wrong, but some are useful."
— George Box

The book is distributed on the "read first, buy later" principle.

2 Notation and Definitions

2.1 Notation

Let's start by revisiting the mathematical notation we all learned at school, but some likely forgot right after the prom.

2.1.1 Data Structures

A *scalar* is a simple numerical value, like 15 or -3.25. Variables or constants that take scalar values are denoted by an italic letter, like x or a.

A *vector* is an ordered list of scalar values, called attributes. We denote a vector as a bold character, for example, $\mathbf{x}$ or $\mathbf{w}$. Vectors can be visualized as arrows that point to some directions as well as points in a multi-dimensional space. Illustrations of three two-dimensional vectors, $\mathbf{a} = [2, 3]$, $\mathbf{b} = [-2, 5]$, and $\mathbf{c} = [1, 0]$ are given in Figure 1. We denote an attribute of a vector as an italic value with an index, like this: $w^{(j)}$ or $x^{(j)}$. The index j denotes a specific *dimension* of the vector, the position of an attribute in the list. For instance, in the vector $\mathbf{a}$ shown in red in Figure 1, $a^{(1)} = 2$ and $a^{(2)} = 3$.

The notation $x^{(j)}$ should not be confused with the power operator, like this x^2 (squared) or x^3 (cubed). If we want to apply a power operator, say square, to an indexed attribute of a vector, we write like this: $(x^{(j)})^2$.

A variable can have two or more indices, like this: $x_i^{(j)}$ or like this $x_{i,j}^{(k)}$. For example, in neural networks, we denote as $x_{l,u}^{(j)}$ the input feature j of unit u in layer l.

A **matrix** is a rectangular array of numbers arranged in rows and columns. Below is an example of a matrix with two rows and two columns,

$$\begin{bmatrix} 2 & 4 & -3 \\ 21 & -6 & -1 \end{bmatrix}.$$

Matrices are denoted with bold capital letters, such as $\mathbf{A}$ or $\mathbf{W}$.

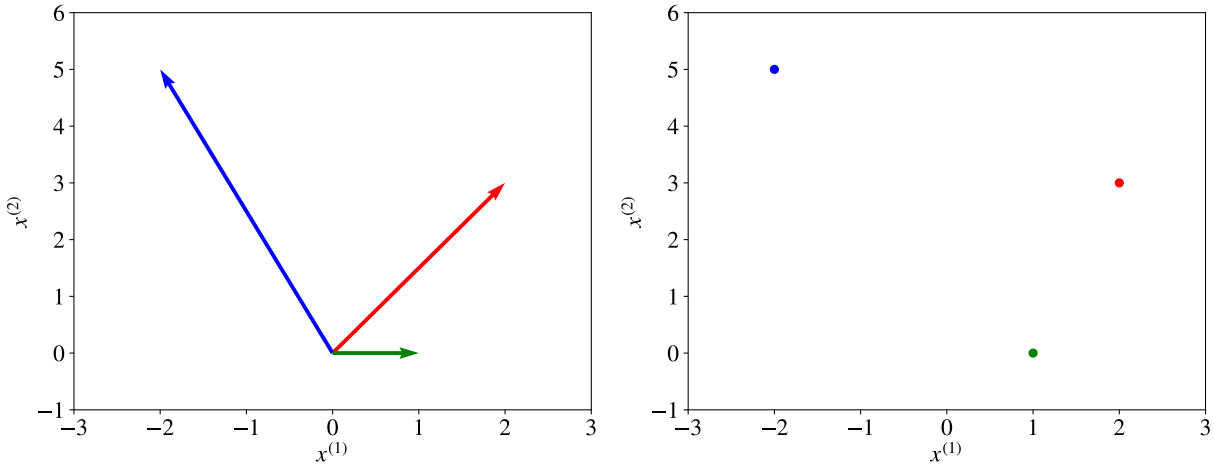


Figure 1: Three vectors visualized as directions and as points.

A *set* is an unordered collection of unique elements. We denote a set as a calligraphic capital character, for example, $\mathcal{S}$. A set of numbers can be finite (include a fixed amount of values). In this case, it is denoted using accolades, for example, $\{1, 3, 18, 23, 235\}$ or $\{x_1, x_2, x_3, x_4, \ldots, x_n\}$. A set can be infinite and include all values in some interval. If a set includes all values between a and b, including a and b, it is denoted using brackets as $[a, b]$. If the set doesn't include the values a and b, such a set is denoted using parentheses like this: (a, b). For example, the set $[0, 1]$ includes such values as 0, 0.0001, 0.25, 0.784, 0.9995, and 1.0. A special set denoted $\mathbb{R}$ includes all numbers from minus infinity to plus infinity.

When an element x belongs to a set $\mathcal{S}$, we write $x \in \mathcal{S}$. We can obtain a new set $\mathcal{S}_3$ as an *intersection* of two sets $\mathcal{S}_1$ and $\mathcal{S}_2$. In this case, we write $\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cap \mathcal{S}_2$. For example $\{1, 3, 5, 8\} \cap \{1, 8, 4\}$ gives the new set $\{1, 8\}$.

We can obtain a new set $\mathcal{S}_3$ as a *union* of two sets $\mathcal{S}_1$ and $\mathcal{S}_2$. In this case, we write $\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$. For example $\{1, 3, 5, 8\} \cup \{1, 8, 4\}$ gives the new set $\{1, 3, 4, 5, 8\}$.

2.1.2 Capital Sigma Notation

The summation over a collection $X = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$ or over the attributes of a vector $\mathbf{x} = [x^{(1)}, x^{(2)}, \ldots, x^{(m-1)}, x^{(m)}]$ is denoted like this:

$$\sum_{i=1}^{n} x_i \stackrel{\text{def}}{=} x_1 + x_2 + \ldots + x_{n-1} + x_n, \text{ or else: } \sum_{j=1}^{m} x^{(j)} \stackrel{\text{def}}{=} x^{(1)} + x^{(2)} + \ldots + x^{(m-1)} + x^{(m)}.$$

The notation $\stackrel{\text{def}}{=}$ means "is defined as".

2.1.3 Capital Pi Notation

A notation analogous to capital sigma is the *capital pi notation*. It denotes a product of elements in a collection or attributes of a vector:

$$\prod_{i=1}^{n} x_i \stackrel{\text{def}}{=} x_1 \cdot x_2 \cdot \ldots \cdot x_{n-1} \cdot x_n,$$

where $a \cdot b$ means a multiplied by b. Where possible, we omit $\cdot$ to simplify the notation, so ab also means a multiplied by b.

2.1.4 Operations on Sets

A derived set creation operator looks like this: $\mathcal{S}' \leftarrow \{x^2 \,|\, x \in \mathcal{S}, x > 3\}$. This notation means that we create a new set $\mathcal{S}'$ by putting into it x squared such that that x is in $\mathcal{S}$, and x is greater than 3.

The cardinality operator $|\mathcal{S}|$ returns the number of elements in set $\mathcal{S}$.

2.1.5 Operations on Vectors

The sum of two vectors $\mathbf{x} + \mathbf{z}$ is defined as the vector $[x^{(1)} + z^{(1)}, x^{(2)} + z^{(2)}, \ldots, x^{(m)} + z^{(m)}]$.

The difference of two vectors $\mathbf{x} - \mathbf{z}$ is defined as $[x^{(1)} - z^{(1)}, x^{(2)} - z^{(2)}, \ldots, x^{(m)} - z^{(m)}]$.

A vector multiplied by a scalar is a vector. For example $\mathbf{x}c \stackrel{\text{def}}{=} [cx^{(1)}, cx^{(2)}, \ldots, cx^{(m)}]$.

A *dot-product* of two vectors is a scalar. For example, $\mathbf{wx} \stackrel{\text{def}}{=} \sum_{i=1}^{m} w^{(i)} x^{(i)}$. In some books, the dot-product is denoted as $\mathbf{w} \cdot \mathbf{x}$. The two vectors must be of the same dimensionality. Otherwise, the dot-product is undefined.

The multiplication of a matrix $\mathbf{W}$ by a vector $\mathbf{x}$ results in another vector. Let our matrix be,

$$\mathbf{W} = \begin{bmatrix} w^{(1,1)} & w^{(1,2)} & w^{(1,3)} \\ w^{(2,1)} & w^{(2,2)} & w^{(2,3)} \end{bmatrix}.$$

When vectors participate in operations on matrices, a vector is by default represented as a matrix with one column. When the vector is on the right of the matrix, it remains a column vector. We can only multiply a matrix by vector if the vector has the same number of rows

as the number of columns in the matrix. Let our vector be $\mathbf{x} \stackrel{\text{def}}{=} [x^{(1)}, x^{(2)}, x^{(3)}]$. Then $\mathbf{Wx}$ is a two-dimensional vector defined as,

$$\begin{aligned}\mathbf{Wx} &= \begin{bmatrix} w^{(1,1)} & w^{(1,2)} & w^{(1,3)} \\ w^{(2,1)} & w^{(2,2)} & w^{(2,3)} \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{bmatrix} \\ &\stackrel{\text{def}}{=} \begin{bmatrix} w^{(1,1)}x^{(1)} + w^{(1,2)}x^{(2)} + w^{(1,3)}x^{(3)} \\ w^{(2,1)}x^{(1)} + w^{(2,2)}x^{(2)} + w^{(2,3)}x^{(3)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{w}^{(1)}\mathbf{x} \\ \mathbf{w}^{(2)}\mathbf{x} \end{bmatrix}\end{aligned}$$

If our matrix had, say, five rows, the result of the product would be a five-dimensional vector.

When the vector is on the left side of the matrix in the multiplication, then it has to be *transposed* before we multiply it by the matrix. The transpose of the vector $\mathbf{x}$ denoted as $\mathbf{x}^\top$ makes a row vector out of a column vector. Let's say,

$$\mathbf{x} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix}, \text{ then } \mathbf{x}^\top \stackrel{\text{def}}{=} \begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix}.$$

The multiplication of the vector $\mathbf{x}$ by the matrix $\mathbf{W}$ is given by $\mathbf{x}^\top\mathbf{W}$,

$$\begin{aligned}\mathbf{x}^\top\mathbf{W} &= \begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix} \begin{bmatrix} w^{(1,1)} & w^{(1,2)} & w^{(1,3)} \\ w^{(2,1)} & w^{(2,2)} & w^{(2,3)} \end{bmatrix} \\ &\stackrel{\text{def}}{=} \begin{bmatrix} w^{(1,1)}x^{(1)} + w^{(2,1)}x^{(2)}, w^{(1,2)}x^{(1)} + w^{(2,2)}x^{(2)}, w^{(1,3)}x^{(1)} + w^{(2,3)}x^{(2)} \end{bmatrix}\end{aligned}$$

As you can see, we can only multiply a vector by a matrix if the vector has the same number of dimensions as the number of rows in the matrix.

2.1.6 Functions

A function is a relation that associates each element x of a set $\mathcal{X}$, the *domain* of the function, to a single element y of another set $\mathcal{Y}$, the *codomain* of the function. A function usually has a name. If the function is called f, this relation is denoted $y = f(x)$ (read f of x), the element x is the argument or input of the function, and y is the value of the function or the output. The symbol that is used for representing the input is the variable of the function (we often say that f is a function of the variable x).

We say that $f(x)$ has a *local minimum* at $x = c$ if $f(x) \geq f(c)$ for every x in some open interval around $x = c$. An *interval* is a set of real numbers with the property that any number

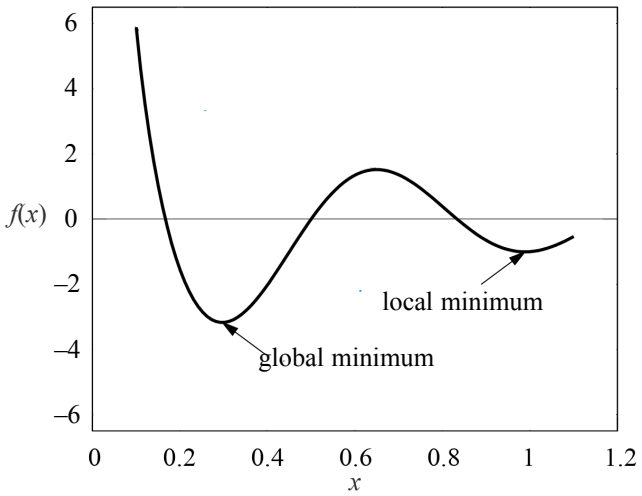


Figure 2: A local and a global minima of a function.

that lies between two numbers in the set is also included in the set. An *open interval* does not include its endpoints and is denoted using parentheses. For example, $(0, 1)$ means "all numbers greater than 0 and less than 1". The minimal value among all the local minima is called the *global minimum*. See illustration in Figure 2.

A vector function, denoted as $\mathbf{y} = \boldsymbol{f}(x)$ is a function that returns a vector $\mathbf{y}$. It can have a vector or a scalar argument.

2.1.7 Max and Arg Max

Given a set of values $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, the operator $\max_{a \in \mathcal{A}} f(a)$ returns the highest value $f(a)$ for all elements in the set $\mathcal{A}$. On the other hand, the operator $\arg\max_{a \in \mathcal{A}} f(a)$ returns the element of the set $\mathcal{A}$ that maximizes $f(a)$.

Sometimes, when the set is implicit or infinite, we can write $\max_a f(a)$ or $\arg\max_a f(a)$.

Operators min and $\arg\min$ operate in a similar manner.

2.1.8 Assignment Operator

The expression $a \leftarrow f(x)$ means that the variable a gets the new value: the result of $f(x)$. We say that the variable a gets assigned a new value. Similarly, $\mathbf{a} \leftarrow [a_1, a_2]$ means that the vector variable $\mathbf{a}$ gets the two-dimensional vector value $[a_1, a_2]$.

2.1.9 Derivative and Gradient

A *derivative* f' of a function f is a function or a value that describes how fast f grows (or decreases). If the derivative is a constant value, like 5 or -3, then the function grows (or decreases) constantly at any point x of its domain. If the derivative f' is a function, then the function f can grow at a different pace in different regions of its domain. If the derivative f' is positive at some point x, then the function f grows at this point. If the derivative of f is negative at some x, then the function decreases at this point. The derivative of zero at x means that the function's slope at x is horizontal.

The process of finding a derivative is called *differentiation*.

Derivatives for basic functions are known. For example if $f(x) = x^2$, then $f'(x) = 2x$; if $f(x) = 2x$ then $f'(x) = 2$; if $f(x) = 2$ then $f'(x) = 0$ (the derivative of any function $f(x) = c$, where c is a constant value, is zero).

If the function we want to differentiate is not basic, we can find its derivative using the *chain rule*. For instance if $F(x) = f(g(x))$, where f and g are some functions, then $F'(x) = f'(g(x))g'(x)$. For example if $F(x) = (5x + 1)^2$ then $g(x) = 5x + 1$ and $f(g(x)) = (g(x))^2$. By applying the chain rule, we find $F'(x) = 2(5x + 1)g'(x) = 2(5x + 1)5 = 50x + 10$.

Gradient is the generalization of derivative for functions that take several inputs (or one input in the form of a vector or some other complex structure). A gradient of a function is a vector of *partial derivatives*. You can look at finding a partial derivative of a function as the process of finding the derivative by focusing on one of the function's inputs and by considering all other inputs as constant values.

For example, if our function is defined as $f([x^{(1)}, x^{(2)}]) = ax^{(1)} + bx^{(2)} + c$, then the partial derivative of function f *with respect to* $x^{(1)}$, denoted as $\frac{\partial f}{\partial x^{(1)}}$, is given by,

$$\frac{\partial f}{\partial x^{(1)}} = a + 0 + 0 = a,$$

where a is the derivative of the function $ax^{(1)}$; the two zeroes are respectively derivatives of $bx^{(2)}$ and c, because $x^{(2)}$ is considered constant when we compute the derivative with respect to $x^{(1)}$, and the derivative of any constant is zero.

Similarly, the partial derivative of function f with respect to $x^{(2)}$, $\frac{\partial f}{\partial x^{(2)}}$, is given by,

$$\frac{\partial f}{\partial x^{(2)}} = 0 + b + 0 = b.$$

The gradient of function f, denoted as ∇f is given by the vector $[\frac{\partial f}{\partial x^{(1)}}, \frac{\partial f}{\partial x^{(2)}}]$.

The chain rule works with partial derivatives too, as I illustrate in Chapter 4.

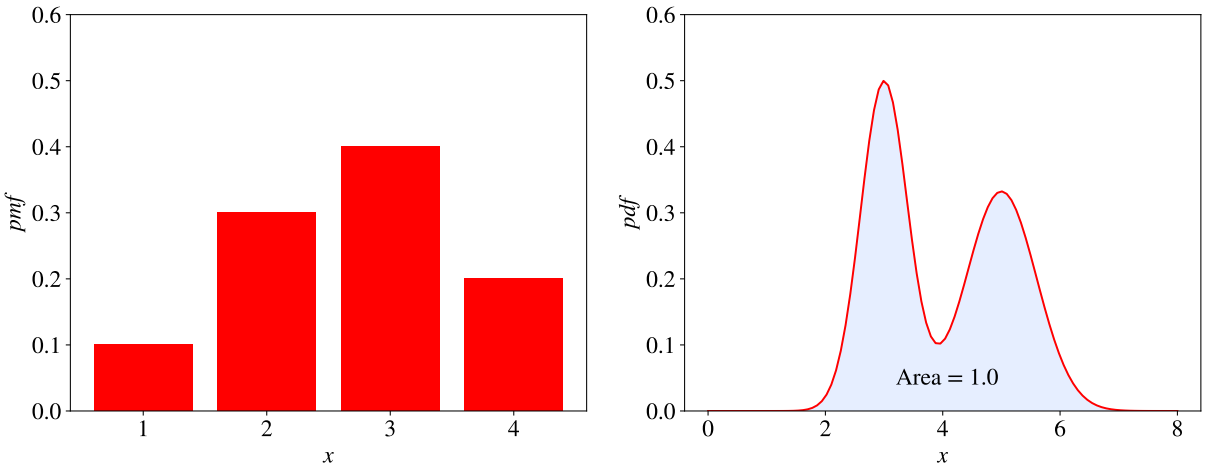


Figure 3: A probability mass function and a probability density function.

2.2 Random Variable

A *random variable*, usually written as an italic capital letter, like X, is a variable whose possible values are numerical outcomes of a random phenomenon. There are two types of random variables: *discrete* and *continuous*.

A *discrete random variable* takes on only a countable number of distinct values such as *red*, *yellow*, *blue* or 1, 2, 3, ...

The *probability distribution* of a discrete random variable is described by a list of probabilities associated with each of its possible values. This list of probabilities is called *probability mass function* (pmf). For example: $\Pr(X = red) = 0.3$, $\Pr(X = yellow) = 0.45$, $\Pr(X = blue) = 0.25$. Each probability in a probability mass function is a value greater than or equal to 0. The sum of probabilities equals 1 (fig. 3a).

A *continuous random variable* (CRV) takes an infinite number of possible values in some interval. Examples include height, weight, and time. Because the number of values of a continuous random variable X is infinite, the probability $\Pr(X = c)$ for any c is 0. Therefore, instead of the list of probabilities, the probability distribution of a CRV (a continuous probability distribution) is described by a *probability density function* (pdf). The pdf is a function whose codomain is nonnegative and the area under the curve is equal to 1 (fig. 3b).

Let a discrete random variable X have k possible values $\{x_i\}_{i=1}^{k}$. The *expectation* of X denoted as $\mathbb{E}[X]$ is given by,

$$\mathbb{E}[X] \stackrel{\text{def}}{=} \sum_{i=1}^{k} x_i \cdot \Pr(X = x_i) = x_1 \cdot \Pr(X = x_1) + x_2 \cdot \Pr(X = x_2) + \cdots + x_k \cdot \Pr(X = x_k), \quad (1)$$

where $\Pr(X = x_i)$ is the probability that X has the value x_i according to the pmf. The expectation of a random variable is also called the *mean*, *average* or *expected value* and is frequently denoted with the letter μ. The expectation is one of the most important *statistics* of a random variable.

Another important statistic is the *standard deviation*, defined as,

$$\sigma \stackrel{\text{def}}{=} \sqrt{\mathbb{E}[(X - \mu)^2]}.$$

Variance, denoted as σ^2 or $var(X)$, is defined as,

$$\sigma^2 = \mathbb{E}[(X - \mu)^2].$$

For a discrete random variable, the standard deviation is given by:

$$\sigma = \sqrt{\Pr(X = x_1)(x_1 - \mu)^2 + \Pr(X = x_2)(x_2 - \mu)^2 + \cdots + \Pr(X = x_k)(x_k - \mu)^2},$$

where $\mu = \mathbb{E}[X]$.

The expectation of a continuous random variable X is given by,

$$\mathbb{E}[X] \stackrel{\text{def}}{=} \int_{\mathbb{R}} x f_X(x)\, dx, \quad (2)$$

where f_X is the pdf of the variable X and $\int_{\mathbb{R}}$ is the *integral* of function xf_X.

Integral is an equivalent of the summation over all values of the function when the function has a continuous domain. It equals the area under the curve of the function. The property of the pdf that the area under its curve is 1 mathematically means that $\int_{\mathbb{R}} f_X(x)\, dx = 1$.

Most of the time we don't know f_X, but we can observe some values of X. In machine learning, we call these values **examples**, and the collection of these examples is called a **sample** or a **dataset**.

2.3 Unbiased Estimators

Because f_X is usually unknown, but we have a sample $S_X = \{x_i\}_{i=1}^N$, we often content ourselves not with the true values of statistics of the probability distribution, such as expectation, but with their *unbiased estimators*.

We say that $\hat{\theta}(S_X)$ is an unbiased estimator of some statistic θ calculated using a sample S_X drawn from an unknown probability distribution if $\hat{\theta}(S_X)$ has the following property:

$$\mathbb{E}\left[\hat{\theta}(S_X)\right] = \theta,$$

where $\hat{\theta}$ is a *sample statistic*, obtained using a sample S_X and not the real statistic θ that can be obtained only knowing X; the expectation is taken over all possible samples drawn from X. Intuitively, this means that if you can have an unlimited number of such samples as S_X, and you compute some unbiased estimator, such as $\hat{\mu}$, using each sample, then the average of all these $\hat{\mu}$ equals the real statistic μ that you would get computed on X.

It can be shown that an unbiased estimator of an unknown $\mathbb{E}[X]$ (given by either eq. 1 or eq. 2) is given by $\frac{1}{N}\sum_{i=1}^N x_i$ (called in statistics the *sample mean*).

2.4 Bayes' Rule

The conditional probability $\Pr(X = x|Y = y)$ is the probability of the random variable X to have a specific value x given that another random variable Y has a specific value of y. The **Bayes' Rule** (also known as the **Bayes' Theorem**) stipulates that:

$$\Pr(X = x|Y = y) = \frac{\Pr(Y = y|X = x)\Pr(X = x)}{\Pr(Y = y)}.$$

2.5 Parameter Estimation

Bayes' Rule comes in handy when we have a model of X's distribution, and this model f_θ is a function that has some parameters in the form of a vector θ. An example of such a function could be the Gaussian function that has two parameters, μ and σ, and is defined as:

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

where $\theta \stackrel{\text{def}}{=} [\mu, \sigma]$.

This function has all the properties of a pdf. Therefore, we can use it as a model of an unknown distribution of X. We can update the values of parameters in the vector θ from the data using the Bayes' Rule:

$$\Pr(\theta = \hat{\theta}|X = x) \leftarrow \frac{\Pr(X = x|\theta = \hat{\theta})\Pr(\theta = \hat{\theta})}{\Pr(X = x)} = \frac{\Pr(X = x|\theta = \hat{\theta})\Pr(\theta = \hat{\theta})}{\sum_{\tilde{\theta}} \Pr(X = x|\theta = \tilde{\theta})}. \quad (3)$$

where $\Pr(X = x|\theta = \hat{\theta}) \stackrel{\text{def}}{=} f_{\hat{\theta}}$.

If we have a sample $\mathcal{S}$ of X and the set of possible values for θ is finite, we can easily estimate $\Pr(\theta = \hat{\theta})$ by applying Bayes' Rule iteratively, one example $x \in \mathcal{S}$ at a time. The initial value $\Pr(\theta = \hat{\theta})$ can be guessed such that $\sum_{\hat{\theta}} \Pr(\theta = \hat{\theta}) = 1$. This guess of the probabilities for different $\hat{\theta}$ is called the *prior*.

First, we compute $\Pr(\theta = \hat{\theta}|X = x_1)$ for all possible values $\hat{\theta}$. Then, before updating $\Pr(\theta = \hat{\theta}|X = x)$ once again, this time for $x = x_2 \in \mathcal{S}$ using eq. 3, we replace the prior $\Pr(\theta = \hat{\theta})$ in eq. 3 by the new estimate $\Pr(\theta = \hat{\theta}) \leftarrow \frac{1}{N} \sum_{x \in \mathcal{S}} \Pr(\theta = \hat{\theta}|X = x)$.

The best value of the parameters θ^* given one example is obtained using the principle of **maximum likelihood**:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{N} \Pr(\theta = \hat{\theta}|X = x_i). \quad (4)$$

If the set of possible values for θ isn't finite, then we need to optimize eq. 4 directly using a numerical optimization routine, such as gradient descent, which we consider in Chapter 4. Usually, we optimize the natural logarithm of the right-hand side expression in eq. 4 because the logarithm of a product becomes the sum of logarithms and it's easier for the machine to work with the sum than with a product[1].

2.6 Parameters vs. Hyperparameters

A hyperparameter is a property of a learning algorithm, usually (but not always) having a numerical value. That value influences the way the algorithm works. Hyperparameters aren't learned by the algorithm itself from data. They have to be set by the data analyst before running the algorithm. I show how to do that in Chapter 5.

Parameters are variables that define the model learned by the learning algorithm. Parameters are directly modified by the learning algorithm based on the training data. The goal of learning is to find such values of parameters that make the model optimal in a certain sense.

[1]Multiplication of many numbers can give either a very small result or a very large one. It often results in the problem of numerical overflow when the machine cannot store such extreme numbers in memory.

2.7 Classification vs. Regression

Classification is a problem of automatically assigning a **label** to an **unlabeled example**. Spam detection is a famous example of classification.

In machine learning, the classification problem is solved by a **classification learning algorithm** that takes a collection of **labeled examples** as inputs and produces a **model** that can take an unlabeled example as input and either directly output a label or output a number that can be used by the analyst to deduce the label. An example of such a number is a probability.

In a classification problem, a label is a member of a finite set of **classes**. If the size of the set of classes is two ("sick"/"healthy", "spam"/"not_spam"), we talk about **binary classification** (also called **binomial** in some sources). **Multiclass classification** (also called **multinomial**) is a classification problem with three or more classes[2].

While some learning algorithms naturally allow for more than two classes, others are by nature binary classification algorithms. There are strategies allowing to turn a binary classification learning algorithm into a multiclass one. I talk about one of them in Chapter 7.

Regression is a problem of predicting a real-valued label (often called a *target*) given an unlabeled example. Estimating house price valuation based on house features, such as area, the number of bedrooms, location and so on is a famous example of regression.

The regression problem is solved by a **regression learning algorithm** that takes a collection of labeled examples as inputs and produces a model that can take an unlabeled example as input and output a target.

2.8 Model-Based vs. Instance-Based Learning

Most supervised learning algorithms are model-based. We have already seen one such algorithm: SVM. Model-based learning algorithms use the training data to create a **model** that has **parameters** learned from the training data. In SVM, the two parameters we saw were $\mathbf{w}^*$ and b^*. After the model was built, the training data can be discarded.

Instance-based learning algorithms use the whole dataset as the model. One instance-based algorithm frequently used in practice is **k-Nearest Neighbors** (kNN). In classification, to predict a label for an input example the kNN algorithm looks at the close neighborhood of the input example in the space of feature vectors and outputs the label that it saw the most often in this close neighborhood.

[2]There's still one label per example though.

2.9 Shallow vs. Deep Learning

A shallow learning algorithm learns the parameters of the model directly from the features of the training examples. Most supervised learning algorithms are shallow. The notorious exceptions are **neural network** learning algorithms, specifically those that build neural networks with more than one **layer** between input and output. Such neural networks are called **deep neural networks**. In deep neural network learning (or, simply, deep learning), contrary to shallow learning, most model parameters are learned not directly from the features of the training examples, but from the outputs of the preceding layers.

Don't worry if you don't understand what that means right now. We look at neural networks more closely in Chapter 6.