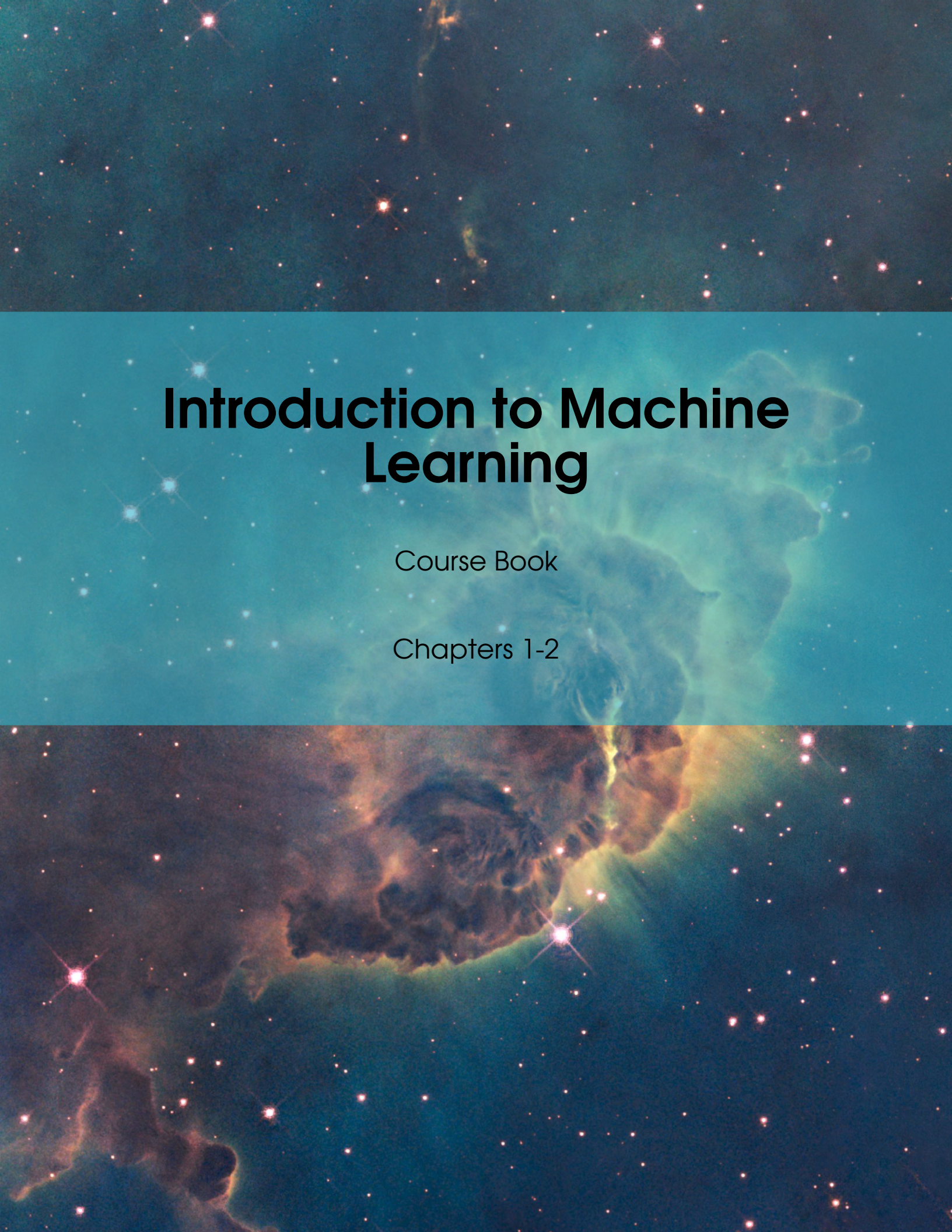


Introduction to Machine Learning

Course Book

Chapters 1-2

Introduction to Machine Learning

Matan Gavish and Gilad Green

Contents

0.1 Preface

0.1.1 Notation

Field/Context	Symbol	Meaning
General Mathematics	$\mathbb{R}$	The set of real numbers
	$\mathbb{R}^d$	The set of d-dimensional vectors over the field of real numbers
	$\mathbb{R}_+$	The set of non-negative real numbers
	$\mathbb{N}$	The set of natural numbers
	$\mathbb{R}^{m\times d}$	The set of real matrices with m rows and d columns
	$\{0,\ldots,m\}$	The set of natural numbers between 0 and m
	$[m]$	The set of natural numbers between 1 and m
	$\lvert S\rvert$	The number of elements in the set S
	$a \coloneqq b$	Defining a as the experssion b
	$a \propto b$	a is of size proportional to the size of b
	$\mathbf{x},\mathbf{y},\mathbf{w}$	Column vectors
	x_i, y_i, w_i	The i-th coordinate of a vector
	$\langle \mathbf{v},\mathbf{u}\rangle$ or $\mathbf{v}^\top\mathbf{u}$	Standard inner product between $\mathbf{v}$ and $\mathbf{u}$
	$\mathbf{v}\otimes\mathbf{u}$ or $\mathbf{v}\mathbf{u}^\top$	The outer product of $\mathbf{v}$ and $\mathbf{u}$
	$\lvert\lvert\cdot\rvert\rvert_1$	$=\sum_{i=1}^d \lvert x_i\rvert$ (the ℓ_1 norm of $\mathbf{x}$)
	$\lvert\lvert\cdot\rvert\rvert_2$ or $\lvert\lvert\cdot\rvert\rvert$	$=\sqrt{\sum_{i=1}^d x_i^2}$ (the ℓ_2 norm of $\mathbf{x}$)
	$\lvert\lvert\cdot\rvert\rvert_\infty$	$= max_i \lvert x_i\rvert$ (the ℓ_∞ norm of $\mathbf{x}$)
	$A_{i,j}$	The jth element of the ith row of A
	$A^\top$	The transpose of A: $A^\top_{i,j} = A_{j,i}$
	$det\,(A)$ or $\lvert A\rvert$	The determinant of the matrix A
	$V^\perp$	The set of vectors that are perpendicular to the vector space V
	$\partial f/\partial \mathbf{x}_i$	The derivative of the function f w.r.t the ith coordinate of $\mathbf{x}$
	∇f	The gradient of the function f
	$\mathbf{J}_\mathbf{x}(f)$	The Jacobian of the function f w.r.t $\mathbf{x}$
	$\overset{!}{=}$	Expressing that we would like to equate the left side to the right side. Usually comes in the context of equating the derivative to zero and solving for the parameter in question.
	Ω	A sample space
	$\mathbb{P}$	The probabiltiy of an event or random variable
	$\mathbb{P}_\mathcal{D}$	The probability of an event or random variable w.r.t the distribution $\mathcal{D}$
	$\mathbb{E}$	The expectation of a random variable
	$x \sim \mathcal{D}$	Sampling x according to $\mathcal{D}$

Field/Context	**Symbol**	**Meaning**
Machine	m	Number of samples
Learning	d, k	Dimension of sample vector; Number of features
	$\mathcal{X}$	Domain set
	$\mathcal{Y}$	Response set
	$\mathcal{Z} = (\mathcal{X} \times \mathcal{Y})$	The product space of domain set and response set
	$S = x_1, \ldots, x_n$	A sequence of samples from domain set
	$S = z_1, \ldots, z_n$	A sequence of samples and responses from product space $\mathcal{Z}$
	$x_1, \ldots, x_m \overset{iid}{\sim} \mathcal{D}$	Sampling m values independently and all according to $\mathcal{D}$
	θ	Scalar or vector representing some parameter of a distribution
	Θ	A set of parameters
	$\widehat{\theta}$	An estimator of θ
	$\mathcal{H}$	Set of functions, named hypothesis class
	$\mathcal{A}$	Learning algorithm
	$\mathcal{O}$	Big-O asymptotic notation

1. Mathematical Basis

1.1 Linear Algebra

1.1.1 Linear Transformations

Definition 1.1.1 — Linear Transformation. Let $V \in \mathbb{R}^d$ and $W \in \mathbb{R}^m$ be two vectors spaces. A function $T : V \to W$ is called a linear transformation of V into W, if $\forall u, v \in V$ and $c \in \mathbb{R}$.

- Additivity: $T(u+v) = T(u) + T(v)$
- Scalar multiplication: $T(cu) = cT(u)$

For V and W of a finite dimension, any linear transformation can be represented by a matrix A. Therefore, from now and on we will focus only on finite-dimensional spaces, and implicitly refer to the matrix representing the linear transformation.

Definition 1.1.2 — Affine Transformation. An *affine transformation* is a transformation of the form $T(u) = Au + w$, where $u \in V, w \in W$.

Notice, that by definition an affine transformation is not a linear transformation. Notice that for a linear transformation A it holds that $A \cdot 0_V = 0_W$, but in the case of an affine transformation where $0 \neq w \in W$ then $T(0_V) = A \cdot 0_V + w \neq 0_w$.

Let us define some vector spaces associated with each linear transformation

Definition 1.1.3 — Fundamental Subspaces. Let A be the matrix corresponding the linear transformation $T : V \to W$. We define the *four fundamental subspaces*:

- Kernel- (or null-) space of A as $Ker(A) := \{x \in V | Ax = 0\}$. Also denotes as $\mathcal{N}(A)$.

- Image- (or column/range-) space of A as $Im(A) \coloneqq \{w \in W | w = Ax, x \in V\}$. Also denotes as $Col(A)$ or $\mathcal{R}(A)$.
- Row space of A as $Im\left(A^\top\right) \coloneqq \left\{x \in V | x = A^\top w, w \in W\right\}$. Equivelantly it can be defined as the column space of $A^\top$ and therefore denoted as $Col\left(A^\top\right)$.
- Null space of $A^\top$ as $Ker\left(A^\top\right) \coloneqq \left\{x \in W | A^\top x = 0\right\}$. This space is also referred to as the left null space of A.

Note that by definition, $Ker(A), Row(A) \subseteq V$ and $Im(A) \subseteq W$. Using the above definitions let us gain some insights into what these vector spaces provide us with.

Definition 1.1.4 Let $A \in \mathbb{R}^{m \times d}$. The rank of A is the maximum number of linearly independent rows of A and denoted by $rank(A)$.

It holds that the rank of A equals both the dimension of the columns space and of the row space of A. As such, we refer to A being of *full rank* if and only if $rank(A) = min(m, d)$. Otherwise we say that A is rank deficient.

Definition 1.1.5 Let $A \in \mathbb{R}^{d \times d}$ be a square matrix. A is called invertible (or non-singular) if there exists a matrix $B \in \mathbb{R}^{d \times d}$ such that $AB = I_d = BA$. We denote the inverse by A^{-1}.

Claim 1.1.1 Let A be a square matrix. The following are equivalent (TFAE):
- A is invertible (non-singular)
- A is full-rank
- $Det(A) \neq 0$
- $Im(A) = \mathbb{R}^m$ (i.e., the image is the whole space)
- $ker(A) = \vec{0}$

■ **Example 1.1** Consider the following scenario: Suppose we are given a set of d linearly independent linear equations, each of the form $y_i = \sum_{j=1}^{d} \mathbf{w}_j \cdot x_{ij}$, where the $x_{i,j}$'s and y_i are given while $\mathbf{w}_j$'s are unknown. We would like to find a solution for this system of equations. That is, a coefficients vector $\mathbf{w} \in \mathbb{R}^d$ that satisfies:

$$\forall i \in [d] \; y_i = \sum_{j=1}^{d} \mathbf{w}_j \cdot x_{ij} = \mathbf{w}^\top x_i$$

Let us rearrange the equations in matrix form. Given a linear equation we will denote all it's x's by the vector $x_i \in \mathbb{R}^d$ where i denotes the numbering of the current equation. Similarly we will arrange all the y's in a vector $y \in \mathbb{R}^d$. Thus, we can represent the problem written above as follows:

$$\text{Find } \mathbf{w} \in \mathbb{R}^d \text{ such that } y = X\mathbf{w}$$

As we assumed that all linear equations are independent, the rows of X are linearly independent. Therefore, it is of full rank and there exists an invertible matrix X^{-1} such that $XX^{-1} = I$. Equipped

with this observation finding $\mathbf{w}$ is simply:

$$y = X\mathbf{w} \Rightarrow X^{-1}y = X^{-1}X\mathbf{w} \Rightarrow, \mathbf{w} = X^{-1}y$$

■

R Let us think of each vector $x_i \in \mathbb{R}^d$ as some independent observation (or sample) we have of some phenomena. Each coordinate of x_i corresponds some measurement we have of this observation. Together with this sample we are given some response value $y_i \in \mathbb{R}$. By solving for $\mathbf{w}$ we learn the relation between the x's and y's. Now suppose we are given a new sample $x \in \mathbb{R}^d$. As we already know the relation between the xs and the ys, we can predict what is the appropriate y value it achieves.
The general problem of finding such vectors is called **Regression**. In the case where the relationship is linear it is called **Linear Regression**. We will discuss linear regression in chapter 2.

1.1.2 Norms, Inner Products and Projections

In many applications in machine learning we are interested in measuring distances between vectors or sizes of vectors, and "using" a vector (or set of vectors) on another vector. For such, let us formulate these notions.

Definition 1.1.6 — Metric. A function on a set $X \subseteq \mathbb{F}^k$ $d : X \times X \to \mathbb{R}_+$ is called a metric function (or distance function) iff for any $v,u,w \in X$ it holds that:

- $d(v,u) = 0 \iff v = u$
- Symmetry: $d(v,u) = d(u,v)$
- Triangle inequality $d(v,u) \leq d(v,w) + d(w,u)$.

These conditions also imply that a metric is non-negative. As such, we also call a metric function a positive-definite function. Some common metric functions are the absolute distance or the Euclidean distance.

Exercise 1.1 Let $v,u \in \mathbb{R}^k$. Show that the absolute distance, defined as the sum of absolute element-wise subtraction between the vectors $d(v,u) := \sum |v_i - u_i|$, is a metric function. ■

Proof. Firstly, notice that for some scalars $a,b \in \mathbb{R}$ it holds that $|a-b| = 0$ iff $a = b$. Therefore d, being a sum of non-negative elements equals zero iff all elements are zero. This takes place iff $v = u$. Next, symmetry of d is achieved through symmetry of the absolute value function. Lastly, let $v,u,w \in \mathbb{R}^k$ then

$$d(v,u) = \sum |v_i - u_i| = \sum |v_i - w_i + w_i - u_i| \leq \sum |v_i - w_i| + \sum |w_i - u_i| = d(v,w) + d(w,u)$$

■

Next, let us define the notion of a *size* of a vector.

Definition 1.1.7 — Norm. A norm is a function $||\cdot|| : \mathbb{R}^d \to \mathbb{R}_+$ that satisfies the following three conditions for all $a \in \mathbb{R}$ and all $u,v \in \mathbb{R}^d$:

- Positive definite: $||v|| \geq 0$ and $||v|| = 0$ iff v is the zero vector.
- Positive homogeneity: $||av|| = |a| \cdot ||v||$.

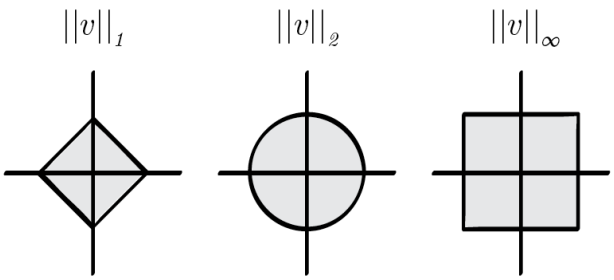
- Triangle inequality: $||v+u|| \leq ||v|| + ||u||$.

We can think of this size in the sense of vector's *distance* from the origin, under some distance function defined by the norm. A few commonly used norms are:

- Absolute norm (ℓ_1): $||v||_1 \coloneqq \sum |v_i|$.
- Euclidean norm (ℓ_2): $||v||_2 \coloneqq \sqrt{\sum x_i^2}$.
- Infinity norm: $||x||_\infty \coloneqq max_i |v_i|$.

R The absolute and Euclidean norms are part of a wider family of norms called the L_p norms defined as $||v||_p \coloneqq \left(\sum |v_i^p|\right)^{1/p}, \quad p \in \mathbb{N}$.

Definition 1.1.8 Let V be a vector space and $||\cdot||$ be a norm over this space. The unit ball of $||\cdot||$ is defined as the set of vectors such that: $B_{||\cdot||} = \{v \in V : ||v|| \leq 1\}$.



Now that we have defined the notions of distances and sizes of vectors, we want to define what it means to "apply" some vector on another.

Definition 1.1.9 — Inner Product. An inner product space is a vector space V over $\mathbb{R}$ together with a map $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ satisfying that $\forall v, u, w \in V, \alpha \in \mathbb{R}$:

- Symmetry: $\langle v, u \rangle = \langle u, v \rangle$
- Linearity: $\langle \alpha v + w, u \rangle = \alpha \langle v, u \rangle + \langle w, u \rangle$
- Non-negativity : $\langle v, v \rangle \geq 0$ and $\langle v, v \rangle = 0 \iff v = 0$

Notice the similarity between the definition of a norm and of an inner product. In fact, given an inner-product space, we are also given a norm on this space.

Claim 1.1.2 — Induced Norm. Let H be an inner product space. Then the function $||\cdot|| : H \to \mathbb{R}_+$ is defined $\forall v \in H$ by $||v|| = \langle v, v \rangle^{\frac{1}{2}}$ is a norm on H.

Exercise 1.2 Let $v, u \in V$. Show that $\langle v, u \rangle = ||v||\,||u|| \cos\theta$, where θ is the angle between v, u. ■

Proof. Recall the Law of Cosines: in a triangle with lengths a, b, c, then

$$c^2 = a^2 + b^2 - 2ab\cos\theta$$

By applying the cosine law to the triangle defined by v and u and $v-u$ we see that:

$$||v-u||^2 = ||v||^2 + ||u||^2 - 2\,||v||\cdot||u||\cdot\cos\theta$$

On the other hand we also know that:

$$||v-u||^2 = \langle v-u, v-u\rangle = \langle v,v\rangle - 2\langle v,u\rangle + \langle u,u\rangle = ||v||^2 + ||u||^2 - 2\langle v,u\rangle$$

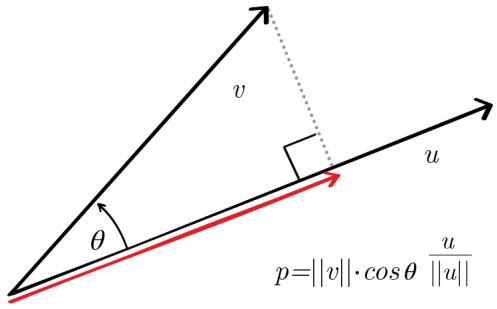
Hence, we conclude that:

$$||v||\cdot||u||\cdot\cos\theta = \langle v,u\rangle$$

■

From the above, we have an expression for the angle between two vectors, using the inner-product. We can therefore define what it means to project one vector onto the other. Using the identity of $\cos\theta$:

$$p = ||v||\cos\theta\cdot\frac{u}{||u||} = ||v||\,\frac{\langle v,u\rangle}{||v||\cdot||u||}\cdot\frac{u}{||u||} = \frac{\langle v,u\rangle}{||u||^2}\cdot u$$



Definition 1.1.10 — Vector Projection. A projection of a vector v onto a vector u, is a vector p of length $||v||\cos\theta$ in the direction of u.

Notice, that for the special case where $\theta = 90^\circ$ we get $\langle v,u\rangle = 0$. In this case we say that the vectors v,u are "orthogonal", and use the notation: $v\perp u$. If v,u are also unit vectors we say that the vectors v,u are "orthonormal" to each other.

Definition 1.1.11 Let $(V,||\cdot||)$ be a normed space. We say that $v\in V$ is a unit vector iff $||v|| = 1$.

Definition 1.1.12 An orthogonal matrix is a square matrix whose columns are unit vectors orthogonal to one another (i.e. they are orthonormal vectors) and whose rows are unit vectors orthogonal to one another.

Lemma 1.1.3 Let $A \in \mathbb{R}^{d\times d}$ orthogonal matrix, then

$$AA^\top = I = A^\top A$$

Putting together the definitions of a vector projection and orthogonal matrices we can define the notion of orthogonal projecting a vector onto some linear subspace.

Definition 1.1.13 — Outer Product. Let $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$. The outer product of u and v, denoted by $u \otimes v$ or $uv^\top$, is defined as an $m \times n$ matrix as follows:

$$(u\otimes v)_{ij} = u_i \cdot v_j, \quad u\otimes v = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdots & u_1v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_mv_1 & u_mv_2 & \cdots & u_mv_n \end{bmatrix}$$

Definition 1.1.14 Let V be a k-dimensional subspace of $\mathbb{R}^d$, and let $v_1, \ldots, v_k$ be an orthonormal basis of V. Define $P = \sum_{i=1}^k v_i \otimes v_i^\top = \sum_{i=1}^k v_i v_i^\top$. The matrix P is an ***orthogonal projection matrix*** onto the subspace V.

The following lemma summarizes some useful properties of orthogonal projection matrices.

Lemma 1.1.4 Let $v_1, \ldots, v_k$ be a set of orthonormal vectors, and let $P = \sum_{i=1}^k v_i \otimes v_i^\top = \sum_{i=1}^k v_i v_i^\top$. P has the following properties:

- P is symmetric
- $P^2 = P$
- The eigenvalues of P are either 0 or 1. $v_1, \ldots, v_k$ are the eigenvectors of P which correspond to the eigenvalue 1.
- $(I - P)P = 0$
- $\forall x \in \mathbb{R}^d$ and $\forall u \in V$, $||x - u|| \geq ||x - Px||$
- $x \in V \Rightarrow Px = x$

R The outer product of two non-zero vectors defines a matrix with rank of 1. Notice, that the orthogonal projection matrix, being a sum of such matrices, is in fact a sum of rank 1 matrices.

1.1.3 Matrix Decompositions

Matrix factorizations/decompositions are a strong tool with many theoretical as well as practical usages. It often appears in many different machine learning approaches, some of which we will encounter.

Definition 1.1.15 Let A be a square matrix. A is diagonalizable if there exists an invertible matrix P such that $P^{-1}AP$ is diagonal.

Next, we would like to see if we could represent A as the multiplication of orthogonal matrices, and a diagonal one.

Definition 1.1.16 — Eigenvector and Eigenvalue. Let A a square matrix. We say that a vector $0 \neq v \in V$ is an eigenvector of A corresponding to an eigenvalue $\lambda \in \mathbb{R}$ if $Av = \lambda v$.

Recall that any square *normal* matrix has an orthonormal basis of eigenvectors. In particular this holds for symmetric matrices:

Claim 1.1.5 Let A be a square symmetric matrix. Then there exists an orthonormal basis $u_1, ..., u_n \in \mathbb{R}^d$ of eigenvectors of A.

A diagonalizable matrix can be written as a product involving its eigenvectors and eigenvalues. This is sometimes known as the Eigenvalue Decomposition (EVD). For symmetric matrices we have:

Theorem 1.1.6 — Matrix Diagonalization - EVD. Let $A \in \mathbb{R}^{d \times d}$ be a real symmetric matrix. Then there exist an orthogonal matrix $U \in \mathbb{R}^{d \times d}$ and a diagonal matrix D such that $A = UDU^\top$ and $D_{i,i}$ are the eigenvalues of A ($i = 1..n$).

This decomposition is very useful. For example, notice that it is very easy to compute high powers of A: $A^k = UDU^\top \cdot UDU^\top \cdot UDU^\top = UD^kU^\top$. It is also easy to compute the inverse of A, if it exists: $A^{-1} = UD^{-1}U^\top$.
A serious drawback of the EVD is not every matrix can be decomposed this way. We now derive another decomposition, one which exists for any matrix - for non-symmetric and even non-square matrices.

Definition 1.1.17 Let $A \in \mathbb{R}^{m \times d}$ and let $v \in \mathbb{R}^d$, $u \in \mathbb{R}^m$ be unit vectors. We say that v, u are right- and left singular vectors of A, respectively, corresponding to a singular value $\sigma \in \mathbb{R}_+$ if $Av = \sigma u$.

Theorem 1.1.7 — Singular Value Decoposition (SVD). Let $A \in \mathbb{R}^{m \times d}$ be a real matrix. A can be written as a singular value decomposition of the form $A = U\Sigma V^\top$, where:

- The matrix $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix whose columns are the left singular values of A.
- The matrix $\Sigma \in \mathbb{R}^{m \times d}$ is a matrix where $\forall i \neq j \; \Sigma_{ij} = 0$ and $\Sigma_{ii} \equiv \sigma_i$ are non-negative and satisfy $\sigma_1 \geq \ldots \geq \sigma_d \geq 0$. The diagonal elements $sigma_i$ are the singular values of A.
- The matrix $V \in \mathbb{R}^{d \times d}$ is an orthogonal matrix whose columns are the right singular values of A.

As this decomposition exists for any real matrix it makes it very useful in many learning applications. It also has many useful properties such as:

- It provides a representation of the range (columns space) and kernel (null space) of the matrix A. The right-singular vector corresponding to vanishing singular values (i.e with value of zero) span $Ker(A)$.
- From the above we realize that the rank of A equals to the number of non-zero singular values. Looking at the singular values themselves (also referred to as spectrum) we can also derive the *effective rank* of the matrix. As performing algebraic operations are sensitive to floating-point errors, positive but very small singular values might cause a rank deficient matrix to appear of full rank. Zeroing singular values beneath a certain threshold are often assumed to be zero. Then the number of singular values over such threshold represents the effective rank of the

matrix.

- As we can derive the rank, if all singular values are strictly larger than zero ($\sigma_d > 0$) then $dim\,(Ker\,(A)) = 0$ and the matrix is invertible.
- Based on the SVD we can define the **Moore–Penrose pseudoinverse** of a matrix $A^\dagger = V\Sigma^\dagger U^\top$ where $\Sigma^\dagger$ is obtained from Σ by replacing every non-zero singular value with its multiplicative inverse. This definition will come in handy when solving the linear regression problem (2.1.2.5).

Beyond the properties above, notice that we can also represent the SVD in a more compact manner. Suppose that $rank\,(A) = r$. This means that the number of non-zero singular values is r, and notice that $r \leq \min\{d,m\}$. When $m \leq d$ then A and Σ are both wide matrices (they have more columns than rows):

$$A = U\Sigma V^\top = \begin{bmatrix} | & & | & & | \\ u_1 & \cdots & u_r & \cdots & u_m \\ | & & | & & | \end{bmatrix} \left[\begin{array}{ccc|ccc} \sigma_1 & \cdots & 0 & & & \\ \vdots & \ddots & \vdots & & 0 & \\ 0 & \cdots & \sigma_r & & & \\ \hline & & & 0 & \cdots & 0 \\ & 0 & & \vdots & \ddots & \vdots \\ & & & 0 & \cdots & 0 \end{array}\right] \begin{bmatrix} - & v_1^\top & - \\ & \vdots & \\ - & v_r^\top & - \\ & \vdots & \\ - & v_d^\top & - \end{bmatrix}$$

Since $\sigma_{r+1}, \ldots \sigma_m$ are all zero, and any off diagonal element of Σ is zero, the left- and right singular values with indices greater than r are multiplied by zeros and do not take part in the final construction of the matrix A. Their purpose is in expanding the set of left- and right singular vectors to form a basis for $\mathbb{R}^m$ and $\mathbb{R}^d$ respectively. This means that the important information carried by the SVD about the matrix A is actually contained in a smaller $r \times r$ matrix, sometimes called the **compact SVD of** A, which we can write as:

$$A = \tilde{U}\tilde{\Sigma}\tilde{V}^\top = \overbrace{\begin{bmatrix} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{bmatrix}}^{m\times r} \overbrace{\begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix}}^{r\times r} \overbrace{\begin{bmatrix} - & v_1^\top & - \\ & \vdots & \\ - & v_r^\top & - \end{bmatrix}}^{r\times d} \tag{1.1}$$

To avoid cluttered notations we will drop the $\tilde{\cdot}$ notation and refer to U, Σ, V in the compact form.

The two decompositions seen above and connected to one another. The following lemma the SVD of A to the EVD of $AA^\top$ and $A^\top A$. In particular, it shows that the SVD of A can be calculated in polynomial time in m and d.

Lemma 1.1.8 Let $A = U\Sigma V^\top$ be an SVD of $A \in \mathbb{R}^{m\times d}$. Then $AA^\top = U\Sigma\Sigma^\top U^\top$ is an EVD of $AA^\top$, and $A^\top A = V\Sigma^\top \Sigma V^\top$ is an EVD of $A^\top A$.

Lemma 1.1.8 therefore states that:

1. The left singular values of A (columns of U) are the eigenvectors of $AA^\top$.
2. The right singular values of A (columns of V) are the eigenvectors of $A^\top A$.
3. The singular values of A are just square root of the (shared) eigenvalues of $AA^\top$ and $A^\top A$. The eigenvalues $\sigma_1^2, \ldots, \sigma_d^2$ are the shared eigenvalues of $A^\top A$ and $AA^\top$.

R Note however, that the inverse claim is not correct. Take, for example, $A = U_1 \Sigma V^\top$ with $U_1 \equiv -U$. Both relations, $AA^\top = U\Sigma\Sigma^\top U^\top$ and $A^\top A = V\Sigma^\top \Sigma V^\top$ are still EVD's but $A \neq U\Sigma V^\top$.

1.2 Multivariate Calculus

Often when considering different machine learning techniques we will consider high dimensional functions $f : \mathbb{R}^d \to \mathbb{R}$. Usually, these functions will represent some distance measurement between our estimated prediction and the true values. As such, we would like to solve the optimization problem of minimizing this distance. Namely:

$$\underset{\mathbf{w}\in\mathbb{R}^d}{argmin}\, f(\mathbf{w})$$

1.2.1 Derivatives, Gradients and Jacobians

1.2.1.1 Derivatives

A common approach for finding such a minimizer is by computing the derivative of the function, equating to zero and solving for our parameters $\mathbf{w}$. When discussing multivariate functions we use gradients rather than scalar derivatives.

Definition 1.2.1 — Derivative. Let $f : \mathbb{R} \to \mathbb{R}$. The derivative of f at point $x \in \mathbb{R}$ is defined as

$$\frac{d}{dx} f(x) := \lim_{a\to 0} \frac{f(x+a) - f(x)}{a}$$

■ **Example 1.2 — ReLU.** Consider the **R**ectified **L**inear **U**nit function defined as the positive part of its argument: $f(x) = x^+ = \max(0,x)$. This function is in common use in the context of artificial neural networks. The derivative if this function is:

$$\frac{df(x)}{dx} = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases}$$

■

Note that at $x = 0$ the derivative of ReLU is undefined. We will discuss ways to handle such cases in **??**. Next, let us expand the derivative definition to multivariate functions.

Definition 1.2.2 — Partial Derivative. Let $f : \mathbb{R}^d \to \mathbb{R}$. The partial derivative of f at point $\mathbf{x} \in \mathbb{R}^d$ with respect to x_i is defined as

$$\begin{aligned} \frac{\partial}{\partial x_i} f(\mathbf{x}) &:= \lim_{a\to 0} \frac{f(\mathbf{x} + a\mathbf{e}_i) - f(\mathbf{x})}{a} \\ &= \lim_{a\to 0} \frac{f(x_1,\ldots,x_i+a,\ldots,x_d) - f(x_1,\ldots,x_i,\ldots,x_d)}{a} \end{aligned}$$

where $\mathbf{e}_i$ is the i-th standard basis vector.

Namely, a partial derivative of a function is its derivative with respect to one of its variables, while all others are kept constant.

■ **Example 1.3 — Max.** For $f(\mathbf{x}) = max_i(x_1, \ldots, x_d)$ the partial derivatives of f at x_i are:

$$\frac{\partial}{\partial x_i} f(\mathbf{x}) = \begin{cases} 1 & i = argmax(x_1, \ldots, x_d) \\ 0 & i \neq argmax(x_1, \ldots, x_d) \end{cases}$$

■

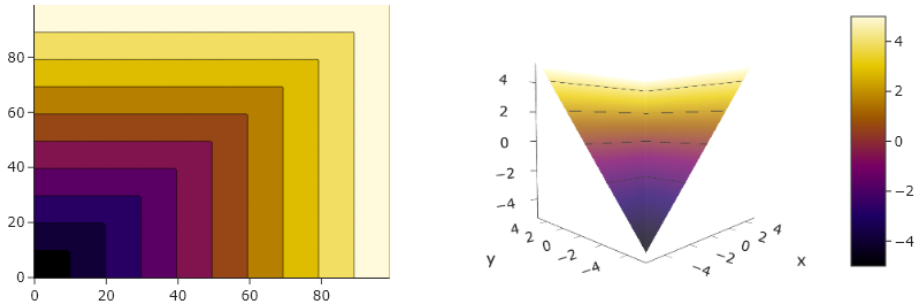


Figure 1.1: *Visualization of bi-variate max function in 3D and 2D. Chapter 1 Examples - Source Code*

■ **Example 1.4 — Polynomial.** For $f(x,y) = x^2 + xy + y^2$ the partial derivatives of f at (x_0, y_0) are

$$\frac{\partial}{\partial x} f(x_0, y_0) = 2x_0 + y_0, \quad \frac{\partial}{\partial y} f(x_0, y_0) = 2y_0 + x_0$$

■

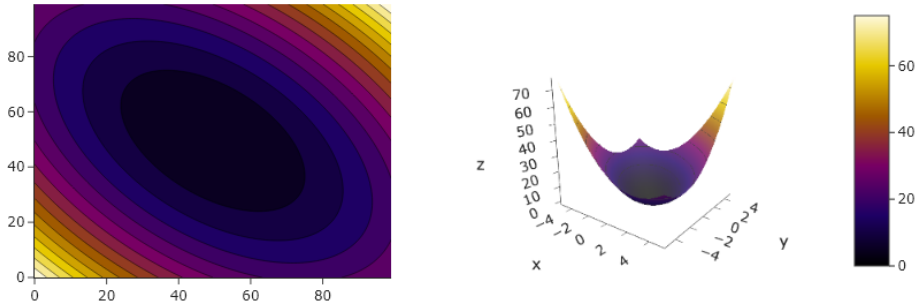


Figure 1.2: *Visualization of bi-variate polynomial of degree 2 in 3D and 2D. Chapter 1 Examples - Source Code*

1.2.1.2 Gradients

Definition 1.2.3 — Gradient. The gradient of f at $\mathbf{x}$ is the vector of partial derivatives:

$$\nabla f(\mathbf{x}) := \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right)^\top$$

■ **Example 1.5** By using the partial derivatives previously calculated of a second order bivariate polynomial $f((x,y) = x^2 + xy + y^2$, the gradient of f at $\mathbf{x}_0 = (x_0, y_0)$ is

$$\nabla f(\mathbf{x}_0) = (2x_0 + y_0, 2y_0 + x_0)^\top.$$

■

Exercise 1.3 — Linear Functional. Let $\mathbf{w} \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}$ defined as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. Compute the gradient of f at point $\mathbf{x}$. ■

Proof. From linearity of the derivative:

$$\frac{\partial}{\partial x_j} f(\mathbf{x}) = \sum_i \frac{\partial}{\partial x_j} f(\mathbf{x})_i = \sum \frac{\partial}{\partial x_j} \mathbf{w}_i \mathbf{x}_i = \mathbf{w}_j$$

Therefore, in vector notation $\nabla f(\mathbf{x}) = \mathbf{w}$. ■

Exercise 1.4 — Norm. Let $f : \mathbb{R}^d \to \mathbb{R}$ be defined by $f(\mathbf{x}) = ||\mathbf{x}||^2$. Compute the gradient of f at point $\mathbf{x}$. ■

Proof. Similar to the previous exercise, using the linearity of the derivative then:

$$\frac{\partial}{\partial x_j} f(x) = \sum_i \frac{\partial}{\partial x_j} x_i^2 = 2x_j$$

which in vector notation can be written as: $\nabla f(\mathbf{x}) = 2\mathbf{x}$. ■

1.2.1.3 Jacobians

In different applications in machine learning, the function we are trying to minimize is a vector-valued function, namely $f : \mathbb{R}^m \to \mathbb{R}^d$.

For example, consider the following scenario. We have gathered historic information of season, time of day, latitude and longitude at different points in the world. Next, we have described some function that given these parameters states the temperature and air-pressure of that location and time. We want to find the the extrema points of this function. In this scenario, the described function gets fours parameters and returns two outputs $f : \mathbb{R}^4 \to \mathbb{R}^2$. In order to find the extrema points we need to generalize the gradient definition to vector-valued functions.

Definition 1.2.4 — Jacobian. Let $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^m$ where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^\top$. The Jacobian of f is the $m \times d$ matrix of all partial derivatives:

$$J_{\mathbf{x}}(\mathbf{f}) := \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_d} \\ \vdots & & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_d} \end{bmatrix}$$

■ **Example 1.6** Let us revisit 1.5 where $f:\mathbb{R}^2\to\mathbb{R}$ defined as $f(\mathbf{x})=x_1^2+x_2^2$. The Jacobian of f is:

$$J_x(\mathbf{f})=\begin{bmatrix}\frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2}\end{bmatrix}=[2x_1,2x_2]=\nabla f(\mathbf{x})^\top$$

■

Notice that for any function where $k=1$ the Jacobian is in fact the transposed gradient vector: $J_{\mathbf{x}}(f)=\nabla f(\mathbf{x})^\top$.

Exercise 1.5 Let $f:\mathbb{R}^d\to\mathbb{R}^m$ defined as $f(\mathbf{x})=A\mathbf{x}$ where $A\in\mathbb{R}^{m\times d}$. Find the Jacobian of f: $J_{\mathbf{x}}(f)$). ■

Proof. Define the set of functions computing each coordinate in the output vector $\forall i\in[m]\quad f_i(\mathbf{x})=A_i^\top\mathbf{x}$. Then the jacobian of f is comprised of the gradients of $f_1,\ldots,f_m$ as rows. Notice that we have already computed the gradient of linear functionals in 1.3 so:

$$J_{\mathbf{x}}(f)=\begin{bmatrix}\nabla f_1(\mathbf{x})^\top\\ \vdots\\ \nabla f_d(\mathbf{x})^\top\end{bmatrix}=\begin{bmatrix}-A_1-\\ \vdots\\ -A_m-\end{bmatrix}=A$$

■

1.2.1.4 Chain Rules

Theorem 1.2.1 — Chain Rule - Univariate. Let $f:\mathbb{R}\to\mathbb{R}$ and $g:\mathbb{R}\to\mathbb{R}$ be two differential functions, then the derivative of the composite $f\circ g$ is:

$$(f\circ g)'\coloneqq(f'\circ g)\cdot g'$$

Namely, for $h(x)=f(g(x))$ then $\forall x\in\mathbb{R}\ h'(x)=f'(g(x))\cdot g'(x)$.

Theorem 1.2.2 — Chain Rule - Multivariate. Let $f:\mathbb{R}^d\to\mathbb{R}^m$ and $g:\mathbb{R}^k\to\mathbb{R}^d$. The Jacobian of the composition $(f\circ g):\mathbb{R}^k\to\mathbb{R}^m$ at $\mathbf{x}$ is

$$J_{\mathbf{x}}(f\circ g)=J_{g(\mathbf{x})}(f)J_{\mathbf{x}}(g)\coloneqq\begin{bmatrix}\frac{\partial f_1(g(\mathbf{x}))}{\partial g_1(\mathbf{x})} & \cdots & \frac{\partial f_1(g(\mathbf{x}))}{\partial g_d(\mathbf{x})}\\ \vdots & & \vdots\\ \frac{\partial f_m(g(\mathbf{x}))}{\partial g_1(\mathbf{x})} & \cdots & \frac{\partial f_m(g(\mathbf{x}))}{\partial g_d(\mathbf{x})}\end{bmatrix}=\begin{bmatrix}\frac{\partial g_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial g_1(\mathbf{x})}{\partial x_k}\\ \vdots & & \vdots\\ \frac{\partial g_d(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial g_d(\mathbf{x})}{\partial x_k}\end{bmatrix}$$

In element-wise notation:

$$J_{\mathbf{x}}(f\circ g)_{i,j}\coloneqq\sum_l\frac{\partial f_i(g(\mathbf{x}))}{\partial g_l(\mathbf{x})}\frac{\partial g_l(\mathbf{x})}{\partial x_j}$$

Exercise 1.6 Let $f(\mathbf{x})=||\mathbf{x}||^2$ and $g(\mathbf{x})=A\mathbf{x}$ for some matrix $A\in\mathbb{R}^{m\times d}$. Calculate the jacobian of $f\circ g$. ■

Proof. From the previous theorem we know that:

$$J_{\mathbf{x}}(f\circ g)=J_{g(\mathbf{x})}(f)J_{\mathbf{x}}(g)$$

As g a linear transformation we have see in 1.5 that $J_{\mathbf{x}}(g) = A$. Notice, that as $Im(f) \subseteq \mathbb{R}$, the jacobian of f equals to the transpose of its gradient. As seen in 1.4, $J_{g(\mathbf{x})}(f) = (2g(\mathbf{x}))^{\top}$. Therefore:

$$J_{\mathbf{x}}(f \circ g) = 2\mathbf{x}^{\top}A^{\top}A$$

■

Next, let us calculate the gradient of the following function: $f(\mathbf{x}) = \frac{1}{2}||A\mathbf{x}-\mathbf{y}||^2$. Applying the chain rule then:

$$\begin{aligned} \nabla f(\mathbf{x}) &= \frac{\partial}{\partial \mathbf{x}}\frac{1}{2}||A\mathbf{x}-\mathbf{y}||^2 \\ &= \frac{1}{2}\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^{\top}A^{\top}A\mathbf{x} - 2\mathbf{y}^{\top}A\mathbf{x} + \mathbf{y}^{\top}y\right) \\ &= A^{\top}A\mathbf{x} - A^{\top}y \\ &= A^{\top}(A\mathbf{x}-\mathbf{y}) \end{aligned} \tag{1.2}$$

This function is known as the Mean Square Error (MSE) and in Figure 2.2 we will use the above derivation in order to find the values of $\mathbf{x}$ that minimize f as above.

■ **Example 1.7 — Soft-Max.** The softmax function defined over $S : \mathbb{R}^d \to [0,1]^d$ returns a vector that its coordinates sum up to 1. It is defined by

$$S(\mathbf{a})_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}}$$

As the coefficients a_j are in the power of the exponent function, all outputted values are strictly positive. Moreover, since the numerator appears in the denominator summed up with some other positive numbers, $S_j < 1$. Therefore, it is in the range $(0,1)$. For example, the 3-element vector $(1,2,5)$ gets transformed into $(0.02, 0.05, 0.93)$. The order of elements by relative size is preserved, and they add up to 1.0. In addition, the third element is now farther away from the first two, Its softmax value is dominating the overall slice of size 1.0 in the output.

Intuitively, the $softmax$ function is a "soft" version of the $argmax$ function. Instead of just selecting one maximal element, softmax breaks the vector up into segments with the maximal input element getting a proportionally larger chunk, but the other elements getting some of it as well. Softmax is often used in neural networks, to map the non-normalized output to a probability distribution over predicted output classes.

Let us calculate the derivative of the softmax function. Denote $g_{i(\mathbf{a})} := e^{a_i}$ and $h(\mathbf{a}) := \sum_{k=1}^{N} g_i(\mathbf{a})$. So:

$$\frac{\partial S_i}{\partial a_j} = \frac{\partial}{\partial a_j}\frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} = \frac{\partial}{\partial a_j}\frac{g_i}{h}$$

Note that for any a_j the derivative of h is e^{a_j}. In the case of g_i, when deriving with respect to a_j we get that the derivative is e^{a_j} only if $i = j$. Otherwise, the derivative is 0. Therefore, the derivative of S_i in the case where $i = j$ is:

$$\frac{\partial}{\partial a_j}\frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} = \frac{e^{a_i}(\sum_{k=1}^{N} e^{a_k}) - e^{a_i}e^{a_j}}{(\sum_{k=1}^{N} e^{a_k})^2} = \frac{e^{a_i}}{(\sum_{k=1}^{N} e^{a_k})} \cdot \frac{(\sum_{k=1}^{N} e^{a_k}) - e^{a_j}}{(\sum_{k=1}^{N} e^{a_k})} = S_i(1 - S_j)$$

It is left to show the derivative in the case where $i \neq j$. ■

1.2.2 First Order Function Approximation

As motivated in the begining of this section, we are often interested in finding the minimizers of some multivariate objective function. Many times, these functions are very hard, or even impossible, to solve analytically. In such cases we might consider approximating the true function with a simpler one that we are able to solve.

Consider a function $f : \mathbb{R} \to \mathbb{R}$ and recall the definition of the Taylor series:

$$T(x_0 + x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} x^n = f(x_0) + f'(x_0)x + \frac{1}{2} f''(x_0)x^2 + \dots$$

A linear approximation (or first order approximation) is an approximation of a general function using a linear function. For a twice continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$, Taylor's theorem implies that for small x

$$f(x_0 + x) \approx f(x_0) + f'(x_0)x$$

We can now extend this theorem to define linear approximations of multivariate functions.

Definition 1.2.5 — Linear Approximation. Let $f : \mathbb{R}^d \to \mathbb{R}$ and $\mathbf{p}_0 \in \mathbb{R}^d$. The linear approximation of f for every $\mathbf{p}$ near $\mathbf{p}_0$ is defined as

$$f(\mathbf{p}_0) + \langle \nabla f(\mathbf{p}_0), \mathbf{p} - \mathbf{p}_0 \rangle$$

Equivalently, if we treat $\mathbf{p}$ as the deviation from $\mathbf{p}_0$ then:

$$f(\mathbf{p}_0 + \mathbf{p}) \approx f(\mathbf{p}_0) + \langle \nabla f(\mathbf{p}_0), \mathbf{p} \rangle$$

So if for example, we consider a bivariate function $f : \mathbb{R}^2 \to \mathbb{R}$, the linear approximation of f near the point (x_0, y_0) is:

$$f(x_0 + x, y_0 + y) \approx f(x_0, y_0) + x \cdot \frac{\partial f(x_0, y_0)}{\partial x} + y \cdot \frac{\partial f(x_0, y_0)}{\partial y}$$

Now, if f is a linear function, intuition dictates that the linear approximation of f would be the function itself. Let $\mathbf{b} \in \mathbb{R}^d$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be defined by $f(\mathbf{x}) = \mathbf{b}^\top \mathbf{x}$. Then, the linear approximation of f at $\mathbf{p}_0$ is

$$\begin{aligned} f(\mathbf{p}_0) + \langle \nabla f(\mathbf{p}_0), \mathbf{p} - \mathbf{p}_0 \rangle &= \mathbf{b}^\top \mathbf{p}_0 + \langle \mathbf{b}, \mathbf{p} - \mathbf{p}_0 \rangle \\ &= \mathbf{b}^\top (\mathbf{p}_0 + \mathbf{p} - \mathbf{p}_0) = f(\mathbf{p}) \end{aligned}$$

Exercise 1.7 Let $f : R^2 \to \mathbb{R}$ be defined by $f(x, y) = \sqrt{x^2 + y^2}$. Calculate the linear approximation of f near $(3, 4)$. ■

Proof. We begin with expressing the gradient of f. So, the partial derivative of f with respect to first argument at point x is:

$$\frac{\partial}{\partial x} = f(x_0, y_0) = 2x_0 \cdots \frac{1}{2\sqrt{x_0^2 + y_0^2}}$$

Therefore the gradient of f is $\nabla f(\mathbf{x}) = \left(\frac{x_0}{\sqrt{x_0^2+y_0^2}}, \frac{y_0}{\sqrt{x_0^2+y_0^2}}\right)^\top$. So for a point (x,y) in the vicinity of $(3,4)$ the linear approximation is:

$$f(3+x,4+y) \approx 5 + \frac{3}{5}x + \frac{4}{5}y$$

If for example $x = 0.1, y = 0.2$ then $f(3+0.1,4+0.2) = 5.2201.. \approx 5.22 = 5 + 3/5 \cdot 0.1 + 4/5 \cdot 0.2$ ■

Another use of first order approximations is as follows. Suppose we investigate some objective function $f : \mathbb{R}^d \to \mathbb{R}$ at point $\mathbf{p}_0 \in \mathbb{R}^d$. Now, we would like to "take a step" in $\mathbb{R}^d$ in the direction where f grows with the fastest rate. That is, we are searching for the direction in space, that if we move in, f grows the most. How can we find such direction?

Recall that $f(\mathbf{p}_0 + \mathbf{p}) \approx f(\mathbf{p}_0) + \nabla f(\mathbf{p}_0) \cdot \mathbf{p}$. In addition, the angle between $\nabla f(\mathbf{p}_0)$ and $\mathbf{p}$ is $\nabla f(\mathbf{p}_0) \cdot \mathbf{p} = ||\nabla f(\mathbf{p}_0)|| \cdot ||\mathbf{p}|| \cos\theta$. As we are only interested in the direction in which to step, and not the step size, let us assume that $||(\mathbf{p})|| = 1$. Since $\cos x \in [-1,1]$, the direction that maximizes f is $\mathbf{p}_{max} \coloneqq \nabla f(\mathbf{p}_0) / ||\nabla(\mathbf{p}_0)||$. Similarly the direction that minimizes f is $\mathbf{p}_{min} \coloneqq -\nabla f(\mathbf{p}_0) / ||\nabla(\mathbf{p}_0)||$. So, if we adapt a step-wise approach, for sufficiently small enough steps sizes, for maximizing/minimizing f, moving in the direction of the gradient or opposite to the gradient is a good approach. We will revisit this concept in **??**.

1.3 Probability and Statistics

A great deal of the machine learning principles are based on concepts from probability theory and statistics. As we will encounter later on, we are often facing a scenario where for the task in hand we:

- Define some parameter (or set of parameters) of interest.
- Define some objective function that depends on this parameter.
- Have some data relevant to the task, from which we try to **estimate** the **parameter** by optimizing the **objective function**.

Consider the following task. We are preparing the traditional "Introduction to Machine Learning Hackathon" and expecting 500 hungry students. For this event we want to order enough pizzas for all the participants. How should we know how many pizzas to order? Let us denote the average number of pizza slices each participant would like to eat by μ. In statistics, this average is called the *population mean*, meaning simply that it refers to an average over the whole group which is under consideration. In this case, all N participants.

Then, a reasonable guess for the number of pizza slices we should order is $X = N \times \mu$. As we do not know μ, and it would take too long to ask each of the participants how many pizza slices they would like to have, we must come up with some way to **estimate** a value of μ. To do so, let us *sample independently* participants: we randomly choose one, call and ask how many pizza slices they would like to eat. We write the answer down as x_1. We then randomly call another one (which could accidentally be the same one) and write the second answer as x_2. We continue m times. Now, to estimate the population mean, μ, we use the average of the answers acquired. In statistics this is

referred to as the *sample mean* and is given by

$$\widehat{\mu} \equiv \frac{1}{m}\sum_{i=1}^{m} x_i.$$

Namely, the empirical average of the parameter in question. Finally, we *predict* that we will need $T = N \times \widehat{\mu}$ pizza slices. The $\widehat{\mu}$ (hat) symbol indicates that this is an estimator of the parameter μ.

Another question is how many participants should we contact? Intuitively, if we ask a very small set of participants our estimation of $\widehat{\mu}$ (and therefore our prediction of T would be very wrong. As we increase the number of participants asked, the *sample size*, we will get a more accurate estimation of $\widehat{\mu}$. To answer how accurate the estimation, let us suppose that we know the true value of μ. Then we should look at the difference between our estimation and the true value, and how it changes as m changes. We can then choose an accuracy parameter $\varepsilon > 0$ and require that $|\widehat{\mu} - \mu| \leq \varepsilon$. By solving for m we will get a bound on the number of participants we should ask.

1.3.1 Fundamental Definitions

1.3.1.1 Probability Space

The basic object in probability theory is that of a *probability space* which consists of two ingredients: the *sample space* and the *probability function*.

Definition 1.3.1 A *sample space* Ω is a set that contains all possible outcomes. $\omega \in \Omega$ denotes a single outcome.

For example the probability space of m coin tosses can be defined as $\Omega = \{H,T\}^m$ or the probability space of coin tosses until heads comes up can be defined as $\Omega = \left\{T^{n-1}H : n \in \mathbb{N}\right\}$. We can also define more complex probability spaces such as the height, weight, blood pressure and sex of patients with a medical condition, as well as the success or failure of an experimental drug: $\Omega = [0,\infty) \times [0,\infty) \times [0,\infty) \times \{M,F\} \times \{Yes,No\}$. As we will see, we often model machine learning tasks with some probability space (even if we do not explicitly define it).

Definition 1.3.2 An *event* A is any subset of possible outcomes, $A \subseteq \Omega$.

Definition 1.3.3 A *probability space* is a tuple [a] $(\Omega, \mathcal{D})$ where Ω is a *sample space* and $\mathcal{D} : 2^{\Omega} \to \mathbb{R}$ is a probability function such that

1. $\mathcal{D}(\Omega) = 1$
2. for all $\omega \in \Omega, \mathcal{D}(\omega) \in [0,1]$
3. for all $A,B \subseteq \Omega$ such that $A \cap B = \emptyset$, we have $\mathcal{D}(A \cup B) = \mathcal{D}(A) + \mathcal{D}(B)$.

[a] For our needs this simple and intuitive definition is sufficient, though, richer definitions exist.

■ **Example 1.8** Suppose we throw two fair dice, then $\Omega = \{1,\ldots,6\}^2$ and $\mathcal{D}((i,j)) = \frac{1}{36}$ ■

Claim 1.3.1 For all $A,B \subseteq \Omega$: $\mathcal{D}(A \cup B) = \mathcal{D}(A) + \mathcal{D}(B) - \mathcal{D}(A \cap B)$

Proof. Let us notice that for both events A, B it holds that:

$$A = (A \setminus B) \cup (A \cap B)$$
$$B = (B \setminus A) \cup (A \cap B)$$

and that $A \cup B = (A \setminus B) \cup (B \setminus A) \cup (A \cap B)$. Therefore

$$\begin{aligned} \mathcal{D}(A \cup B) &= \mathcal{D}(A \setminus B) + \mathcal{D}(B \setminus A) + \mathcal{D}(A \cap B) \\ &= \mathcal{D}(A) - \mathcal{D}(A \cap B) + \mathcal{D}(B) - \cancel{\mathcal{D}(A \cap B)} + \cancel{\mathcal{D}(A \cap B)} \end{aligned}$$

■

Definition 1.3.4 Events $A, B \subseteq \Omega$ are said to be *independent* if the occurrence of one does not affect the probability of occurrence of the other, namely:

$$\mathcal{D}(A \cap B) = \mathcal{D}(A) \cdot \mathcal{D}(B)$$

Exercise 1.8 Show that if A and B are independent then A and B^c $(\Omega \setminus B)$ are also independent. ■

Proof. As $\mathcal{D}(A) = \mathcal{D}(A \cap B) + \mathcal{D}(A \cap B^c)$ Then $\mathcal{D}(A \cap B^c) = \mathcal{D}(A)(1 - \mathcal{D}(B)) = \mathcal{D}(A) \cdot \mathcal{D}(B^c)$ ■

Theorem 1.3.2 — The union bound. Let $(\Omega, \mathcal{D})$ be a probability space. The probability function is *sub-additive*, i.e., for any sequence (A_k) of events,

$$\mathcal{D}\left(\cup_{k=1}^{\infty} A_k\right) \leq \sum_{k=1}^{\infty} \mathcal{D}\left(A_k\right)$$

Proof. Let $B_1 = A_1$. For each $k \in \{2, 3, \ldots\}$, let $B_k = A_k \setminus \cup_{i=1}^{k-1} A_i$. Note that the $B_1, \ldots, B_k$ are disjoint, and that $\cup_{i=1}^{k} A_i = \cup_{i=1}^{k} B_i$. Also, since $B_k \subseteq A_k$, $\mathcal{D}(B_k) \leq \mathcal{D}(A_k)$ for every $k \in \mathbb{N}$. It follows that:

$$\mathcal{D}\left(\cup_{k=1}^{\infty} A_k\right) = \mathcal{D}\left(\cup_{k=1}^{\infty} B_k\right) = \sum_{k=1}^{\infty} \mathcal{D}\left(B_k\right) \leq \sum_{k=1}^{\infty} \mathcal{D}\left(A_k\right)$$

■

1.3.1.2 Random Variables

So far we have asked questions like: does an event happen or not and what is the probability of the event. How would we be able to model different questions such as, "how much". For example, suppose if a coin flip turns head we get a dollar, we can ask how many dollars would we get after 3 rounds?

In order to answer this question, we should identify the sample space, which is

$$\Omega = \{HHH, HHT, HTH, THH, HTT, THT, TTH, TTT\}$$

Then, we would need to define a reward function that will quantify ones's profit for each outcome of the experiment ($\omega \in \Omega$): $X(HHH) = 3$, $X(HHT) = 2$ and so on. This quantifying function X is called a *random variable* (RV).

Definition 1.3.5 Given a probability space $(\Omega, \mathcal{D})$, a real-valued *random variable (RV)* is a function $X : \Omega \to \mathbb{R}$.

Definition 1.3.6 Let Ω be a discrete probability space and X a random variable over Ω. The ***probability mass function (PMF)*** of X is defined as:

$$\mathcal{D}(\{X = x\}) = \sum_{\omega : X(\omega) = x} \mathcal{D}(\omega)$$

Instead of writing $\mathcal{D}(\{X = x\})$ we can simplify notation as $\mathcal{D}_X(\{x\})$, and further simply as $\mathcal{D}(x)$. This simplified notation, omitting any reference to the random variable, is useful when the context is clear and we shall use it often. This omission is similar to saying, for example, "the probability of (H,T) is 1/4" instead of "the probability of getting (H,T) by tossing a fair coin twice is 1/4".

■ **Example 1.9** A fair coin is tossed twice. The probability space is given by $\Omega = \{T,H\}^2$. As this is a fair coin then $\forall \omega \in \Omega \; \mathcal{D}(\omega) = \frac{1}{4}$. Let X denote the number of heads obtained in each outcome (ω). The possible values of X are 0, 1, and 2. The probability of each value is

$$\mathcal{D}(x) = \begin{cases} \frac{1}{4} & x = 0, 2 \\ \frac{1}{2} & x = 1 \\ 0 & \text{else} \end{cases}$$

■

Definition 1.3.7 Let Ω be a continuous probability space and X a random variable over Ω. We say that X is a *continuous random variable* if there exists $f(x) \geq 0$ so that we can write, for every $D \subset \mathbb{R}$

$$\mathcal{D}(X \in D) = \int_D f(x)\, dx$$

The function f is called the ***probability density function*** **(PDF)** of X.

In particular, this means that for any $a, b \in \mathbb{R}$, where $a \leq b$:

$$\mathcal{D}(X \in [a,b]) = \int_a^b f(x)\, dx$$

As before, f should actually be denoted by $f_X(x)$ to emphasize that it characterizes the random variable X, but often the subscript X is omitted. In this course we assume that $f(x)$ is a regular function and therefore the probability mass function for a single point is 0: $\mathcal{D}(X = a) = \int_a^a f(x)\, dx = 0$ for any $a \in \mathbb{R}$. Note that the probability density function satisfies that

- $f(x) \geq 0$
- $\int_{-\infty}^{\infty} f(x)\, dx = 1$
- $f(x)$ is a probability *density*, not a probability. For example, it could be that $f(x) > 1$.

1.3.1.3 Mean and Variance

When dealing with random variables (and later on with distributions) we are often interested in different measures of these random variables. The two most common and widely used measures are the mean (expected value) and variance.

Definition 1.3.8 The ***expected value*** of a random variable X is

$$\mathbb{E}[X] := \sum_{x \in \mathcal{X}} x\mathcal{D}(x) \qquad \text{or} \qquad \mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x)\, dx$$

Claim 1.3.3 Let X, Y be two random variable with finite expected values $\mathbb{E}[X], \mathbb{E}[Y]$. Then:

- Linearity of expectation: $\mathbb{E}[aX + Y] = a\mathbb{E}[X] + \mathbb{E}[Y]$.
- Expectation of function of random variable (Law of the unconscious statistician): $\mathbb{E}[g(X)] = \sum g(x)\mathcal{D}(x)$.
- If X and Y are independent then: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- If $X \leq Y$ then $\mathbb{E}[X] \leq \mathbb{E}[Y]$.

Definition 1.3.9 Let X be a random variable with $\mathbb{E}[X]$, then the ***variance*** of X is

$$Var(X) := \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

The ***standard deviation*** of X is defined as $\sigma := \sqrt{Var(X)}$.

Claim 1.3.4 Let X, Y be two random variables with finate expected values and variances. Then:

- $Var(X) = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2$
- For scalars $a, b \in \mathbb{R}$, $Var(aX + b) = a^2 Var(X)$
- The variance of X is non-negative: $Var(X) \geq 0$. Equality holds when $Var(X) = 0 \iff X$ is a constant.
- $$\begin{aligned} Var(X+Y) &= \mathbb{E}\left[(X + Y - \mathbb{E}(X) - \mathbb{E}(Y))^2\right] \\ &= Var(X) + Var(Y) + 2\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \end{aligned}$$
- If $X_1, \ldots, X_n$ are independent then $V(\sum X_i) = \sum V(X_i)$

Definition 1.3.10 The ***covariance*** (joint variability) of X and Y is the expected value of the product of their deviations from the expected values

$$Cov(X,Y) := [(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

The covariance is also denoted as $\sigma(X,Y)$ or σ_{XY}.

Claim 1.3.5 Let X, Y be two random variables with finite expectation and variance. Then:

- Symmetry: $Cov(X,Y) = Cov(Y,X)$.
- For scalars $a, b, c, d \in \mathbb{R}$, $Cov(aX + b, cY + d) = a \cdot c \cdot Cov(X,Y)$
- $Cov(X,X) = Var(X)$

Note the different meaning implied by the above definitions: The variance measures the variation of a single random variable (like the height of a person in a population), whereas covariance is a measure of how much two random variables (like the height and weight) vary together.

■ **Example 1.10** Let Z be a random variable such that $P(Z=1)=p, P(Z=0)=1-p$. The variance of Z is: $Var[Z]=\mathbb{E}\left[Z^2\right]-(\mathbb{E}[Z])^2=p-p^2=p(1-p)$. ■

Exercise 1.9 Let $X \sim Unif([a,b])$ for $a,b \in \mathbb{R}$. Calculate the expectation and variance of X. ■

Proof.

$$\begin{aligned}\mathbb{E}[X] &= \frac{1}{b-a}\int_a^b x dx = \frac{b+a}{2}\\ \mathbb{E}\left[X^2\right] &= \frac{1}{b-a}\int_a^b x^2 dx = \frac{b^2+ab+a^2}{3}\end{aligned}$$

$$Var(X) = \frac{b^2+ab+a^2}{3} - \frac{(b+a)^2}{4} = \frac{(b-a)^2}{12}$$

■

1.3.2 A Little Statistics: Mean and Variance Estimation

The focus of this book is *Statistical* Machine Learning. When we study *Probability* we assume a known distribution and calculate probability of events of interest. When we study *Statistics* we turn the question on its head: using samples from an unknown probability distribution, we try to *estimate* or *test* properties of the unknown distribution. Recall the pizza slices example: we were interested in finding out how many pizza slices the participants will eat. We did not know what is the distribution of number of pizza slices participants eat. Given *samples* that were drawn from this unknown distribution, we attempted to *estimate* the distribution's mean. Let us formulate what we have done above and estimate both the expected value and the variance.

Let X be a random variable and let $\mathcal{D}(x)$ be its (unknown) distribution. Let $X_1,\ldots,X_m \overset{i.i.d}{\sim} D(x)$ be m random variables of this distribution. The *i.i.d* denotes these are *independent and identically distributed* random variables. We are given *data samples*: $x_1,\ldots,x_m$, where each x_i is a *number* obtained by sampling X_i. In statistics, $\mathbb{E}[X]$ is called the *population* mean - "population" added to emphasize that it is the mean of the unknown distribution, and not of the observed data. Similarly, $Var(X)$ is called the *population variance*.

There are many ways to estimate $\mathbb{E}[]$ and $Var(X)$ based on those m samples, and each one is called an *estimator* (that is, the function that estimates the value of some parameter of a random variable). Throughout this book we are going to use different estimators for different tasks. For the case of sample mean and variance we will use:

- ***Sample mean*** (an estimator for the population mean)

$$\widehat{\mu}_X = \frac{1}{m}\sum_{i=1}^{m} x_i$$

- ***Sample variance*** (an estimator for the population variance)

$$\widehat{\sigma}_X^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i-\widehat{\mu}_X)^2$$

As meantioned before, whenever the context is clear, we will omit the subscript X and write $\widehat{\mu}$ and $\widehat{\sigma}^2$ instead of $\widehat{\mu}_X$ and $\widehat{\sigma}_X^2$. Note that the sample variance is somewhat different from the variance of the x_i's because of the division by $m-1$ instead of m. The difference between the two will be explained later on in the book.

Exercise 1.10 Let X be some random variable. Show that the expected values of the samples mean and sample variance estimators equals to the true parameters they estimate. That is their expectation is the population mean and population variance.

Solution. Taking the expectation value of the sample mean one gets

$$\mathbb{E}(\widehat{\mu}_X) = \mathbb{E}\left(\frac{1}{m}\sum_{i=1}^{m} x_i\right) = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}(x_i) = \mathbb{E}(X)\frac{1}{m}\sum_{i=1}^{m} 1 = \mathbb{E}(X) = \mu_X$$

where we used the linearity property of the expectation and that the samples are $i.i.d$. Taking the expectation value of the sample variance one gets

$$\begin{aligned}\mathbb{E}\left(\widehat{\sigma}_X^2\right) &= \frac{1}{m-1}\sum_{i=1}^{m}\mathbb{E}\left((x_i-\widehat{\mu}_X)^2\right)\\ &= \frac{1}{m-1}\sum_{i=1}^{m}\mathbb{E}\left(x_i^2 - 2x_i\frac{1}{m}\sum_{j=1}^{m}x_j + \frac{1}{m^2}\sum_{k,j=1}^{m}x_k x_j\right)\end{aligned}$$

The X_i's are i.i.d which implies that $\mathbb{E}(x_i) = \mathbb{E}(X)$ and $\mathbb{E}\left(x_i^2\right) = \mathbb{E}\left(X^2\right)$ for every i, as well as that $\mathbb{E}(x_k x_j) = \mathbb{E}^2(X) = \mu_X^2$ for every $k \neq j$. Substituting into the above sums we obtain that:

$$\mathbb{E}\left(\widehat{\sigma}_X^2\right) = \mathbb{E}\left(X^2\right) - \mathbb{E}^2(X) = Var(X) = \sigma_X^2$$

■

Definition 1.3.11 Let $\widehat{\theta}$ be an estimator of θ. The *bias* of $\widehat{\theta}$ is the expected deviation between θ and the estimator: $B\left(\widehat{\theta}\right) := \mathbb{E}\left[\widehat{\theta}\right] - \theta$. $\widehat{\theta}$ is said to be *unbiased* if $B\left(\widehat{\theta}\right) = 0$.

From what we have seen in the above exercise, $\widehat{\mu}_X$ is an unbiased estimator of the population mean, μ_X, and $\widehat{\sigma}_X^2$ is an unbiased estimator of the population variance σ_X^2. If we calculate the expected variance of the x_i's using $\frac{1}{m}\sum_{i=1}^{m}(x_i-\hat{\mu}_X)^2$, we will see that we do not get the value σ_X^2, but rather is $\frac{m-1}{m}\sigma_X^2$. Therefore this estimator is a *biased* estimator of σ_X^2.

R Biased estimator, though there expectation is not the true value of the parameters can be useful in different cases. Therefore, we would sometimes use biased estimators and sometimes unbiased estimators.

1.3.3 Multivariate Probabilities

Up to now, we only dealt with random variables taking a single value. In many machine learning applications however, we often face a situation where we have multiple properties we would like to use in order to perform prediction. To do that we consider multivariate random variables and multivariate distributions.

Definition 1.3.12 — Random vector. A (column) ***random vector***: $X := (X_1,\ldots,X_d)^\top$, is a finite collection of random variables, denoted $X_1,\ldots,X_d$, defined on a common probability space $(\Omega,\mathcal{D})$.

Definition 1.3.13 — Joint distribution. Given random variables $X_1, \ldots, X_d$, that are defined on a probability space, the joint probability distribution for $X_1, \ldots, X_d$ is a probability distribution that gives the probability that each of $X_1, \ldots, X_d$ falls in any particular range (for continuous RVs) or discrete set (for discrete RVs) of values specified for that variable.

Definition 1.3.14 — Joint PDF of a random vector. Two random variables X_1 and X_2 are ***jointly continuous*** if there exists a non-negative function $f_{X_1,X_2} : \mathbb{R}^2 \to \mathbb{R}$, such that, for any set $A \in \mathbb{R}^2$, it holsd that

$$\mathcal{D}\left((X,Y) \in A\right) = \int_A f_{X_1,X_2}(x_1,x_2)\, dx_1 dx_2$$

The function f_{X_1,X_2} is called the ***joint probability density function (JPDF)*** of X_1 and X_2. Often, if the context is clear, one omits the subscripts of f and simply writes $f(x_1,x_2)$.

Given $X := (X_1, \ldots, X_d)^\top$, each of the scalar random variables $X_1, \ldots, X_d$ can be characterized by its PDF. However, unless the scalar RV's are mutually independent, the PDF of each coordinate of a random vector does not completely describe the probabilistic behavior of the whole vector. For instance, the behavior of the random vectors $X = (X_1, X_2)$ and $\tilde{X} = (X_1, X_1)$ is drastically different even if X_1 and X_2 have an identical PDF. Consider for example, $X_1, X_2 \sim Unif([-a,a])$ and $X_2 = -X_1$ and compare the probability to find X in the square $[0,1] \times [0,1]$ to that of finding $\tilde{X}$ in the same square.

1.3.3.1 Normal Distribution

As an example of random vectors, we now consider specific family of distributions - the Multivariate Normal (also called Multivariate Gaussian) Distributions. Due to different limit theories, such as the Central Limit Theorem, and due to nice mathematical properties of this distribution, it is often used to model different probabilistic scenarios.

Definition 1.3.15 — (Univariate) Normal Distribution. A random variable x has a ***normal distribution*** with expectation μ and variance σ^2 if it has a PDF of the form:

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

In this case we write: $x \sim \mathcal{N}(\mu, \sigma^2)$

Definition 1.3.16 — Multivariate Normal Distribution. A random vector $\mathbf{x} \in \mathbb{R}^d$ has a ***multivariate normal distribution*** with expectation μ and covariance matrix (see definition below) Σ if it has a joint PDF of the form:

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)\right\}$$

In this case we write: $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$

Observe that definition 1.3.16 is generalization of 1.3.15, i.e. when $d = 1$ both definitions are same. Often, we are interested only in how one of the variates distributes, without the influence of the rest.

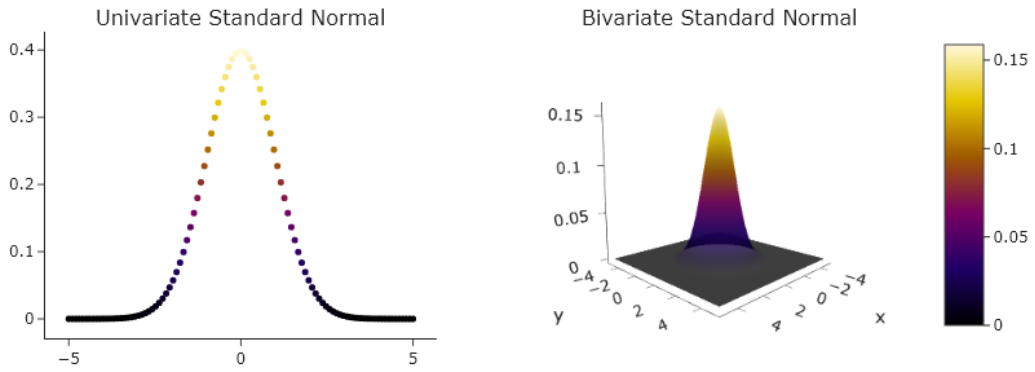


Figure 1.3: *Uni- and bivariate normal distributions.* *Chapter 1 Examples - Source Code*

For example, suppose we have joint probability function of the variables "height" and "weight" but we are interested in how the "height" distributes. To do so we would like to integrate out the "weight" variable. This leads to the following definition:

Definition 1.3.17 The ***Marginal distribution*** of a subset if a collection of random variables with a joint probability distribution, is the probability distribution of the variables in the set:

$$f(\mathbf{x}) = \int_{\mathbf{y}} f(\mathbf{x},\mathbf{y})\, d\mathbf{y}$$

where the $\mathbf{y}$ integration is over all the random variables not in $\mathbf{x}$.

Exercise 1.11 — Marginal of Bivariate Normal. Let $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$ where $\mathbf{x} \in \mathbb{R}^2, \mu = (\mu_1, \mu_2)^\top$ and $\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$ Find the PDF of the marginal distribution of x_1.

Observe that we can write the PDF as follows:

$$\begin{aligned}
f(\mathbf{x}) &= \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right) \\
&= \frac{1}{\sqrt{(2\pi)^2\sigma_1^2\sigma_2^2}} \exp\left(-\frac{1}{2}\begin{bmatrix} x_1-\mu_1 & x_2-\mu_2 \end{bmatrix}\begin{bmatrix} \sigma_1^{-2} & 0 \\ 0 & \sigma_2^{-2} \end{bmatrix}\begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}\right) \\
&= \frac{1}{\sqrt{(2\pi)^2\sigma_1^2\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - \frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right) \\
&= \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right)
\end{aligned}$$

Using the definition of the marginal distribution:

$$\begin{aligned}
f(x_1) &= \int_{-\infty}^{\infty} f(x_1,x_2)\, dx_2 \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right) dx_2 \\
&= \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right) \cdot \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right) dx_2
\end{aligned}$$

Notice that now we integrate a function of a uni-variate Gaussian for all values $x_2 \in (-\infty, +\infty)$. Therefore this integral equals to 1 and we are left with:

$$f(x_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2\right)$$

Which by definition 1.3.15 is a univariate Gaussian of the form $x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$. ■

1.3.3.2 Covariance Matrix

Unlike the univariate case, when dealing with a multivariate distribution the variance is a matrix rather than a vector.

Definition 1.3.18 — Covariance Matrix. Let $\mathbf{X} \coloneqq (X_1, X_2, ..., X_d)^\top$ be a random vector. The *Covariance Matrix* Σ is the $d \times d$ matrix whose (i, j) entry is the covariance $\Sigma_{ij} \coloneqq \sigma(X_i, X_j)$:

$$\Sigma \coloneqq \begin{pmatrix} \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_1 - \mathbb{E}[X_1])] & \dots & \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_d - \mathbb{E}[X_d])] \\ \vdots & \ddots & \vdots \\ \mathbb{E}[(X_d - \mathbb{E}[X_d])(X_1 - \mathbb{E}[X_1])] & \dots & \mathbb{E}[(X_d - \mathbb{E}[X_d])(X_d - \mathbb{E}[X_d])] \end{pmatrix}$$

- In matrix notation we can express the covariance matrix as:

$$\Sigma \coloneqq \mathbb{E}\left[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top\right]$$

- The diagonal elements of Σ are $\sigma(X_i, X_i) \equiv \sigma_{X_i}^2 \equiv Var(X_i)$. *Sigma* is a symmetric positive semi-definite matrix.

Back to statistics, let us consider how to estimate the covariance matrix based on a given sample. In this context, Σ is called the *population covariance matrix*. Let us start with the bi-variate case $d = 2$. Consider a two-dimensional random vector $\mathbf{X} = (X_1, X_2)^\top$, where, for example, X_1 is the height of a person and X_2 is the weight.

Taking a sample of m people from the population, we equip the data elements with two indices, the first stands for person's designated number and the second for the feature (1 for height, 2 for weight). Therefore, the height samples are denoted by $x_{1,1}, \dots, x_{1,m}$ and the corresponding weight samples are $x_{2,1}, \dots, x_{2,m}$. Note that we can write the data as a matrix $X \in \mathbb{R}^{m \times d}$ where m is the *sample size* and d is the dimension (number of random variables). In our case, $d = 2$ and

$$X = \begin{bmatrix} x_{1,1} & x_{2,1} \\ \vdots & \vdots \\ x_{m,1} & x_{m,2} \end{bmatrix} = (\mathbf{x}_1, \dots, \mathbf{x}_m)^\top$$

Definition 1.3.19 — Sample Covariance. The unbiased estimator of the ***sample covariance*** of the i'th and j'th random variables is given by:

$$\widehat{\sigma}(X_i, X_j) = \frac{1}{m-1} \sum_{k=1}^{m} (x_{k,i} - \widehat{\mu}_i)(x_{k,j} - \widehat{\mu}_j)$$

where $\widehat{\mu}_i$ is the sample mean of the random variable X_i.

In particular, notice that for the case where $i = j$ we are left with the unbiased estimator previously seen. We can now define the sample covariance matrix $\widehat{\Sigma}$.

Definition 1.3.20 — Sample Covariance Matrix. Let $X = (X_1, \ldots, X_d)^\top$ be a d-dimensional random vector. Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^d$ be m i.i.d samples of X. The ***sample covariance matrix*** is a square d-by-d matrix $\widehat{\Sigma}$ such that $\widehat{\Sigma}_{i,j} = \widehat{\sigma}(X_i, X_j) \quad i, j = 1, \ldots, d$.

In matrix notation, the (biased) estimator for the sample covariance matrix is given by:

$$\widehat{\Sigma} := \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i - \widehat{\mu})(\mathbf{x}_i - \widehat{\mu})^\top = \frac{1}{m} \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}}$$

for $\widetilde{\mathbf{X}}$ being the centered matrix: $\widetilde{\mathbf{X}}_{\cdot,i} := \mathbf{X}_{\cdot,i} - \widehat{\mu}$. The unbiased estimator is given by:

$$\widehat{\Sigma} := \frac{1}{m-1} \sum_{i=1}^{m} (\mathbf{x}_i - \widehat{\mu})(\mathbf{x}_i - \widehat{\mu})^\top = \frac{1}{m-1} \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}}$$

Exercise 1.12 Let $\mathbf{X} = \begin{pmatrix} 150 & 45 \\ 170 & 74 \\ 184 & 79 \end{pmatrix}$ be samples of height and weight of 3 different people. Calculate the covariance matrix of the given sample.

Let us begin with centering the data. That is, subtract the empirical mean from each sample The sample mean is: $\widehat{\mu} = (168, 66)^\top$, so:

$$\mathbf{X}_{centered} = \mathbf{X} - \begin{pmatrix} 168 & 66 \\ 168 & 66 \\ 168 & 66 \end{pmatrix} = \begin{pmatrix} 150 & 45 \\ 170 & 74 \\ 184 & 79 \end{pmatrix} - \begin{pmatrix} 168 & 66 \\ 168 & 66 \\ 168 & 66 \end{pmatrix} = \begin{pmatrix} -18 & -21 \\ 2 & 8 \\ 16 & 13 \end{pmatrix}$$

Now, following the definition of the sample covariance matrix:

$$\widehat{\Sigma} = \frac{1}{3-1} \mathbf{X}_{centered}^\top \mathbf{X}_{centered} = \begin{pmatrix} 292 & 301 \\ 301 & 337 \end{pmatrix}$$

■

1.3.3.3 Linear Transformations of the Data Set

Let us look at how linear transformations affect the data set and as a result, the covariance matrix. Linear transformation can be of two types: scaling and rotating. Although we will now focus on the two-dimensional case, these results are easily generalized to higher dimensional data. The covariance matrix for two dimensions, $d = 2$, is

$$\Sigma = \begin{pmatrix} \sigma(X_1, X_1) & \sigma(X_1, X_2) \\ \sigma(X_2, X_1) & \sigma(X_2, X_2) \end{pmatrix}$$

We begin with a sample of points taken *i.i.d* from a bi-variate Gaussian with a zero mean and equal

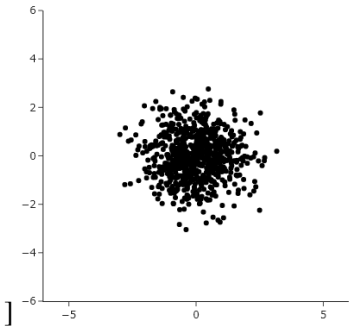


Figure 1.4: *Uncorrelated variables with identical variance. Chapter 1 Examples - Source Code*

variances $\sigma^2_{X_1} = \sigma^2_{X_2} \equiv \sigma^2$ (Figure 1.4). In particular this means that X_1 and X_2 are uncorrelated and the covariance matrix Σ is therefore of the form $\sigma^2 I_2$. In Figure 1.4 we can indeed see that there is no apparent relation between a points x coordinate and y coordinate. When we calculate the sample covariance matrix, as the sample size m increases, our estimation tends to $\widehat{\Sigma} = \sigma^2 I_2$.
Next, will us look at how linear transformations affect our data and the sample covariance matrix $\widehat{\Sigma}$ (Figure 1.5). We start transforming our data by multiplication by a scaling matrix:

$$S = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$$

Notice how this transformation stretches (or shrinks) the x_1 and x_2 components of each sample by multiplying them by s_1 and s_2 respectively. In addition, as the scaling matrix is diagonal, the stretching of the x axis is uncorrelated with that of the y axis. The sample covariance matrix of the transformed data is:

$$\widehat{\Sigma}_{scaled} = \frac{1}{m-1}(\mathbf{X}S)^\top \mathbf{X}S = S^\top \left(\frac{1}{m-1}\mathbf{X}^\top \mathbf{X}\right) S = \begin{pmatrix} (s_1\widehat{\sigma})^2 & 0 \\ 0 & (s_2\widehat{\sigma})^2 \end{pmatrix}$$

.

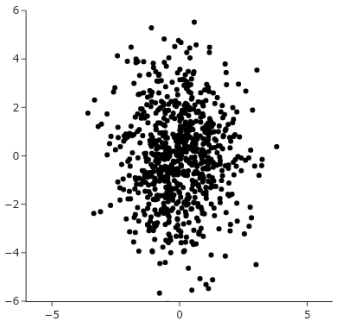


Figure 1.5: *Uncorrelated scaled variables with different scaling in the x and y axis. Chapter 1 Examples - Source Code*

Lastly, over the scaled data let us follow with a rotation transformation (Figure 1.6). Recall that any orthogonal matrix is in fact a rotation matrix, and specifically the rotation matrix of degree θ in $\mathbb{R}^2$

is given by:

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

As we can see in Figure 1.6 the transformed data shows a correlation between the x and y axes. To understand the reasoning behind let us calculate the sample covariance matrix of the scaled and then rotated data.

$$\begin{aligned} \widehat{\Sigma}_{rotated} &= = \tfrac{1}{m-1}(\mathbf{X}SR)^\top(\mathbf{X}SR) = R^\top S^\top\left(\tfrac{1}{m-1}\mathbf{X}^\top\mathbf{X}\right)SR = R^\top\left(S^\top\widehat{\Sigma}S\right)R \\ &= R^\top\begin{bmatrix}(s_1\widehat{\sigma})^2 & 0\\ 0 & (s_2\widehat{\sigma})^2\end{bmatrix}R = \widehat{\sigma}^2\begin{bmatrix} s_1^2\cos^2\theta + s_2^2\sin^2\theta & \sin\theta\cos\theta\left(s_2^2-s_1^2\right)\\ \sin\theta\cos\theta\left(s_2^2-s_1^2\right) & s_1^2\sin^2\theta + s_2^2\cos^2\theta\end{bmatrix} \end{aligned}$$

As the off diagonal element of the covariance matrix is non-zero, the two variables are correlated. Note that if we would not have applied an *asymmetric* scaling (i.e if $s_1 = s_2$), the off diagonal elements would have been zero. This means that rotation alone is not sufficient to induce correlations between random variables.

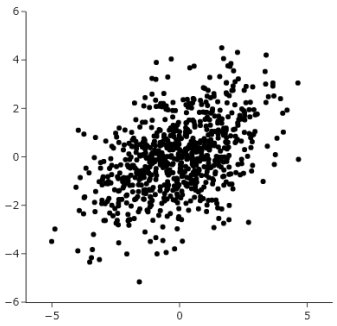


Figure 1.6: *Correlated variables produced by rotation of uncorrelated variables. Chapter 1 Examples - Source Code*

R The form obtained above, where we multiply an orthogonal matrix R by a diagonal matrix and then by another orthogonal matrix $R^\top$ is in fact the EVD of the sample covariance matrix of the transformed data. We will return to this point when we discuss the Principal Component Analysis algorithm (**??**).

1.3.4 Probability Inequalities

Being able to bound (from below or from above) the probability of different events is an important tool in machine learning. They are used in different senarios such as bounding the errors of a learning algorithm or concluding how good can we expect our algorithm to perform for a given sample size (and how this will change if we increase the sample size). For example, the law of large numbers states that if we take the empirical average (the sample mean) of enough i.i.d. random variables, it will be, most likely, very close to the expected value. Probability inequalities provide us a way to estimate how many samples are "enough".

■ **Example 1.11** Suppose we have a bag containing red and blue balls and we would like to estimate the fraction p of red balls in the bag by randomly picking one ball at a time and then putting it back in. The straightforward strategy is to draw m samples and then to use the following estimation ("prediction")

$$\widehat{p} = \frac{1}{m}\sum_{i=1}^{m} x_i = \frac{\text{Number of red balls}}{m}$$

where x_i equals 1 if the i'th ball came out red and 0 otherwise. It is clear that $\mathbb{E}[\widehat{p}] = p$. However, we might not reach this exact value due to the fact that we get only limited information. We are only sampling m balls so we might get "unlucky" and end up with a non-representative sample. Although less likely, this can happen even if m is very large, even larger than the total number of balls, because each time we returned the ball to the bag. Thus, typically, we must settle for a certain accuracy - an additive error - that is, we will be happy to know that if we sample more than m times, our estimation, $\widehat{p}$, is going to be within a distance $\varepsilon > 0$ from the real p. However, no matter how large m is, there is no absolute guarantee that $\widehat{p}$ will have that required accuracy.

Since m is finite, there is always a finite chance we sample the same color again and again. So, at most we can require to have enough confidence, say, with probability of $1-\delta$ where $\delta > 0$ is small, that $\hat{p}$ is accurate enough ("accurate enough" defined as within ε from our target, p). And now that we posed a realistic requirement, we can ask how large m should be in order to satisfy it. ■

The above example demonstrate the general type of questions we will use probability inequalities for: Given an accuracy parameter $\varepsilon > 0$ and a confidence parameter $\delta \in (0,1)$, how many samples are needed to guarantee that with probability of at least $1-\delta$, our estimate is within an additive error of at most ε. That is, we aim at constructing an algorithm (a "learner") whose estimations are probably (with probability at least $1-\delta$) approximately (within additive error of at most ε) correct. We then say that given m samples or more, our learner's estimation will be *probably approximately correct (PAC)*. The theoretical framework of PAC will be discussed in details in **??**.

1.3.4.1 Markov's and Chebyshev's inequalities

We now recall Markov's and Chebyshev's inequalities and then apply both inequalities to derive concentration inequalities for averages.

Theorem 1.3.6 Markov's inequality: Let X be a nonnegative random variable (i.e. $Im(X) \subseteq \mathbb{R}_{\geq 0}$) and deonte the expectation value of X by $\mathbb{E}[X]$. For any $a > 0$:

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

Proof. Let $f(x)$ be the density function of x. Since X is non-negative so $f(x) = 0$ for $x < 0$. Thus,

$$\frac{\mathbb{E}[X]}{a} = \frac{1}{a}\int_0^\infty f(x)x\,dx \geq \frac{1}{a}\int_{x=a}^\infty f(x)x\,dx \geq \frac{1}{a}\int_{x=a}^\infty f(x)a\,dx = \mathbb{P}(X \geq a)$$

■

Corollary 1.3.7 Let $X_1,\ldots,X_m$ be m i.i.d. non-negative random variables and denote the expectation value of each by $\mathbb{E}[X_i] = \mathbb{E}[X]$ $i = 1,\ldots,m$. Denote also $\overline{X} = \frac{1}{m}\sum_{i=1}^m X_i$. Then for any

$a > 0$:

$$\mathbb{P}\left[\overline{X} \geq a\right] \leq \frac{\mathbb{E}[X]}{a}$$

Proof. As $\overline{X}$ is a non-negative random variable let us apply Markov's Inequality. So: $\mathbb{P}\left(\overline{X} \geq a\right) \leq \mathbb{E}\left[\overline{X}\right]/a$. Now

$$\mathbb{E}\left[\overline{X}\right] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} X_i\right] = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[X_i] = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[X] = \mathbb{E}[X]$$

and therefore, we obtain the desired bound. ■

Markov's inequality gives a bound in terms of the expectation value $\mathbb{E}[X]$, but it can be used to obtain also a bound in terms of the variance of X. This bound is called Chebyshev's inequality and does not require X to be non-negative.

Theorem 1.3.8 **Chebyshev's inequality:** For a random variable X with finite mean $\mathbb{E}[X]$ and a variance $Var(X)$ and for every $a > 0$

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq a] \leq \frac{Var(X)}{a^2}$$

Proof. Consider the random variable $Y = (X - \mathbb{E}[X])^2$. This is a non-negative random variable and as such we can apply Markov's inequality over it. We obtain that

$$\mathbb{P}\left[(X - \mathbb{E}[X])^2 \geq a^2\right] \leq \frac{Var(X)}{a^2}$$

To finish the proof, simply observe that $\mathbb{P}[|X - \mathbb{E}[X]| \geq a] = \mathbb{P}\left[(X - \mathbb{E}[X])^2 \geq a^2\right]$. ■

Unlike in the case of the expectation value, the variance of the average of i.i.d. random variables is not equal to the variance of the original random variable. For example, for m i.i.d. random variables, $X_1, \ldots, X_m$, with a variance $Var(X_i) = Var(X)$, $i = 1..m$, we have

$$V\left[\frac{1}{m}\sum_{i=1}^{m} X_i\right] = \frac{1}{m^2} V\left[\sum_{i=1}^{m} X_i\right] = \frac{1}{m^2}\sum_{i=1}^{m} Var(X_i) = \frac{1}{m^2}\sum_{i=1}^{m} Var(X) = \frac{1}{m} Var(X)\ .$$

where we used the fact the $Var[X_1 + X_2] = Var(X_1) + Var(X_2)$ for independent X_1 and X_2. In particular, the variance of the average tends to zero when m tends to infinity. This leads to a much better concentration inequality.

Corollary 1.3.9 Let $X_1, \ldots, X_m$ be m i.i.d random variables with a finite variance $Var(X)$. Denoting $\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$. For any $a > 0$, it holds that

$$\mathbb{P}\left[\left|\overline{X} - \mathbb{E}\left[\overline{X}\right]\right| \geq a\right] = \mathbb{P}\left[\left|\overline{X} - \mathbb{E}[X]\right| \geq a\right] \leq \frac{Var(X)}{m \cdot a^2}$$

For a positive integer k, the k-th *moment* of a random variable X is defined by $\mathbb{E}[X^k]$. As we just saw, Markov's inequality exploits only information about the first moment, while Chebyshev's inequality uses both the first and the second moments. Comparing the bounds in 1.3.7 and 1.3.9, we see that using both the first and the second moments, we obtain better concentration inequalities than using only the first. We shall see soon that more generally, the more moments we use, the better the bounds we get.

1.3.4.2 Coin Prediction Example

Let us consider the problem of estimating the bias of a coin, or in short 'coin prediction'. This will allow us to introduce the important concept of the *sample complexity*, to demonstrate the usefulness and limitations of the Markov's and Chebyshev's inequalities as well as a new one called Hoeffding's inequality.

Formally, a coin flip is a Bernoulli random variable Z which takes the value 1 for "Heads" or 0 for "tails". We shall denote the probability distribution of Z by $\mathcal{D}_p$, where $0 \leq p \leq 1$ and $\mathcal{D}_p(1) = p, \mathcal{D}_p(0) = 1-p$ are the probabilities to obtain 1 and 0 correspondingly. Let this be a fair coin where $p = 1/2$. Let the following be a sequence of m tosses of this coin $Z_1, \ldots, Z_m \overset{iid}{\sim} Ber(p)$ and denote the results of the tosses (that is, the samples) by $S = (z_1, \ldots, z_m)$. Denote the probability of obtaining a specific S by $\mathcal{D}_p^m(S)$ (or simply by $\mathcal{D}^m(S)$).

A coin prediction *learning algorithm*, $\mathcal{A}$, is a procedure which takes as an input a sequence S, drawn according to $\mathcal{D}_p^m$, and produces as an output an estimation of p. This estimation, also called "prediction", is denoted by $\mathcal{A}(S)$ or $\widehat{p}(S)$ or simply $\widehat{p}$. Since S is finite, we do not expect our estimation $\widehat{p}$ to be exact. Instead, we will settle for an algorithm that yields a $\widehat{p}$ which satisfies $|\widehat{p} - p|| \leq \varepsilon$, for some $0 < \varepsilon < 1$ which is called the *accuracy* parameter. Even then, there is always (unless p equals 1 or 0) *some* chance that the drawn sequence would be highly non-representative. For example, it can come out all 0's in spite of p being close to 1. So it is impossible to obtain a guarantee that $|\widehat{p} - p| \leq \varepsilon$ holds with absolute certainty.

Hence, we introduce a *confidence* parameter $\delta \in (0,1)$, and require that the event $\{S : |\widehat{p} - p| > \varepsilon\}$ occurs with a probability of at most δ. In other words, we require our algorithm to be such that the probability of flipping the coin m times and obtaining a sequence S that causes it to produce an inaccurate estimation, $\widehat{p}(S)$ ('inaccurate' meaning $|\widehat{p} - p| > \varepsilon$) is smaller or equal to δ.

Intuitively, the larger the number of flips, the more information we have about the coin and the better chance we have to satisfy the accuracy and confidence requirements. Since the accuracy and confidence parameters ε and δ are fixed, there should be some finite number of flips $m_{\mathcal{A}}$ (which depends on ε, δ and $\mathcal{A}$) such that for any sample of size $m \geq m_{\mathcal{A}}$ our algorithm satisfies the above accuracy and confidence requirements. If such $m_{\mathcal{A}}$ exists, we say that our algorithm is a *learning algorithm* for the task of coin prediction. Mathematically speaking, we are looking for an algorithm that satisfies the following definition:

Definition 1.3.21 Let $\mathcal{A}$ be an algorithm, which will be also denoted as $\widehat{p}(S)$ that given a set of coin tosses samples $S \in \{0,1\}^m$ returns $\widehat{p} \in [0,1]$ and satisfies the following conditions:

- For any $\varepsilon, \delta \in (0,1)$, there exists a non-negative integer $m_{\mathcal{A}}(\varepsilon, \delta)$ such that, if a sequence S of m numbers, where $m \geq m_{\mathcal{A}}$, is generated according to $\mathcal{D}_p^m$, then, **for any** $0 \leq p \leq 1$:

  $$\mathcal{D}_p^m\left[|\widehat{p}(S) - p| > \varepsilon\right] \leq \delta$$

  Namely, the probability to get a sample S such that the algorithm's output $\widehat{p}(S)$ will not be in the interval $[p \pm \varepsilon]$ is less or equals to δ.

- If a sequence S of m numbers, where $m < m_{\mathcal{A}}$, is generated, **there exists a** p, with $0 \leq p \leq 1$, such that:

  $$\mathcal{D}_p^m\left[|\widehat{p}(S) - p| > \varepsilon\right] > \delta$$

The function $m_{\mathcal{A}}(\varepsilon, \delta)$: $[0,1] \times [0,1] \to \mathbb{N}$ is called the *sample complexity* of the algorithm $\mathcal{A}$.

The first condition means that regardless to the true value of p, it is enough to draw $m_{\mathcal{A}}$ samples (i.e., to toss the coin $m_{\mathcal{A}}$ times) in order to know p, with a certainty of $1-\delta$ and an accuracy of $\pm\varepsilon$. The second conditions means that, at least for some values of p, drawing $m_{\mathcal{A}}(\varepsilon, \delta) - 1$ samples would *not* be enough for that.

Note that the confidence requirement must hold *independently of the procedure by which the coins are provided*. That is, independently of the way $\widehat{p}$ is chosen. It does not matter if, for example, the coin is randomly drawn from a pile of coins where most coins are approximately fair (most of their p's are close to $1/2$) or where most are faked in a specific way, (their p's are concentrated around, say, $\frac{1}{4}$), or where all p's are equally probable.

Choosing an Algorithm

Let us now look for an algorithm that satisfies the above definition. Given a sample $S = (z_1, \ldots, z_m)$ the most straightforward estimate of p is the empirical proportion of heads (ones), namely

$$\widehat{p}(S) = \frac{1}{m}\sum_{i=1}^{m} z_m$$

So we have a very simple algorithm for coin prediction, "Count the ones and divide by the number of tosses". Let us show that this algorithm satisfies the definition above.

We begin with noticing that this estimator, being the empirical mean of the parameter is an unbiased estimator of p. As such it can achieve, for the right sample S, that $|\widehat{p} - p| = 0$. Next, we will test the quality of $\widehat{p}$ as an estimator of p. In other words, how many coin flips we need to ensure that $\widehat{p}$ is, most probably, very close to its expectation value, p? To answer this question, we can use concentration inequalities.

Estimating Sample Complexity Using Markov's Inequality

For a direct application of Markov's inequality we take $|\widehat{p} - p|$ as our random variable. We then need to calculate $\frac{1}{\varepsilon}\mathbb{E}\left[|\widehat{p} - p|\right]$ and extract its dependence on the sample size m. By doing so we achieve the following bound:

$$\mathcal{D}_p^m\left[|\widehat{p} - p| \geq \varepsilon\right] \leq \frac{1}{\sqrt{4m\varepsilon^2}}$$

If we select m to be

$$m \geq \left\lceil \frac{1}{4\varepsilon^2} \cdot \frac{1}{\delta^2} \right\rceil$$

then the right-hand side is smaller or equals to δ. So, for any $\varepsilon, \delta \in (0,1)$, if we sample $m_{\mathcal{A}}(\varepsilon,\delta) \geq \left\lceil \frac{1}{4\varepsilon^2} \cdot \frac{1}{\delta^2} \right\rceil$ samples then this learning algorithm achieves that

$$\mathcal{D}_p^m \left[|\widehat{p}(S) - p| > \varepsilon \right] > \delta$$

Estimating Sample Complexity Using Chebyshev's Inequality

To improve the upper bound seen above (i.e find a sample complexity function that will need less samples), let us use Chebyshev's inequality. As the variance of a Bernoulli random variable is $p(1-p) \leq 1/4$, when applying corollary 1.3.9 we get:

$$\mathcal{D}_p^m \left[|\widehat{p} - p| \geq \varepsilon \right] = \mathcal{D}_p^m \left[|\widehat{p} - \mathbb{E}[\widehat{p}]| \geq \varepsilon \right] \leq \frac{p(1-p)}{m\varepsilon^2} \leq \frac{1}{4m\varepsilon^2}$$

We see that the bound obtained using Chebyshev's inequality tends to zero as $\frac{1}{m}$ while the one obtained from Markov's inequality tends to zero as $\frac{1}{\sqrt{m}}$. This fact enables us to obtain a better bound for the sample complexity:

Corollary 1.3.10 The sample complexity of coin prediction is bounded above by $m(\varepsilon,\delta) \leq \left\lceil \frac{1}{4\varepsilon^2} \cdot \frac{1}{\delta} \right\rceil$.

Estimating Sample Complexity Using Hoeffding's Inequality

A natural question which arises is whether the obtained bound is optimal (tight). Indeed, we can further improve the bound by exploiting the fact that our random variable not only has a finite variance, but it is also bounded between 0 and 1. For this end we use Hoeffding's inequality for the average of independent and bounded random variables.

Theorem 1.3.11 — Hoeffding's inequality. Let $X_1, \ldots X_m$ be independent and bounded random variables with $a_i \leq X_i \leq b_i$. Let $\overline{X} = \frac{1}{m}\sum_{i=1}^m X_i$. Then,

$$\mathbb{P}\left[\left| \overline{X} - \mathbb{E}\left[\overline{X}\right] \right| \geq \varepsilon \right] \leq 2\exp\left(\frac{-2m^2\varepsilon^2}{\sum_{i=1}^m (b_i - a_i)^2} \right)$$

Corollary 1.3.12 Let $X_1, \ldots X_m$ be a sequence of m i.i.d random variables, each with an expectation value $\mathbb{E}[X]$ and all of which are bounded: $a \leq X_i \leq b$. Denote $\overline{X} = \frac{1}{m}\sum_{i=1}^m X_i$ then:

$$\mathbb{P}\left[\left| \overline{X} - \mathbb{E}[X] \right| \geq \varepsilon \right] \leq 2\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2} \right)$$

Proof. To conclude the corollary from 1.3.11 notice that as $X_1, \ldots, X_m$ all share the same expectation and bounds the:

$$\mathbb{E}\left[\overline{X}\right] = \frac{1}{m}\sum_{i=1}^m \mathbb{E}[X] = \mathbb{E}[X], \quad \sum_{i=1}^m (b_i - a_i)^2 = m \cdot (b-a)^2$$

By placing these expressions in 1.3.11 we conclude the inequality. ■

By applying Hoeffding's inequality to the case of coin prediction problem, then for a sample of size m, we obtain that:

$$\mathcal{D}_p^m \left[|\widehat{p} - p| \geq \varepsilon \right] \leq 2 \exp\left(-2m\varepsilon^2\right)$$

Therefore, using Hoeffding's Inequality we are able to get a bound which converges exponentially in m. By taking $m \geq \left\lceil \frac{1}{2\varepsilon^2} \cdot \log(\frac{2}{\delta}) \right\rceil$ samples we obtain that this probability is bound above by δ as required and conclude that:

Corollary 1.3.13 The coin prediction algorithm, $\widehat{p}(S)$, which estimates p by the number of ones (heads) divided by the number of coin flips, is a learning algorithm satisfying definition **??** with a sample complexity which is bounded above by $m_{\mathcal{A}}(\varepsilon, \delta) \leq \left\lceil \frac{1}{2\varepsilon^2} \cdot \log(\frac{2}{\delta}) \right\rceil$.

2. Linear Regression

2.1 Regression Models

Imagine that we work for an online store and would like to predict the "customer lifetime value", that is, the total future net revenue that an online customer will provide to the store. In order to do so, we choose different customer properties that we think might be relevant such as age, income, total spending in the website, average monthly visits, etc. The vector space defined over all possible values of these properties (commonly called features) is our **sample domain**. We denote it by $\mathcal{X}$. In the case of the online store $\mathcal{X} := \mathbb{R}^d$, for d the number of features. In addition, we also define the **response set** $\mathcal{Y}$ which in this case represents the customers' lifetime value. So in our case, $\mathcal{Y} = \mathbb{R}$.

Next, we collect these all these details for m customers. Each customer is represented as a *sample* – a pair $(\mathbf{x}, y)$ where $\mathbf{x} \in \mathcal{X}$ is the column vector of features of the sample (also called data point) and the corresponding response $y \in \mathcal{Y}$. This is our *training dataset*. We would like to use our training dataset to find a way to **estimate** or **predict** the lifetime values of any new customer using solely their feature vector. This setup is sometimes called **Batch Supervised Learning**. In order to do this, we will build a *regression model*.

A regression model is a way to represent a functional relation between a set of explanatory variables (features) in $\mathcal{X}$ and a scalar response, also referred to as dependent variable, in $\mathcal{Y}$. So, we will **assume** that there exists some function $f : \mathcal{X} \to \mathcal{Y}$ that captures this relation for each sample $x \in \mathcal{X}$ and its response $y \in \mathcal{Y}$. This function f is unknown to us and we would like to find it. It may be deterministic or it may contain a random component.

Let's assume first that the relation between $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ is *deterministic*. So we assume that there exists a function $f : \mathcal{X} \to \mathcal{Y}$ such that, each sample we observe, now or in the future, is of

the form $(\mathbf{x}, y)$ with $y = f(\mathbf{x})$. In particular for our training set $y_i = f(\mathbf{x}_i)$ for every training sample $i = 1 \ldots m$. Our goal is to *learn* f from a training sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, so we can estimate or predict the value $f(\mathbf{x})$ for a new value $\mathbf{x}$. A sample we haven't seen in our training set – a new sample – is sometimes called a *test* sample. Using the training sample S we will create a function that we hope is as similar as possible to the unknown function f. The function we create is called a **prediction rule** and we denote it by $\hat{f}$ or h_S. (The notation h_S emphasizes that our prediction rule depends on the training sample S).

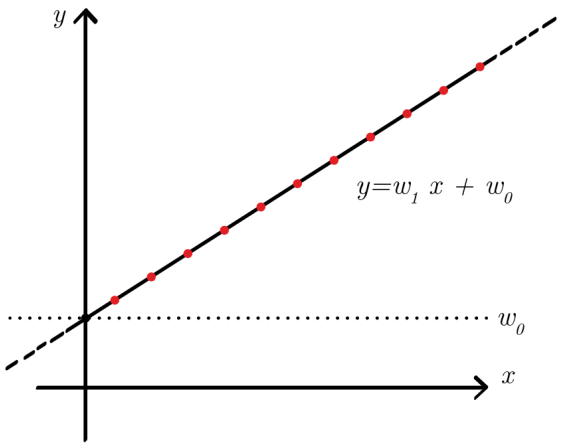


Figure 2.1: *Illustration of a regression model with $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \mathbb{R}$. Red dots are samples. The solid curve is the learned prediction rule $\hat{f}$.*

For reasons we discuss later (**??**), whenever we try to model a functional relation f, we restrict ourselves to a specific family of functions. Such a family is referred to as a *hypothesis class*. We decide on the hypothesis class before looking at the data, and the prediction rule we find must be in the chosen hypothesis class.

While we can build regression models over various domains $\mathcal{X}$, the simplest domain to consider is the Euclidean space $\mathbb{R}^d$ where each point x is a feature vector with d real numbers. In this chapter and in most of this book, we consider $\mathcal{X} := \mathbb{R}^d$.

2.1.1 Linear Regression

Let us assume that the relation $\mathcal{X} \to \mathcal{Y}$ is *linear*. This is perhaps the simplest relation we can describe. Formally, we define the *linear model*, or the *linear hypothesis class*, as the set of linear functions from the domain set to the response set:

$$\mathcal{H}_{reg} := \left\{ h(x_1, \ldots, x_d) = w_0 + \sum_{i=1}^{d} x_i w_i \;\middle|\; w_0, w_1, \ldots, w_d \in \mathbb{R} \right\}. \tag{2.1}$$

In statistics, learning f from a training sample is known as *linear regression*[1]. Each function $h \in \mathcal{H}_{reg}$ is characterized by the **weights** (also known as regression coefficients) $w_1, \ldots, w_d$ representing the d

[1] The name "regression" refers to a statistical phenomenon known as "regression to the mean".

features and an **intercept** w_0. To simplify the notation, for a given sample $\mathbf{x} = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$ we add a zero-th coordinate with the value of 1, and define $\mathbf{x} = (1, x_1, \ldots, x_d)^\top \in \mathbb{R}^{d+1}$. Using this notation each function in the linear hypothesis class can be written in the form $h(\mathbf{x}) := \langle \mathbf{x}, \mathbf{w} \rangle = \mathbf{x}^\top \mathbf{w}$. For the remainder of this chapter, we will assume that the intercept is already incorporated into the weights vectors, so we can define linear hypothesis class equivalently as

$$\mathcal{H}_{reg} := \left\{ h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} \mid \mathbf{w} \in \mathbb{R}^{d+1} \right\} \tag{2.2}$$

Note that by convention, the first coordinate of $\mathbf{w}$ is the intercept w_0.

So, given a training set S, we are looking for a vector $\mathbf{w} \in \mathbb{R}^{d+1}$ such that $y_i = \mathbf{x}_i^\top \mathbf{w}$ for all $i \in [m]$. This setup should be familiar from 1.1. However, as we will see, we may not be able to find a vector $\mathbf{w}$ for which all these equalities hold exactly.

The regression matrix

Let us arrange the training data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ in matrix form. We define the *response vector* as the column vector $\mathbf{y} \in \mathbb{R}^m$ and the *regression matrix* (or *design matrix*) $\mathbf{X} \in \mathbb{R}^{m \times (d+1)}$ as follows.

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \vdots & \\ — & \mathbf{x}_m & — \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Note that m rows of $\mathbf{X}$ represent our m training samples and the $d+1$ columns of $\mathbf{X}$ represent the intercept and d features. In this notation, we are looking for a vector $\mathbf{w} \in \mathbb{R}^{d+1}$ that satisfies a system of m linear equations in the variable $\mathbf{w}$,

$$\mathbf{X}\mathbf{w} = \mathbf{y} \tag{2.3}$$

We must have enough training samples

At this point, we will assume that $m \geq d+1$, namely, that we have enough training samples so that the linear system (2.3) is not under-determined. In practical terms, this means that we have at least as many training samples as we have features. In our online store example, this means that we must collect data on $m \geq d+1$ customers before we start training our regression model, where d is the number of features we collect on each customer (e.g. age, income, total spending, number of monthly visits to the website, etc).

2.1.2 Designing A Learning Algorithm

2.1.2.1 Realizablity

Recall that to derive the problem of finding $\mathbf{w} \in \mathbb{R}^{d+1}$ that satisfies (2.3) we have restricted ourselves to describing functional relations $\mathcal{X} \xrightarrow{f} \mathcal{Y}$ such that $f \in \mathcal{H}_{reg}$. The case where there exists a solution for (2.3) is called the **Realizable** case. Let $\hat{\mathbf{w}}$ be a solution for (2.3), then the prediction rule we choose is $\hat{f} = (\mathbf{x})\mathbf{x}^\top \hat{\mathbf{w}}$.

The case where there is no $f \in \mathcal{H}_{reg}$ that satisfies the system of equations (i.e there is no solution for the system) is called the **Non-Realizable** case. In this case, since we decided to choose a

prediction rule in $\mathcal{H}_{reg}$, we must settle for finding $\hat{f} \in \mathcal{H}_{reg}$ which is *"most fitting"* for our purposes.

Our learning algorithm for linear regression must address both the realizable and non-realizable cases. In the realizable case, to find the rule f, all we need to do is solve the linear system (2.3) for $\mathbf{w}$. But what will we do in the non-realizable case, where $f \notin \mathcal{H}_{reg}$? How should we choose the prediction rule $\hat{f}$?

2.1.2.2 Loss Function

One way to choose $\hat{f} \in \mathcal{H}_{reg}$ in the non-realizable case is to assign each $f \in \mathcal{H}_{reg}$ with some measure of quality, and choose the "best" f. The function defined to measure the quality is called a **loss function** and it measures the quality of the hypothesis by comparing between the true- and predicted values:

$$\sum_{i=1}^{m} L\left(f\left(\mathbf{x}_i\right), \hat{f}\left(\mathbf{x}_i\right)\right), \quad i = 1, \ldots, m$$

We will then pick the prediction rule which is "best fitting"/"most likely" given our training data and with respect to the loss function we chose. Two commonly used loss functions for regression problems are the **Absolute Value Loss**

$$L\left(y, \hat{f}\left(\mathbf{x}\right)\right) := \left|y - \hat{f}\left(\mathbf{x}\right)\right| \tag{2.4}$$

or the **Squared Loss**

$$L\left(y, \hat{f}\left(\mathbf{x}\right)\right) := \left(y - \hat{f}\left(\mathbf{x}\right)\right)^2 \tag{2.5}$$

We will focus on the linear regression setup when using the square loss function.

2.1.2.3 Empirical Risk Minimization

As we are concerned for the performance of a prediction rule $\hat{f}$ on a new data point $\mathbf{x}$ by $\left(\hat{f}\left(\mathbf{x}\right) - y\right)^2$, it makes sense to choose $\hat{f}$ that minimizes that same loss L *on the training data we already have*. This strategy for choosing $\hat{f}$ is known as **Empirical Risk Minimization**. For a given prediction rule $\hat{f} \in \mathcal{H}$, the quantity

$$\sum_{i=1}^{m} L\left(y_i, \hat{f}\left(\mathbf{x}_i\right)\right)$$

where $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ is our training data, is called the **empirical risk**. In the case of the square loss, the empirical risk of the linear function $\hat{f}\left(\mathbf{x}_i\right) = \mathbf{x}_i^\top \mathbf{w}$ is given by:

$$\sum_{i=1}^{m} \left(y_i - \mathbf{x}_i^\top \mathbf{w}\right)^2 = ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 = \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)^\top \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right) \tag{2.6}$$

2.1.2.4 Least Squares

Minimizing the empirical risk of (2.6) means minimizing the sum of squares of the deviations of the responses from a linear function. In other words, we choose the linear function in $\mathcal{H}_{reg}$ that is closest to the responses in terms of the squared error distance. The deviation $y_i - \mathbf{x}_i^\top \mathbf{w}$ is called the i-th **residual** and the total empirical risk in our case is called **Residual Sum of Squares** (or **RSS**):

$$RSS_{\mathbf{X},\mathbf{y}}\left(\mathbf{w}\right) := ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2$$

To simplify notation we often write $RSS(\mathbf{w})$ keeping the dependence on $\mathbf{X}, \mathbf{y}$ implicit. So to learn the linear function by empirical Risk minimization we want to find

$$\underset{\mathbf{w}\in\mathbb{R}^d}{argmin}\, RSS(\mathbf{w}) = \underset{\mathbf{w}\in\mathbb{R}^d}{argmin}\, ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 \tag{2.7}$$

It is important to notice that the optimization problem (2.7) addresses both the realizable and non-realizable cases:

- In the realizable case, as $\mathbf{y} \in Im(\mathbf{X})$ we know there exists at least one solution $\widehat{\mathbf{w}}$ such that $\mathbf{X}\widehat{\mathbf{w}} = \mathbf{y}$. Such a solution will achieve a value of zero. As the RSS function is bounded below by zero, such a solution is therefore a minimizer of the RSS.
- In the non-realizable case, as $\mathbf{y} \notin Im(\mathbf{X})$ there is no solution $\widehat{\mathbf{w}}$ such that $\mathbf{X}\widehat{\mathbf{w}} = \mathbf{y}$. Therefore, no vector $\widehat{\mathbf{w}}$ will achieve a value of zero for the RSS objective. Instead, we decide to find a vector that is "good enough" in the sense of minimizing the squared loss.

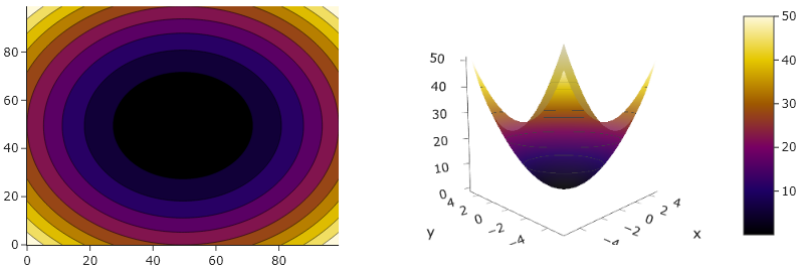


Figure 2.2: *Illustration of the RSS function over $\mathbb{R}^2$ for $\mathbf{X}$ of full rank. Chapter 2 Examples - Source Code*

A *necessary* condition for $\mathbf{w}$ to be a minimizer of the function $||\mathbf{y} - \mathbf{X}\mathbf{w}||^2$ is that all its partial derivative vanish at $\mathbf{w}$. Recalling the definition of the inner product, this condition can be written as:

$$\frac{\partial}{\partial w_j} RSS(\mathbf{w}) = -2\sum_{i=1}^{m} (\mathbf{x}_i)_j \cdot (y_i - \mathbf{x}_i\mathbf{w}) = 0 \tag{2.8}$$

for all $j = 0, \ldots, d$, where $(\mathbf{x}_i)_j$ is the j-th entry of $\mathbf{x}_i$. It is the $x_{j,i}$ element of the matrix $\mathbf{X}$. Notice that this constructs a system of $d+1$ linear equations in $\mathbf{w}$. We can organize (2.8) as such to get the form below. Recall that we have already derived this function in **??**.

$$\nabla RSS(\mathbf{w}) = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \tag{2.9}$$

2.1.2.5 The Normal Equations

So a minimizer of (2.7) must also be a solution for the following linear system, known as the **Normal Equations**:

$$\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \quad \Longleftrightarrow \quad \mathbf{X}^\top\mathbf{y} = \mathbf{X}^\top\mathbf{X}\mathbf{w} \tag{2.10}$$

Geometric Interpretation

Let us derive a geometric interpretation of linear regression and gain a better understanding what the solution to (2.10) might be like. We usually think of $\mathbf{X} \in \mathbb{R}^{m\times(d+1)}$ as a matrix that consists of m rows, one for each training sample. Instead, we can equivalently think of $\mathbf{X}$ as a matrix that consists of $d+1$ columns, one for each feature (and the intercept). Define

$$\mathbf{X} \coloneqq \begin{bmatrix} | & & | \\ \varphi_0 & \cdots & \varphi_d \\ | & & | \end{bmatrix}$$

and recall that the vector space spanned by the columns of $\mathbf{X}$ is:

$$span\,(\varphi_0,\ldots,\varphi_d) = Im\,(\mathbf{X}) \subset \mathbb{R}^m$$

Since we assume $m \geq d+1$, $Im\,(\mathbf{X})$ is a linear subspace of $\mathbb{R}^m$. If we have many more samples than features, $m \gg d+1$, then $Im\,(\mathbf{X})$ is just a small subspace of $\mathbb{R}^m$. If we have the minimal number of samples possible, $m = d+1$, and the vectors $\varphi_0,\ldots,\varphi_d$ form an independent set, then the subspace fills the entire space: $Im\,(\mathbf{X}) = \mathbb{R}^m$.

Now, consider the response vector $\mathbf{y} \in \mathbb{R}^m$. Note that it may or may not belong to the subspace $Im\,(\mathbf{X})$:

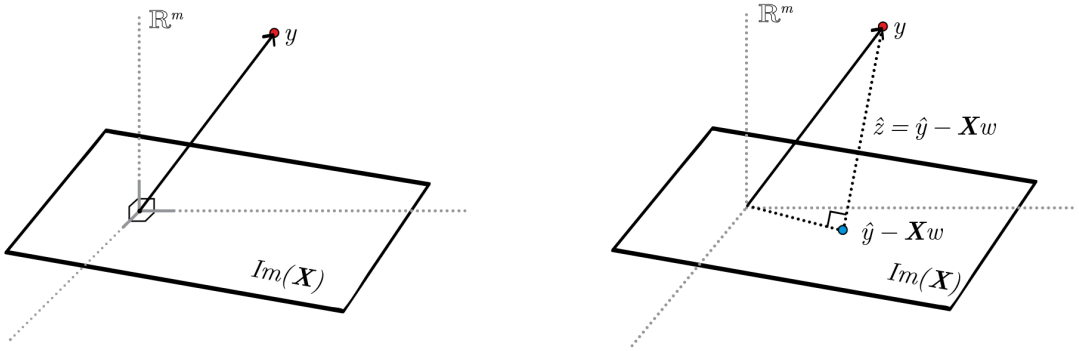
- If $\mathbf{y} \in Im\,(\mathbf{X})$ then by definition $\mathbf{y}$ is a linear combination of $\varphi_0,\ldots,\varphi_d$ and there exists a vector $\mathbf{w} \in \mathbb{R}^{d+1}$ such that $\mathbf{X}\mathbf{w} = \mathbf{y}$. This is the realizable case. We can now differentiate between two sub-cases:
  - If $\varphi_0,\ldots,\varphi_d$ are linearly independent, then $\mathbf{y}$ can be expressed as a *unique* linear combination of the columns of $\mathbf{X}$. In this case the linear system (2.10) has a unique solution.
  - If however $\varphi_0,\ldots,\varphi_d$ are in fact linearly dependent, then there are infinitely many ways to express $\mathbf{y}$ as a linear combination of the columns of $\mathbf{X}$. Any one of these ways is a valid solution for (2.10).
- If $\mathbf{y} \notin Im\,(\mathbf{X})$ then $\mathbf{y}$ is not a linear combination of $\varphi_0,\ldots,\varphi_d$. As such there is no vector $\mathbf{w} \in \mathbb{R}^{d+1}$ that satisfies $\mathbf{X}\mathbf{w} = \mathbf{y}$. This is the non-realizable case. In this case we decided to choose the vector $\mathbf{w}$ for which $RSS(\mathbf{w}) = ||\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}||$ is minimal.

Now we are able to understand what is "normal" about the normal equations (2.10). Observe that the equations (2.8), from which we have derived the normal equations, can be equivalently written as

$$\left\langle \varphi_j, \mathbf{y} - \mathbf{X}\mathbf{w} \right\rangle = 0 \quad j = 0,\ldots,d \tag{2.11}$$

We conclude that $\mathbf{w}$ is a solution to the normal equations if and only if $\mathbf{y} - \mathbf{X}\mathbf{w}$ is perpendicular to $\varphi_0,\ldots,\varphi_d$. Since these vectors span the subspace $Im\,(\mathbf{X})$, another way to write this is $\mathbf{y} - \mathbf{X}\mathbf{w} \in Im\,(\mathbf{X})^{\perp}$.

Let $\hat{\mathbf{w}}$ be a solution to the normal equations and define $\hat{\mathbf{y}} \coloneqq \mathbf{X}\hat{\mathbf{w}}$. Note that $\hat{\mathbf{y}}$, the vector where the i-th entry is the prediction on the i-th training sample $\mathbf{x}_i$, is $\hat{\mathbf{y}} \in Im\,(\mathbf{X})$. In this notation, when solving



(a) *In the non-realizable case, the response vector* $\mathbf{y}$ *lies outside* $Im(\mathbf{X})$*, the subspace spanned by the columns of* $\mathbf{X}$*. In this case there is no solution for the system* $\mathbf{Xw} = \mathbf{y}$*.*

(b) *If* $\hat{\mathbf{w}}$ *is a solution to the normal equations, then* $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$ *is an orthogonal projection of the response vector* $\mathbf{y}$ *onto* $Im(\mathbf{X})$*. The difference* $\hat{\mathbf{z}} = \mathbf{y} - \hat{\mathbf{y}}$ *is therefore perpendicular (normal) to* $Im(\mathbf{X})$*.*

Figure 2.3: Geometric interpretation of linear regression

the normal equations, namely when seeking to minimize the RSS, we minimize $||\mathbf{y} - \hat{\mathbf{y}}||$. Define the **residual vector** $\hat{\mathbf{z}} := \mathbf{y} - \hat{\mathbf{y}}$. Note that from (2.11) we get that $\hat{\mathbf{z}} \in Im(\mathbf{X})^{\perp}$. In other words, if $\hat{w}$ is a solution to the normal equations then $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$ is the **orthogonal projection** of $\mathbf{y}$ on $Im(\mathbf{X})$ and $\hat{\mathbf{z}} = \mathbf{y} - \hat{\mathbf{y}}$ is a *normal* (a perpendicular vector) to $Im(\mathbf{X})$. Hence the name the "normal equations".

Solving The Normal Equations

As we have seen, from a geometric perspective, if $m \geq d+1$, solving the normal equations means finding a vector $\hat{\mathbf{w}}$ such that $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$ is the orthogonal projection of $\mathbf{y}$ on $Im(\mathbf{X})$. We can deduce from this two important facts about the existence and uniqueness of a solution to the normal equations:

- **Existence:** As a linear system, the normal equations can have either (i) no solutions, (ii) a unique solution, or (iii) an infinite number of solutions that constitute an affine subspace. From the geometric interpretation we see that (i) is impossible. Indeed it can be shown that the normal equations must have at least one solution, so that they have a unique solution or an infinite number of solutions.
- **Uniqueness:**
  - If the columns of $\mathbf{X}$ form a linearly independent set (equivalently, if $dim(Ker(\mathbf{X})) = 0$) then the projection $\hat{\mathbf{y}}$ can be described uniquely as a linear combination of the columns $\varphi_0, \ldots, \varphi_d$, namely, there exists a unique $\hat{\mathbf{w}}$ such that $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$. This vector of coefficients $\hat{w}$ is a unique solution to the normal equations.
  - If the columns of $\mathbf{X}$ contain linear dependencies (equivalently, if $dim(Ker(\mathbf{X})) > 0$) then the projection $\hat{\mathbf{y}}$ can be described as infinitely many linear combinations of the columns $\varphi_0, \ldots, \varphi_d$. Any such linear combination will suffice and we simply need to find *one* vector $\hat{w}$ such that $\hat{y} = \mathbf{X}\hat{\mathbf{w}}$.

Case 1: Linearly Independent Feature Vectors: $dim(Ker(\mathbf{X})) = 0$

It can be shown that for any matrix A then $Ker(A) = Ker(A^{\top}A)$. Since we assume that $Ker(\mathbf{X})$ is trivial, we know that the square symmetric matrix $\mathbf{X}^{\top}\mathbf{X}$ has a trivial kernel, namely, is invertible. Ex.1

This means that $\mathbf{w}$ satisfies $\mathbf{X}^\top\mathbf{y} = \mathbf{X}^\top\mathbf{X}\mathbf{w}$ if and only if $\mathbf{w} = \left[\mathbf{X}^\top\mathbf{X}\right]^{-1}\mathbf{X}^\top\mathbf{y}$. So in this case the unique solution to the normal equations is

$$\hat{\mathbf{w}} := \left[\mathbf{X}^\top\mathbf{X}\right]^{-1}\mathbf{X}^\top\mathbf{y}$$

■ **Example 2.1** Let us find the estimator $\widehat{\mathbf{w}}$ for the following scenario. Suppose we are interested in estimating the running times in a 100 meters long race, besed on an athlete's height and weight. We gathered the details of the 4 top ranking athletes in the 2016 Rio Olympics:

Athlete	Weight (kg)	Height (cm)	Running Time (sec)
Usain Bolt	94	195	9.81
Justin Gatlin	79	185	9.89
Andre de Grasse	70	176	9.91
Yohan Blake	80	180	9.93

So the features are the *weight, height* and the response is *running time*. To fit a linear regression model to the data we begin with arranging it in a matrix and adding the intercept:

$$\mathbf{X} := \begin{bmatrix} 1 & 94 & 195 \\ 1 & 79 & 185 \\ 1 & 70 & 176 \\ 1 & 80 & 180 \end{bmatrix}, \quad \mathbf{y} := \begin{bmatrix} 9.81 \\ 9.89 \\ 9.91 \\ 9.93 \end{bmatrix}$$

As we have proven above, the estimator is given by the closed form of $\widehat{\mathbf{w}} := \left[\mathbf{X}^\top\mathbf{X}\right]^{-1}\mathbf{X}^\top\mathbf{y}$. Over given data we obtain that $\widehat{\mathbf{w}} \approx (11.38, 0.003, -0.009)^\top$ (up to rounding up numbers).

Next, let us use this estimator to estimate the running times of a new sample $\mathbf{x} = (1, 74, 176)^\top$:

$$\widehat{y} = \mathbf{x}^\top\widehat{\mathbf{w}} = \left\langle \begin{bmatrix} 1 \\ 74 \\ 176 \end{bmatrix}, \begin{bmatrix} 11.38 \\ 0.003 \\ -0.009 \end{bmatrix} \right\rangle = 10.018$$

■

Case 2: Linearly Dependent Feature Vectors: $dim(Ker(\mathbf{X})) > 0$

The columns of $\mathbf{X}$ are linearly dependent and therefore there are infinitely many was to express the projection $\hat{\mathbf{y}}$ as a linear combination of the columns of $\mathbf{X}$. Since we need some way, it would be convenient if we could find a solution $\hat{w}$ that is close to the origin in $\mathbb{R}^{d+1}$ (rather than a solution with very large norm, say). An excellent way to do this uses the SVD of $\mathbf{X}$.

Definition 2.1.1 Let $\mathbf{X} \in \mathbb{R}^{m\times d+1}$ and let $\mathbf{X} - U\Sigma V^\top$ be its SVD. The **Moore-Penrose pseudoin-**

verse of $\mathbf{X}$ is $\mathbf{X}^\dagger = V\Sigma^\dagger U^\top$ where $\Sigma^\dagger$ is a $d \times m$ diagonal matrix defined by:

$$\Sigma^\dagger_{i,i} = \begin{cases} 1/\Sigma_{i,i} & \Sigma_{i,i} \neq 0 \\ 0 & \Sigma_{i,i} = 0 \end{cases}$$

This is a generalization of the inverse matrix and indeed when the matrix $\mathbf{X}$ is invertible then then $\mathbf{X}^\dagger = \mathbf{X}^{-1}$. An important property of the pseudoinvers is that for a linear system of equations $A\mathbf{x} = \mathbf{b}$ with an infinite number of solutions, then $A^\dagger\mathbf{b}$ is a solution with minimal ℓ_2 norm, namely Ex.2

$$A^\dagger\mathbf{b} = \text{argmin}\left\{||\mathbf{x}||_2 \;\middle|\; A\mathbf{x} = \mathbf{b}\right\}$$

In our case, a solution to the normal equations with minimal ℓ_2 norm, and as such closest to the origin with respect to the Euclidean norm, is given by

$$\hat{\mathbf{w}} := X^\dagger\mathbf{y}$$

It can be shown that when dealing with a matrix of linearly independent columns ($dim\,(Ker\,(\mathbf{X})) = 0$) then the previously found solution $\hat{\mathbf{w}} = \left[\mathbf{X}^\top\mathbf{X}\right]^{-1}\mathbf{X}^\top\mathbf{y}$ equals to $X^\dagger\mathbf{y}$. We conclude that the formula $X^\dagger\mathbf{y}$ *always* gives us a solution to the normal equations: the unique solution if the solution is unique, and the solution with minimal ℓ_2 norm if not. Ex.3

It is left to show that the solution to the normal equations found above indeed minimize the RSS. We have derived the normal equations by asking that all partial derivatives of the RSS vanish. This means that a solution to the normal equations is an extremal point. Let us prove, for the case where $dim\,(Ker\,(\mathbf{X})) = 0$, that the found solution is indeed a minimum.

Claim 2.1.1 Assume $dim\,(Ker\,(\mathbf{X})) = 0$ and let $\hat{\mathbf{w}}$ satisfy $\mathbf{X}^\top\mathbf{y} = \mathbf{X}^\top\mathbf{X}\hat{\mathbf{w}}$. Then $\hat{\mathbf{w}}$ is a global minimizer of $RSS\,(\mathbf{w})$.

Proof. We know that $\hat{\mathbf{w}}$ is an extremal point of RSS. Now,

$$\frac{\partial^2 RSS_{\mathbf{X},\mathbf{y}}}{\partial w_k \partial w_l}\bigg|_{\hat{\mathbf{w}}} = -2\frac{\partial \sum_{i=1}^m \left(y_i - \sum_{j=1}^d x_{ij}\hat{w}_j\right)\mathbf{X}_{\cdot,k}}{\partial w_l} = 2\sum_{i=1}^m \mathbf{X}_{\cdot,k}\mathbf{X}_{\cdot,l} = 2\left[\mathbf{X}^\top\mathbf{X}\right]_{kl} \quad \forall k,l \in [d+1]$$

The matrix $\mathbf{X}^\top\mathbf{X}$ is a positive semi-definite matrix, and since we assumed $dim(Ker(\mathbf{X})) = 0$, it strictly positive definite. It follows that $RSS(\hat{\mathbf{w}})$ is a minimum. ■

Corollary 2.1.2 The vector $\widehat{\mathbf{w}} := \mathbf{X}^\dagger\mathbf{y}$ is always a solution to the normal equations and a minimizer of optimization problem (2.7), namely, of the residual sum of squares (RSS).

2.1.3 Numerical Considerations When Implementing

So far we have designed the learning algorithm. Now we want to *implement* it, namely, write efficient code that implements the algorithm that we have designed. The field of *numerical linear algebra* assists in addressing this challenge as the implementation of every machine learning algorithm is eventually reduced to performing linear algebra computations (e.g. matrix-vector or matrix-matrix products, matrix inverses and matrix decompositions). In the case of linear regression, as we have

seen, to write software that trains a linear regression model, we need to be able to calculate a matrix SVD.

In your basic linear algebra courses you worked with mathematical objects over real and complex vector spaces. Likely, you did not stop to wonder how to compute (say) the inverse of a matrix on a computer. This is not as simple as it may sound. Computers do not calculate over $\mathbb{R}$, they use bits and more specifically floating-point arithmetics with finite precision. There is an entire field in the intersection of mathematics and computer science, known as numerical linear algebra, that studies the accuracy and complexity of algorithms for computing linear algebraic quantities and matrix decompositions. As a machine learning expert, you must be as knowledgable as possible regarding the numerical implementation of your learning algorithms. You should care *deeply* about how your algorithms are implemented and when they break numerically.

Let's see a simple example for a numerical consideration in our case of linear regression. (We will discover that the SVD is even more useful than we thought.) Recall that if $dim(Ker(\mathbf{X})) = 0$, and equivalently if $\mathbf{X}^\top\mathbf{X}$ is not singular (invertible) then we have a simple formula for training our linear regression model. But what happens if $\mathbf{X}^\top\mathbf{X}$ is "almost singular"? Sometimes $\mathbf{X}^\top\mathbf{X}$ is formally invertible but *close to singular*. This happens if columns of $\mathbf{X}$ are almost co-linear or if one column of $\mathbf{X}$ is almost spanned by other columns. When this happens, if we are not careful we will run into numerical trouble. For example:

- Suppose we use the formula $\hat{\mathbf{w}} = \left[\mathbf{X}^\top\mathbf{X}\right]^{-1}\mathbf{X}^\top\mathbf{y}$ and try to compute $[\mathbf{X}^\top\mathbf{X}]^{-1}$ using (say) Gauss elimination, we'll find that Gauss elimination may yield wildly incorrect results.
- Suppose we use the pseudoinverse formula and compute $\mathbf{X}^\dagger$. When $\mathbf{X}^\top\mathbf{X}$ is close to singular, we'll discover that the smallest singular values σ_i of $\mathbf{X}$ are very very small; when we try to compute $1/\sigma_i$ for the pseudoinverse with floating-point arithmetics, $1/\sigma_i$ will not be precise.

There is a simple practical solution for this problem: we choose a "numerical precision threshold" $\varepsilon > 0$ in advance. We can choose, say, $\varepsilon := 10^{-8}$. We then change the definition of the pseudoinverse slightly and define

$$\Sigma^{\dagger,\varepsilon}_{i,i} = \begin{cases} 1/\sigma_i & \sigma_i > \varepsilon \\ 0 & \sigma_i \leq \varepsilon \end{cases}.$$

This ensures that even if the columns of $\mathbf{X}$ are close to being linearly dependent our implementation will be numerically stable.

Recap

Before we continue, let's recap what we have seen so far:

- We have a regression problem over the sample space $\mathcal{X} = \mathbb{R}^d$ and response space $\mathcal{Y} = \mathbb{R}$. We have training data that consists of m samples, $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^d$, and m responses $y_1, \ldots, y_m \in \mathbb{R}$. The i-th coordinate of the vector $\mathbf{x}_i$ is the i features as measured for the i-th training sample. y_i is the response as measured for the i-th sample.
- We're looking for a linear prediction rule $\hat{f} \in \mathcal{H}_{reg}$. A linear prediction rule has an intercept w_0. To simplify notation, we create the m-by-$d+1$ regression matrix X (whose first column is a column of ones) and the response vector $\mathbf{y} \in \mathbb{R}^m$.
- Since any function in $\mathcal{H}_{reg}$ has the form $\mathbf{x} \mapsto \mathbf{x}^\top\mathbf{w}$ for some $\mathbf{w}$, we are looking to choose a vector $\hat{\mathbf{w}} \in \mathbb{R}^{d+1}$
- We decided to choose the vector $\hat{w}$ that minimizes the function $RSS(\mathbf{w}) = ||\mathbf{X}\mathbf{w} - \mathbf{y}||^2$. Why?

in the realizable case where $\mathbf{y} = \mathbf{X}\mathbf{w}$ for some $\mathbf{w}$, the minimum of RSS is zero and the minimizer $\hat{\mathbf{w}}$ will satisfy $\mathbf{y} = \mathbf{X}\hat{\mathbf{w}}$. In the non-realizable case, where there is no solution to the system $\mathbf{y} = \mathbf{X}\mathbf{w}$, minimizing RSS will find a function in $\mathcal{H}_{reg}$ that best fits our training data (in the sense of empirical risk minimization for square loss).

- To find the vector $\hat{\mathbf{w}}$ that minimizes $RSS(\mathbf{w})$, we have to solve the normal equations. This is a linear system of equations that involves $\mathbf{X}$ and $\mathbf{y}$. To solve them, we calculate the singular value decomposition (SVD) of the regression matrix X and take $\hat{\mathbf{w}} = X^{\dagger}\mathbf{y}$. This works both when the columns of X (the training features) are linearly independent and when they are not. When they are independent, this recovers the famous linear regression formula $\hat{\mathbf{w}} = \left[\mathbf{X}^{\top}\mathbf{X}\right]^{-1}\mathbf{X}^{\top}\mathbf{y}$.
- There is a nice geometric interpretation of $\hat{w}$ as the expansion coefficients of $\hat{\mathbf{y}}$, the orthogonal projection of $\mathbf{y}$ on $Im(\mathbf{X})$, in the spanning set that consists of the features (the columns of X).
- When we find the vector $\hat{\mathbf{w}}$ this way, we say that we are *training a linear regression model* using our training data. Once we have found $\hat{\mathbf{w}}$, the training stage is over and we have a prediction rule: for a new (test) sample in $\mathbb{R}^d$ we add 1 to the vector and obtain $\mathbf{x} \in \mathbb{R}^{d+1}$, and predict its response using $\hat{f}(\mathbf{x}) = \mathbf{x}^{\top}\hat{\mathbf{w}}$.
- To make our learning algorithm numerically stable in the case where the columns of $\mathbf{X}$ are almost linearly dependent, we make a slight change in the definition of the pseudoinverse $\mathbf{X}^{\dagger}$ and only include the reciprocals of those singular values that are larger than a predetermined threshold ε.

2.1.4 A Statistical Model - Adding Noise

So far we assumed that the response y was a deterministic function of the sample $\mathbf{x}$, and that there was some deterministic function $f : \mathcal{X} \to \mathcal{Y}$ that underlies the relation $\mathcal{X} \to \mathcal{Y}$. This is an unrealistic assumption - in reality, measurements always contain randomness. In our online store example, we may consider that the revenue y measured for a customer is the sum of a deterministic component $\mathbf{x}^{\top}\mathbf{w}$ (where $\mathbf{x}$ is the customer's feature vector) and some random component z. This means that our dataset will not look like Figure 2.1 even if it is well described by the linear model. Instead is more likely to look like Figure 2.4.

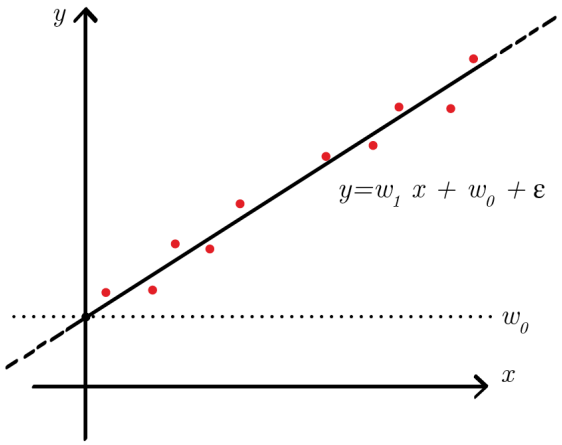


Figure 2.4: *Illustration of a linear regression model where training data is noisy*

To address this problem we describe a probabilistic model of the data. Let us assume, as before,

that the relation $\mathcal{X} \to \mathcal{Y}$ linear, but with an additional factor capturing randomness in the relation. Suppose now that there exists a function $f : \mathcal{X} \to \mathcal{Y}$ such that the response for sample $\mathbf{x}$ is $y = (\mathbf{x}) + z$ where z is some random variable. We assume that the noise z in a sample is identically distributed and independent of the noise in any other sample. In particular, our training sample S is

$$\left(x_i\,,\, f\left(\mathbf{x}_i\right) + z_i\right) \qquad i = 1, \ldots, m$$

with $z_1, \ldots z_m$ being *iid*: independent and identically distributed. Let us adapt the learning algorithm we designed for the deterministic case to the probabilistic (noisy) case. Let us choose the linear hypothesis class $\mathcal{H}_{reg}$ as before, so that our learning algorithm will output a linear prediction rule. We also assume that we have enough training data to learn, namely that $m \geq d+1$. We assume that there is a vector $\mathbf{w} \in \mathbb{R}^{d+1}$ such that for every sample vector $\mathbf{x}_i$ in our data follows the model:

$$y_i = \mathbf{x}_i^\top \mathbf{w} + z_i\,.$$

Denoting the noise vector $\mathbf{z} := (z_1, \ldots, z_m)^\top$ we have in matrix notation

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{z}$$

Note that the vector $\mathbf{y}$ will typically not be in $Im\,(\mathbf{X})$, so that system $\mathbf{y} = \mathbf{X}\mathbf{w}$ has no solutions. As before, using the square loss function, and learning by the empirical risk minimization principal then:

$$\hat{\mathbf{w}} := \text{argmin}_{\mathbf{w}}\, ||y - \mathbf{X}\mathbf{w}||^2$$

This means that our learning algorithm remains the same. We learn by solving the normal equation. As mentioned, $\mathbf{y} \notin Im\,(\mathbf{X})$ since the noise "pushed" $\mathbf{y}$ out of $Im\,(\mathbf{X})$. As we have seen, solving the normal equations is equivalent to projecting $\mathbf{y}$ back onto $Im\,(\mathbf{X})$, so our algorithms effectively attempts to remove the noise and recover the original prediction rule f.

2.1.4.1 The Maximum Likelihood principle

Another approach to solving the problem of linear regression with noise is as follows. Suppose we assume further that the noise is Gaussian $z_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$. This means that the i-th observation is independently distributed $y_i \sim \mathcal{N}(\mathbf{x}_i^\top \mathbf{w}, \sigma^2)$. In vector notation, we are assuming that the responses in our training sample follow a multivariate Gaussian distribution:

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{X}\mathbf{w}, \sigma^2 I_m\right) \tag{2.12}$$

Now, suppose we *knew* the weight vector $\mathbf{w}$, we could then ask the following question: Given a fixed design matrix X and a known coefficients vector $\mathbf{w}$, what is the probability of observing the response vector $\mathbf{y}$? As each sample is independent to the others, the probability density is the product of the Gaussian densities of each sample

$$p(\mathbf{y}|\mathbf{w}) = \prod_{i=1}^{m} \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\mathbf{x}_i^\top \mathbf{w} - y)^2}{2\sigma^2}\right)\right] \tag{2.13}$$

This is a question in probability: We know $\mathbf{w}$ and ask for the chance to observe $\mathbf{y}$?

However, when we design a learning algorithm, we are actually interested in the reverse question. We have the training sample, including the response vector $\mathbf{y}$. We are interested in a way

to choose a linear prediction rule in $\mathcal{H}_{reg}$ and, equivalently, a vector $\mathbf{w}$. We can ask: what is the most "likely" value of $\mathbf{w}$ given the response vector that we observed. This approach is known as the **Maximum Likelihood** (ML) principle. This principle suggests that we choose $\mathbf{w}$ for which the probability density of getting the observed $\mathbf{y}$ is maximal. To make this formal, we first define the likelihood function:

Definition 2.1.2 — Likelihood. Let X be a random variable following some probability distribution $\mathcal{F}$ with a density function f that depends on a parameter $\theta \in \Theta$. The *likelihood function* is

$$\mathcal{L}(\theta|X) := f_\theta(X).$$

As an example, let's find the likelihood function in our case – as a function of the vector $\mathbf{w}$ – given an observed response vector $\mathbf{y}$:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}|X,\mathbf{y}) &= \mathbb{P}\left(y_1 = \mathbf{x}_1^\top \mathbf{w}, \ldots, y_m = \mathbf{x}_m^\top \mathbf{w} | \mathbf{X}, \mathbf{w}\right) \\
&= \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2}{2\sigma^2}\right) \\
&= \frac{1}{(2\pi\sigma^2)^{m/2}} \prod_{i=1}^m \exp\left(-\frac{\left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2}{2\sigma^2}\right) \\
&= \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^m \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2\right)
\end{aligned}$$

Formally, the maximum likelihood estimator chooses the parameter that maximizes the likelihood function:

Definition 2.1.3 — Maximum Likelihood Estimator. Let $\mathcal{L}$ be the likelihood function of some probability distribution $\mathcal{F}$ depending on parameter $\theta \in \Theta$ and let X a random variable following $\mathcal{F}$ with parameter θ. The *Maximum Likelihood Estimator* (MLE) for θ is

$$\widehat{\theta}^{MLE} := \underset{\theta \in \Theta}{argmax}\, \mathcal{L}(\theta|X)$$

As an example, let us find the MLE for our linear regression model **??**:

$$\begin{aligned}
\widehat{\mathbf{w}}^{MLE} &= \underset{\mathbf{w}}{argmax}\, \mathcal{L}(\mathbf{w}|\mathbf{y}) \\
&= \underset{\mathbf{w}}{argmax}\, \log \mathcal{L}(\mathbf{w}|\mathbf{y}) \\
&= \underset{\mathbf{w}}{argmax}\, \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^m \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2\right) \\
&= \underset{\mathbf{w}}{argmin}\, \sum_{i=1}^m \left(\mathbf{x}_i^\top \mathbf{w} - y_i\right)^2
\end{aligned}$$

we therefore conclude that the MLE (assuming i.i.d Gaussian noise) is identical to the Least Squares estimator obtained from a completely different principle - that of empirical risk minimization.

2.2 Polynomial fitting

We now turn to a specific example of linear regression, which will help us gain important general insights, and is also quite useful in and of itself. We will use our learning algorithm for linear regression to predict the value of an unknown real function $g : \mathbb{R} \to \mathbb{R}$. We are given points $a_1 < a_2 < \ldots < a_m \in \mathbb{R}$ and labels $y_1, \ldots, y_m$. In the noiseless case, we know that $y_i = g(a_i)$ for an

unknown g. We would like to predict the value of g on points beyond the training set. The hypothesis class is:

$$\mathcal{H}^d_{poly} = \left\{ x \mapsto p_{\mathbf{w}}(x) \,|\, \mathbf{w} \in \mathbb{R}^{d+1} \right\} \tag{2.14}$$

where $p_{\mathbf{w}}(a) = \sum_{i=1}^{d} w_i a^i$.

Given a training sample $(a_i, y_i)_{i=1}^m$ we would like to choose the polynomial coefficient vector $\mathbf{w}$ using least squares, namely to find

$$\hat{w} := \underset{\mathbf{w}\in\mathbb{R}^{d+1}}{argmin} \frac{1}{m} \sum_{i=1}^{m} (y_i - p_{\mathbf{w}}(a_i))^2 \tag{2.15}$$

Let's create an equivalent linear regression problem. Define the design matrix $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} 1 & a_1 & a_1^2 & \cdots & a_1^d \\ 1 & a_2 & a_2^2 & \cdots & a_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_m & a_m^2 & \cdots & a_m^d \end{bmatrix}$$

Notice that this is a *Vandermonde matrix*, hence it is of full rank. Convince yourself that using our learning algorithm for linear regression for the problem defined by $\mathbf{X}$ and $\mathbf{y}$ finds the vector $\hat{\mathbf{w}}$ from (2.15). After finding $\hat{\mathbf{w}}$, we can predict the value of the unknown function g at a new point a using the value $p_{\hat{\mathbf{w}}}(a)$.

R Here we discuss polynomial fitting where $\mathcal{X} = \mathbb{R}$. With very little adaptation, we could also allow the input data to be $\mathcal{X} = \mathbb{R}^d$, $d > 1$. In such cases the defined polynomial could include terms of multiplication of two (or more) features. We will encounter such an example in **??**.

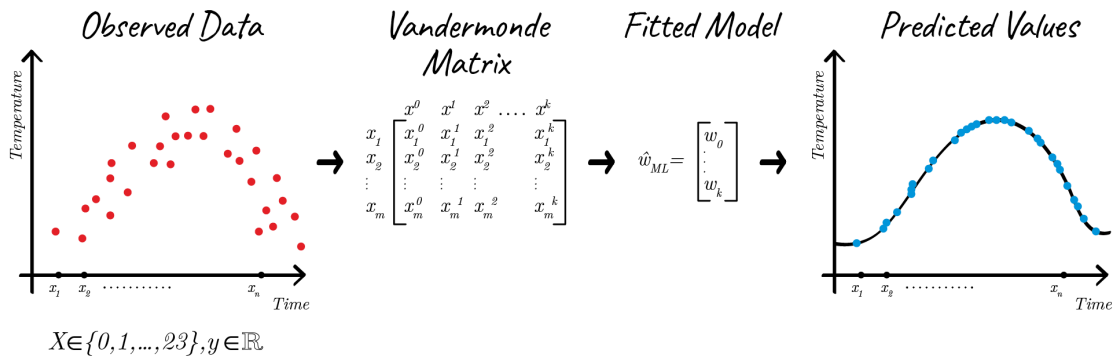


Figure 2.5: Scheme of Polynomial Fitting: *Dataset of hourly temperature where x_i denotes time of day and y_i the temperature. Fitting polynomial of degree* 4.

2.3 Exercises

1. Let $\mathbf{A}$ be some matrix. Show that $Ker(\mathbf{A}) = Ker\left(\mathbf{A}^\top \mathbf{A}\right)$.
2. Let $\mathbf{A}$ be an invertible matrix. Show that $\mathbf{A}^\dagger = \mathbf{A}^{-1}$.
3. Let $\mathbf{X}, \mathbf{y}$ be a linear regression problem where the columns of $\mathbf{X}$ are linearly independent. Show that $\left[\mathbf{X}^\top \mathbf{X}\right]^{-1} \mathbf{X}^\top = \mathbf{X}^{\dagger\top} \mathbf{y}$.