Dauphine | PSL
UNIVERSITÉ PARIS

Lecture notes on stochastic gradient methods

Clément W. Royer

M2 IASD/M2 MASH - 2020/2021

- The last version of these notes can be found at:
  https://www.lamsade.dauphine.fr/~croyer/ensdocs/OML/LectureNotesOML-SGD.pdf.
- Comments, typos, etc, can be sent to `clement.royer@dauphine.psl.eu`.
- **Major updates of the document**
  - 2020.10.22: First release of the notes, with Chapters 1 and 2.

Foreword

The purpose of these lectures is to introduce the stochastic gradient method. Like any presentation of such a widely studied topic, these notes have their own biases. Those include that of the author as well as the references they are mainly based upon [1, 3, 4]. Rather than giving a necessarily incomplete literature review of all variants of this method, and their applications, these notes intend to convey the main principles behind stochastic gradient. More precisely, the goals that the lectures aim for are the following.

- Understand the motivation behind the stochastic gradient algorithm, and its relevance in a learning setting;
- Provide a comparison between stochastic gradient and gradient descent on both a theoretical and a practical standpoint;
- Review major variants on the stochastic gradient framework;
- Observe the performance of stochastic gradient in practice.

Notations

- Scalars (i.e. reals) are denoted by lowercase letters: $a, b, c, \alpha, \beta, \gamma$.
- Vectors are denoted by **bold** lowercase letters: $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$.
- Matrices are denoted by **bold** uppercase letters: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.
- Sets are denoted by **bold** uppercase cursive letters : $\mathcal{A}, \mathcal{B}, \mathcal{C}$.
- The set of natural numbers (nonnegative integers) is denoted by $\mathbb{N}$; the set of integers is denoted by $\mathbb{Z}$.
- The set of real numbers is denoted by $\mathbb{R}$. Our notations for the subset of nonnegative real numbers and the set of positive real numbers are $\mathbb{R}_+$ and $\mathbb{R}_{++}$, respectively.
- The notation $\mathbb{R}^d$ is used for the set of vectors with $d \in \mathbb{N}$ real components; although we may not explicitly indicate it in the rest of these notes, we always assume that $d \geq 1$.
- A vector $\boldsymbol{x} \in \mathbb{R}^d$ is thought as a column vector, with $x_i \in \mathbb{R}$ denoting its i-th coordinate in the canonical basis of $\mathbb{R}^d$. We thus write $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$, or, in a compact form, $\boldsymbol{x} = [x_i]_{1 \leq 1 \leq d}$.
- Given a column vector $\boldsymbol{x} \in \mathbb{R}^d$, the corresponding row vector is denoted by $\boldsymbol{x}^{\mathrm{T}}$, so that $\boldsymbol{x}^{\mathrm{T}} = [x_1 \ \cdots \ x_d]$ and $[\boldsymbol{x}^{\mathrm{T}}]^{\mathrm{T}} = \boldsymbol{x}$. The scalar product between two vectors in $\mathbb{R}^n$ is defined as $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{d} x_i y_i$.
- The Euclidean norm of a vector $\boldsymbol{x} \in \mathbb{R}^d$ is defined by $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}$.
- We use $\mathbb{R}^{n \times d}$ to denote the set of real rectangular matrices with n rows and d columns, where n and d will always be assumed to be at least 1. If $n = d$, $\mathbb{R}^{d \times d}$ refers to the set of square matrices of size d.
- We identify a matrix in $\mathbb{R}^{d \times 1}$ with its corresponding column vector in $\mathbb{R}^d$.
- Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{A}_{ij}$ refers to the coefficient from the i-th row and the j-th column of $\boldsymbol{A}$: the diagonal of $\boldsymbol{A}$ is given by the coefficients $\boldsymbol{A}_{ii}$. Provided this notation is not ambiguous, we use the notations $\boldsymbol{A}$, $[\boldsymbol{A}_{ij}]_{\substack{1 \leq i \leq n \\ 1 \leq j \leq d}}$ and $[\boldsymbol{A}_{ij}]$ interchangeably.
- For every $d \geq 1$, $\mathbf{I}_n$ refers to the identity matrix in $\mathbb{R}^{d \times d}$ (with 1s on the diagonal and 0s elsewhere).

Contents

Chapter 1

Introduction

This course is concerned with optimization problems arising in data-related applications. Such formulations have gained tremendous interest in recent years, due to the increase in computational power that enable significant advances in fields such as image processing. One of the most fundamental tools behind data science is optimization,that combines mathematical formulations and algorithmic procedures. We describe below the motivation behind studying optimization techniques tailored to data-related applications, and provide a detailed description of optimization problems and algorithms.

1.1 Motivation

The words *machine learning* are widely used as a way to characterize any task that involves manipulating data : nevertheless, its precise meaning can be difficult to formalize. Meanwhile, other keywords such as *data mining*, *data analysis*, *artificial intelligence* or *Big Data* also denote fields that involve data and/or a learning process. In these notes, we focus on the link between data-related tasks and optimization; although we will denote our applications of interest as pertaining to machine learning, we point out that a more general, possibly better suited categorization would be that of **data science**.

In this course, we will indeed consider machine learning through two main goals:

1) Extract patterns from data, possibly in terms of statistical properties;

2) Use this information to infer or make predictions about yet unseen data.

A number of such machine learning tasks involve an optimization component, see Figure 1.1. As a result, for the purpose of these notes, we will view machine learning as a field making use of statistics and optimization, with the latter being our area of interest. Nevertheless, we point out that computer science features such as data management and parallel computing have also been instrumental to the success of machine learning, and thus should eventually be integrated with optimization to form efficient algorithms.

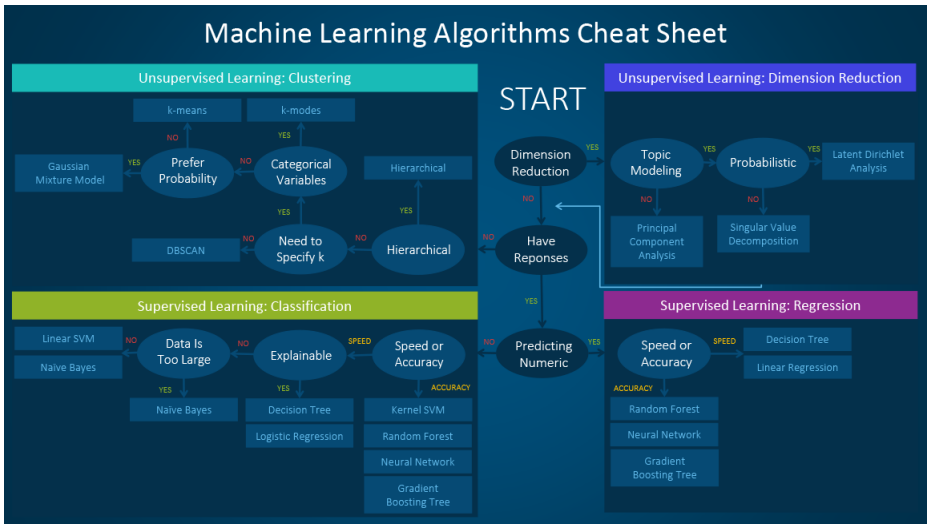

Figure 1.1: A diagram for choosing a machine learning technique appropriate to a given problem; about half of the leaves (Linear SVM, Logistic regression, etc) are directly connected to optimization. *Source: https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/*

1.2 General formulation

1.2.1 From expected to empirical risk

We consider an input space $\mathcal{X}$ and an output space $\mathcal{Y}$. Our goal is to determine a mapping $h : \mathcal{X} \to \mathcal{Y}$ such that, for every input $\boldsymbol{x} \in \mathcal{X}$, the value $h(\boldsymbol{x})$ is an accurate prediction of the true output $\boldsymbol{y} \in \mathcal{Y}$. Suppose that the examples in our dataset are sampled from a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$. We seek a predictor function h that yields a small expected risk, where

$$R(h) := \mathbb{P}\left(h(\boldsymbol{x}) \neq \boldsymbol{y}\right) = \mathbb{E}\left[\mathbf{1}(h(\boldsymbol{x}) \neq \boldsymbol{y})\right], \tag{1.2.1}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function of an event. In practice, we rarely know the distribution of the data, and we can only access a sample $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ of the distribution. In this case, we can quantify how good our prediction is *on this dataset* by considering the **empirical risk** function defined as

$$R_n(h) := \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(h(\boldsymbol{x}_i) \neq \boldsymbol{y}_i). \tag{1.2.2}$$

Unlike the expected risk function, the empirical risk function can usually be computed, as it corresponds to data points that are available. Through arguments based on the law of large numbers, one can ensure that, with sufficiently many samples, the difference between empirical and expected risk can be bounded with high probability.

1.2.2 Loss and prediction functions

The previous measures of risk are exact, in that they directly measure whether a model correctly predicts an output given the input. Their definition can however lead to functions of h that are

discontinuous or combinatorial in nature. This can pose numerous challenges in designing algorithms that compute the model with the lowest risk possible. For this reason, a common practice consists in introducing a **loss function**, that quantifies the discrepancy (or lack thereof) between the output of a model $h(\boldsymbol{x})$ and the true output $\boldsymbol{y}$. That is, for every sample $(\boldsymbol{x}, \boldsymbol{y})$ from the desired distribution, we use

$$\mathbf{1}(h(\boldsymbol{x}) \neq \boldsymbol{y}) \approx \ell(h(\boldsymbol{x}), \boldsymbol{y}),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$ is a given loss function.

In addition, rather than considering a generic set of models, we assume that models can be characterized by means of a vector in $\mathbb{R}^d$ (where d can be extremely large). Therefore, we will use the notation $h(\cdot; \boldsymbol{w}) : \boldsymbol{x} \mapsto h(\boldsymbol{x}; \boldsymbol{w})$.

Overall, we will approximate the expected risk as follows:

$$R(h) \approx R(\boldsymbol{w}) := \int_{\mathcal{X} \times \mathcal{Y}} \ell(h(\boldsymbol{x}; \boldsymbol{w}); \boldsymbol{y})) = \mathbb{E}\left[\ell(h(\boldsymbol{x}; \boldsymbol{w}); \boldsymbol{y})\right], \tag{1.2.3}$$

while the empirical risk will be estimated by

$$R_n(h) \approx R_n(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i; \boldsymbol{w}); \boldsymbol{y}_i)). \tag{1.2.4}$$

Our learning goal will then be to compute a model (that is, a vector $\boldsymbol{w}$) that yields the lowest value of the empirical risk. As a result, we consider the following **empirical risk minimization** (ERM) problem :

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i; \boldsymbol{w}); \boldsymbol{y}_i)). \tag{1.2.5}$$

In the next section, we will see some examples of such ERM problems.

Remark 1.2.1 *Although computing a model based on the empirical risk represents a reasonable approach to computing a good model, the ideal goal would be to compute a solution that would* **generalize** *to unseen examples from the distribution. This is a very challenging issue from an optimization perspective, as optimization classically assumes that the problem formulation encapsulates all there is to know about this problem.*

1.3 Examples of learning models

1.3.1 Linear regression

Linear least squares is arguably the most classical problem in data analysis. We consider a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Our goal is to compute a linear model that best fits (or explains) the data. We define this model as a function $h : \mathbb{R}^d \to \mathbb{R}$, and we parameterize it through a vector $\boldsymbol{w} \in \mathbb{R}^d$, so that for any $\boldsymbol{x} \in \mathbb{R}^d$, we have $h(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{w}$. For every example $(\boldsymbol{x}_i, y_i)$ in the dataset, we evaluate how we fit the data based on the squared error $(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)^2$. We then compute a model by solving the following optimization problem

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{2n}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2. = \frac{1}{n} \sum_{i=1}^{n} \tfrac{1}{2}\left[(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)^2 + \lambda\|\boldsymbol{w}\|^2\right], \tag{1.3.1}$$

where $\lambda > 0$ is a regularization parameter. From an optimizer's point of view, problem (1.3.1) is well understood: this is a strongly convex, quadratic problem, and its solution can be computed in close form.

In a typical linear regression setting, one assumes that there exists an underlying truth but that the measurements are noisy, i.e.

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}^* + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \in \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a vector with i.i.d. entries following a standard normal distribution: this is illustrated in Figure 1.2.

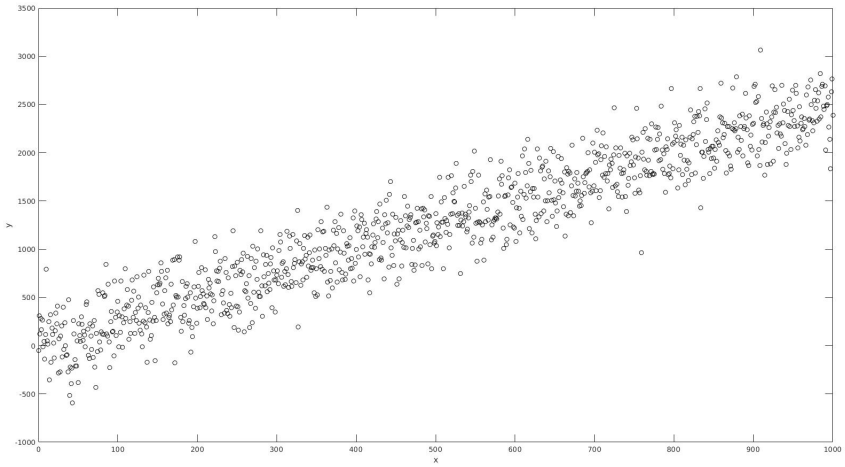

Figure 1.2: Noisy data generated from a linear model with Gaussian noise.

In this setting, we wish to compute the most likely value for $\boldsymbol{w}^*$, while being robust to variance in the data. To this end, we suppose that $\boldsymbol{y}$ follow s a Gaussian distribution of mean $\boldsymbol{X}\boldsymbol{w}$ and of covariance matrix $\mathbf{I}$. We also assume a prior Gaussian distribution on the entries of $\boldsymbol{w}$, in order to reduce the variance with respect to the data. As a result, an estimate of $\boldsymbol{w}^*$, called the maximum a posteriori estimator, can be computed by solving

$$\max_{\boldsymbol{w}\in\mathbb{R}^d} L(y_1, \ldots, y_n; \boldsymbol{w}) := \left[\frac{1}{\sqrt{2\pi}}\right]^m \exp\left(-\frac{1}{2}\sum_{i=1}^{m}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)^2 - \frac{\lambda}{2}\|\boldsymbol{w}\|^2\right). \tag{1.3.2}$$

The solutions of this maximization problem are the same than the solutions of the linear least-squares problem (1.3.1). The resulting solution can be shown to possess very favorable statistical properties: in particular, for λ close to 0, its expected value is close to $\boldsymbol{w}^*$.

Linear regression (with or without the regularization) has been extensively studied in optimization and statistics; however, when the number of samples is extremely large, it still poses a number of challenges in practice, as the solution of the problem cannot be computed exactly.

1.3.2 Neural networks

Neural networks have enabled the most impressive, recent advances in perceptual tasks such as image recognition and classification. Thanks to the increase in computational capabilities realized

over the past decade, it is now possible to train extremely large neural networks, so that they can learn efficient representations of the data.

Given an input vector $\boldsymbol{x}_i \in \mathbb{R}^{d_0}$, a neural network represents a prediction function $h : \mathbb{R}^{d_0} \to \mathbb{R}^{d_J}$, which applies a series of transformations in layers $\boldsymbol{x}_i = \boldsymbol{x}_i^{(0)} \mapsto \boldsymbol{x}_i^{(1)} \mapsto \cdots \mapsto \boldsymbol{x}_i^{(J-1)} \mapsto \boldsymbol{x}_i^{(J)}$. The j-th layer typically performs the following transformation:

$$\boldsymbol{x}_i^{(j)} \;=\; \boldsymbol{\sigma}\left(\boldsymbol{W}_j \boldsymbol{x}_i^{(j-1)} + \boldsymbol{b}_j\right) \in \mathbb{R}^{d_j}, \tag{1.3.3}$$

where $\boldsymbol{W}_j \in \mathbb{R}^{d_j \times d_{j-1}}$, $\boldsymbol{b}_j \in \mathbb{R}^{d_j}$ and $\boldsymbol{\sigma} : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ is a componentwise nonlinear function, e.g. $\boldsymbol{s}(\boldsymbol{y}) = \left[\frac{1}{1+\exp(-y_i)}\right]_i$ (sigmoid function) or $\boldsymbol{s}(\boldsymbol{y}) = [\max(0, y_i)]_i$. As a result, we have $\boldsymbol{x}_i^{(J)} = h(\boldsymbol{x}_i; \boldsymbol{w})$, where $\boldsymbol{w} \in \mathbb{R}^d$ gathers all the parameters $\{(\boldsymbol{W}_1, \boldsymbol{b}_1), \ldots, (\boldsymbol{W}_J, \boldsymbol{b}_J)\}$ of the layers.

The optimization problem corresponding to training this neural network architecture involves a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and the choice of a loss function ℓ, and usually results in the following formulation

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell\left(h(\boldsymbol{x}_i; \boldsymbol{w}), y_i\right). \tag{1.3.4}$$

This optimization problem is highly nonlinear and nonconvex in nature, which makes it particularly slow to solve using algorithms such as gradient descent. Moreover, it typically involves costly algebraic operations, as the number of layers and/or parameters is tremendously large in modern deep neural network architectures. Therefore, problem (1.3.4) also possesses characteristics that are not accounted for in its formulation. The optimization algorithms that efficiently tackle this problem are those that can both guarantee convergence and perform well in practice.

Chapter 2

Stochastic gradient methods

In this chapter, we describe the stochastic gradient method in the context of finite-sum problems, which we introduce in the next section. We will then motivate the use of stochastic gradient methods in this setting, then discuss a basic framework from an algorithmic and theoretical point of view.

2.1 Finite-sum optimization and gradient descent

Suppose that we have access to data samples $\{(x_i, y_i)\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$, $y_i \in \mathbb{R}$, that are drawn from an unknown distribution. As in the examples described in the previous chapter, we seek a predictor function or a model h such that $h(\boldsymbol{x}_i) \approx y_i$ for every $i = 1, \dots, n$. Rather than optimizing over a space of models, we assume that a given model is defined by means of a vector $\boldsymbol{w} \in \mathbb{R}^d$ (i.e. $h(\boldsymbol{x}_i) = h(\boldsymbol{x}_i; \boldsymbol{w})$). Therefore, we only need to determine the vector $\boldsymbol{w}$ in order to obtain the model.

To assess the accuracy of our model in predicting the data, we make use of a loss function $\ell : (h, y) \mapsto \ell(h, y)$, that penalizes pairs (h, y) for which $h \neq y$. The loss at a given sample of the dataset thus is $\ell(h(\boldsymbol{w}; \boldsymbol{x}_i), y_i)$: in order to account for all samples, we consider the average of all losses as our objective to be minimized. This gives rise to the following optimization problem.

Definition 2.1.1 (Finite-sum optimization problem) *Let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be a dataset where for every $i = 1, \dots, n$, $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$ and $y_i \in \mathbb{R}$, a class of predictor functions $\{h(\cdot; \boldsymbol{w})\}_{\boldsymbol{w} \in \mathbb{R}^d}$ and a loss function ℓ, we define the corresponding optimization problem:*

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^n f_i(\boldsymbol{w}), \qquad f_i(\boldsymbol{w}) := \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), y_i) \ \forall i = 1, \dots, n. \tag{2.1.1}$$

One key property of the formulation (2.1.1) is that every term in the finite sum only involves **one example from the dataset**.

Solving the problem with gradient descent Suppose that the functions f_i are continously differentiable. In that case, the objective function f in (2.1.1) also is continuously differentiable, and we can apply the gradient descent method.[1] The k-th iteration of this method is

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k) = \boldsymbol{w}_k - \frac{\alpha_k}{n} \sum_{i=1}^n \ell(h(\boldsymbol{w}; \boldsymbol{x}_i), y_i),$$

[1] See the lecture notes of Gabriel Peyré and Irène Waldspurger for more detail about gradient descent.

where $\alpha_k > 0$ is a given stepsize. From this update, we see that one iteration of gradient descent requires to go over the **entire dataset** in order to compute the gradient vector. In a big data setting where the number of samples n is very large, this cost can be prohibitive.

Remark 2.1.1 *In stochastic optimization, the data samples might be generated directly from the distribution, and be available in a streaming fashion. Instead of involving a discrete average on the sample, the resulting optimization problem would involve a mathematical expectation of the form*

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \mathbb{E}_{(x,y)}\left[f_{(x,y)}(\boldsymbol{w})\right].$$

In such a context, the full gradient may not be computable, even if the underlying function is smooth. However, most of the reasoning in the next sections will apply to this setting.

2.2 Stochastic gradient framework

2.2.1 Algorithm

At its core, the idea of the stochastic gradient method is remarkably simple. Starting from the problem $\min_{\boldsymbol{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{w})$, and assuming each component function f_i is differentiable, the method picks an index i at random and takes a step in the direction of the negative gradient of the component function f_i.

Algorithm 1: Stochastic gradient method.

Initialization: $\boldsymbol{w}_0 \in \mathbb{R}^d$.
for $k = 0, 1, \ldots$ **do**
  Draw a random variable Compute a stepsize or learning rate $\alpha_k > 0$.
  Draw a random index $i_k \in \{1, \ldots, n\}$.
  Compute the new iterate as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k). \tag{2.2.1}$$

end

The key motivation for this process is that using a single data point at a time results in updates that are n **times cheaper than a full gradient step**. Note, however, that using a single component does not necessarily lead to convergence, as illustrated by the following example.

Example 2.2.1 *Consider the problem* $\min_{w\in\mathbb{R}} \frac{1}{2}(f_1(w) + f_2(w))$ *with* $f_1(w) = 2w^2$ *and* $f_2 = -w^2$. *Starting from* $w_k > 0$, *drawing* $i_k = 2$ *will necessarily lead to an increase in the function value.*

In finite-sum problems arising from machine learning, the data samples are correlated enough that an update according to one sample might lead to improvement with respect to other samples as well: this is a key reason for the success of stochastic gradient methods in this setting.

Remark 2.2.1 *Algorithm 1 is often referred to as Stochastic Gradient Descent, or SGD, by analogy with Gradient Descent. However, this algorithm is not a descent method in general (as we will see in the next section, it can however produce descent in expectation). For this reason, we will term these methods as stochastic gradient methods.*

2.2.2 Batch stochastic gradient methods

The main part of Algorithm 1 consists in the update

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k),$$

where the index i_k is drawn at random.. One can thus consider stochastic gradient estimates that are built using *several* samples at once : this is the idea behind batch stochastic gradient.

Formally, the update of a batch stochastic gradient method is given by

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(\boldsymbol{w}_k) \tag{2.2.2}$$

where $S_k \subset \{1, \ldots, n\}$ is drawn at random. When S_k consists in a single index, we recover the usual stochastic gradient algorithm; conceptually, one could also consider a set S_k of cardinality n, in which case we would recover the usual gradient descent algorithm[2].

Overall, two batch regimes can be distinguished:

- $|S_k| \approx n$, which has a cost essentially equivalent to that of a full gradient update;
- $|S_k| = n_b \ll n$, also called mini-batching, which may be advantageous in theory and variance reduction while still being affordable in practice. The resulting method is called mini-batch SG.

We note that it is possible to provide a unified view of batch and stochastic gradient methods; of more interest to us is the comparison of the performance of these various schemes, which is usually done in terms of **epochs**.

Definition 2.2.1 (Epoch) *For problem (2.1.1), an epoch represents n calculations of a sample gradient ∇f_i.*

As a result, one iteration of gradient descent is an epoch, but an epoch corresponds to n iterations of Algorithm 1 and n/n_b iterations of a batch stochastic gradient method with a fixed batch size of n_b.

2.3 Theoretical analysis of stochastic gradient

In this section, we establish convergence guarantees for the stochastic gradient method, in the strongly convex and nonconvex settings. We specify the assumptions under which our analysis is performed, then discuss the dependency of the convergence results to the choice of the stepsize.

2.3.1 Assumptions and first properties

We now describe the main arguments in deriving convergence rates for stochastic gradient. For simplicity, and ease of exposure, we will focus on a specific class of functions.

Assumption 2.3.1 *The objective function $f = \frac{1}{n}\sum_{i=1}^n f_i$ belongs to $\mathcal{C}_L^{1,1}(\mathbb{R}^d)$ for $L > 0$ and there exists $f_{low} \in \mathbb{R}$ such that for every $\boldsymbol{w} \in \mathbb{R}^d$, $f(\boldsymbol{w}) \geq f_{low}$. Moreover, every function f_i belongs to $\mathcal{C}^1(\mathbb{R}^d)$.*

[2] The gradient descent method is sometimes called the batch gradient algorithm in machine learning.

The smoothness assumption above is instrumental to analyzing optimization schemes, as it provides the following upper bound on the objective:

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) + \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k) + \frac{L}{2}\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\|^2. \tag{2.3.1}$$

For the gradient descent method, one can leverage this inequality to guarantee descent at iteration k (that is, $f(\boldsymbol{w}_{k+1}) < f(\boldsymbol{w}_k)$) for an appropriate choice of stepsize[3].

In the case of Algorithm 1, it is not meaningful to study $f(\boldsymbol{w}_{k+1})$, as this value is random. We can however look at its expected value over i_k, which leads to the following result.

Proposition 2.3.1 *Under Assumption 2.3.1, consider the k-th iteration of Algorithm 1. Then,*

$$\mathbb{E}_{i_k}\left[f(\boldsymbol{w}_{k+1})\right] \leq f(\boldsymbol{w}_k) - \alpha_k \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}\, \mathbb{E}_{i_k}\left[\nabla f_{i_k}(\boldsymbol{w}_k)\right] + \frac{L\alpha_k^2}{2}\, \mathbb{E}_{i_k}\left[\|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2\right].$$

In light of Example 2.2.1, we know that the stochastic gradient method may not lead to decrease in the function value. However, we will provide guarantees in expectation under additional assumptions on how the stochastic gradient estimate $\nabla f_{i_k}(\boldsymbol{w}_k)$.

Assumption 2.3.2 (Assumptions on stochastic gradient) *At any iteration of Algorithm 1 of index k, i_k is drawn such that:*

1. *The index i_k does not depend from the previous indices $i_0, \ldots, i_{k-1}$;*
2. $\mathbb{E}_{i_k}\left[\nabla f_{i_k}(\boldsymbol{w}_k)\right] = \nabla f(\boldsymbol{w}_k)$;
3. $\mathbb{E}_{i_k}\left[\|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2\right] \leq \sigma^2 + \|\nabla f(\boldsymbol{w}_k)\|^2$ *with* $\sigma^2 \geq 0$.

The first property of Assumption 2.3.2 forces the stochastic gradient $\nabla f_{i_k}(\boldsymbol{w}_k)$ to be an unbiased estimate of the true gradient $\nabla f(\boldsymbol{w}_k)$. The second property controls the variance of the norm of this stochastic gradient, so as to control the variations in its magnitude due to noise. Several strategies can be designed to draw an index i_k that satisfies these properties, the most classical of which is given below.

Example 2.3.1 (Uniform sampling) *Suppose that the indices $\{i_k\}_k$ are i.i.d. random variables that are uniformly at random in $\{1, \ldots, n\}$. Then Algorithm 1 satisfies Assumption 2.3.2.*

Under these assumptions, we have the following result.

Proposition 2.3.2 *Under Assumptions 2.3.1 and 2.3.2, at the k-th iteration of Algorithm 1, one has*

$$\mathbb{E}_{i_k}\left[f(\boldsymbol{w}_{k+1}) - f(\boldsymbol{w}_k)\right] \leq -\left(\alpha_k - \frac{L\alpha_k^2}{2}\right)\|\nabla f(\boldsymbol{w}_k)\|^2 + \frac{L\alpha_k^2}{2}\sigma^2. \tag{2.3.2}$$

A stochastic gradient update will thus lead to decrease **in expectation**. Such a property suffices to derive convergence rates (or complexity results) for stochastic gradient applied to strongly convex, convex or nonconvex problems. Those results heavily depend upon the formula for the step sizes $\{\alpha_k\}_k$.

[3] See the previous lectures of G. Peyré and I. Waldspurger.

2.3.2 Analysis in the strongly convex case

We will illustrate the various challenges posed by this choice in the context of strongly convex functions.

Assumption 2.3.3 *There exists $\mu > 0$ such that the objective function is μ-strongly convex, i.e. for every $(\boldsymbol{w}, \boldsymbol{x}) \in (\mathbb{R}^d)^2$, we have*

$$f(\boldsymbol{x}) \geq f(\boldsymbol{w}) + \nabla f(\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{w}) + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{w}\|^2. \tag{2.3.3}$$

and possesses a unique global minimizer $\boldsymbol{w}^*$. We let $f^* = f(\boldsymbol{w}^*)$.

Assumption 2.3.3 has the following useful consequence.

Lemma 2.3.1 *Let Assumptions 2.3.1 and 2.3.3 hold. Then, for every $\boldsymbol{w} \in \mathbb{R}^d$, we have*

$$\|\nabla f(\boldsymbol{w})\|^2 \;\geq\; 2\mu\,(f(\boldsymbol{w}) - f^*)\,. \tag{2.3.4}$$

Our results will be provided in expectation. For any k, we will exploit the independence assumption on the indices i_k and write

$$\mathbb{E}\left[f(\boldsymbol{w}_k)\right] = \mathbb{E}_{i_0}\left[\mathbb{E}_{i_1}\left[\ldots \mathbb{E}_{i_{k-1}}\left[f(\boldsymbol{w}_k)\right]\right]\right].$$

We first provide a global rate result in the case of a constant step size.

Theorem 2.3.1 (SG with constant stepsize) *Let Assumptions 2.3.1, 2.3.2 and 2.3.3 hold. Consider Algorithm 1 applied with a constant stepsize*

$$\alpha_k = \alpha \in \left(0, \tfrac{1}{2L}\right] \forall k.$$

Then,

$$\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \frac{\alpha L \sigma^2}{2\mu} + (1 - \alpha\mu)^k \left[f(\boldsymbol{w}_0) - f^* - \frac{\alpha L \sigma^2}{2\mu}\right]. \tag{2.3.5}$$

At first glance, it thus seems that we obtain a convergence rate similar to that of gradient descent. Indeed, for gradient descent, one can show under the same assumption on α_k that

$$f(\boldsymbol{w}_k) - f^* \leq (1 - \alpha\mu)^k \left[f(\boldsymbol{w}_0) - f^*\right].$$

In the case of Algorithm 1, however, we observe that residual terms appear in the convergence rate, that are related to the variance of the gradient estimator. As a result, SG with constant stepsize can only be guaranteed to converge towards a **neighborhood** of the optimal function value f^*. The result of Theorem 2.3.1 illustrates this interplay between the choice of the (constant) stepsize and the residual noise in the problem: if we set α to a small value, the variance-related terms are reduced, but the convergence rate of the method is slower.

Remark 2.3.1 (A practical constant stepsize approach) *A common practical strategy in machine learning consists in running the algorithm with a value α until the method stalls (which can indicate that the smallest neighborhood attainable with this stepsize choice has been reached). When that occurs, the stepsize can be reduced, and the algorithmic run can continue until it stalls*

again, then the stepsize will be further reduced, etc (say $\alpha, \alpha/2, \alpha/4$, etc). This process can lead to convergence guarantees, in that it is possible to reach any neighborhood of the optimal value f^*. However, the convergence rate is sublinear, in the sense that

$$\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \mathcal{O}\left(\frac{1}{k}\right)$$

This choice of stepsize is adaptive, in that it is designed to reach closer and closer neighborhoods as the algorithm proceeds. However, it requires the method to be able to detect stalling, and act upon it.

In the original stochastic gradient method (proposed by Robbins and Monro in 1951), the stepsize sequence was required to satisfy

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty,$$

which implies that $\alpha_k \to 0$. In our next result, we thus consider the case of diminishing stepsizes.

Theorem 2.3.2 (SG with diminishing stepsize) *Let Assumptions 2.3.1, 2.3.2 and 2.3.3, and consider Algorithm 1 applied with a decreasing stepsize sequence $\{\alpha_k\}_k$ satisfying*

$$\alpha_k = \frac{\beta}{k+\gamma},$$

where $\beta > \frac{1}{\mu}$ and $\gamma > 0$ is chosen such that $\alpha_0 = \frac{\beta}{\gamma} \leq \frac{1}{L}$. Then,

$$\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \frac{\nu}{\gamma + k}, \tag{2.3.6}$$

where

$$\nu = \max\left\{\gamma(f(\boldsymbol{w}_0) - f^*), \frac{\beta^2 L \sigma^2}{2(\beta\mu - 1)}, \right\}.$$

From the result of Theorem 2.3.2, we see that choosing a decreasing stepsize results in a sublinear convergence rate, which is worse than the rate for stochastic gradient with constant stepsize. However, note that such a choice enables to reach any neighborhood of the optimal value.

Remark 2.3.2 *Choosing the stepsize (or, in machine learning language,* tuning the learning rate*) is one of the most critical issues in implementing stochastic gradient methods. As the rates above suggest, defining the stepsize according to the Lipschitz and strong convexity constants is critical to the performance of the method.*

Results for mini-batch variants Suppose that we use a batch variant of stochastic gradient with a fixed batch size n_b, under Assumption 2.3.2. Then, if the sample sets are drawn uniformly at random, it is possible to show that the variance term σ^2 is reduced to $\frac{\sigma^2}{n_b}$. We can then study the impact of this result on the convergence rate; if we choose a constant stepsize $\alpha > 0$, for instance, one obtains

$$\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \frac{\alpha L \sigma^2}{2\mu n_b} + (1 - \alpha\mu)^k \left[f(\boldsymbol{w}_0) - f^* - \frac{\alpha L \sigma^2}{2\mu n_b}\right].$$

This result could also be obtained by choosing the stepsize $\frac{\alpha}{n_b}$ in Algorithm 1, i. e. standard stochastic gradient. In terms of worst-case cost, we thus see that the algorithms appear comparable. For a given neighborhood of the optimal value, a batch variant can converge in the same number of iterations using larger stepsizes, however those iterations are more expensive. The interest of using a batch size comes from the possibility of parallelizing (stochastic) gradient evaluations, provided a distributed environment is available.

2.3.3 Analysis of stochastic gradient in the nonconvex case

Stochastic gradient (or some variant thereof) is the method of choice for training neural networks, which is usually a nonconvex problem, as illustrated in section 1.3.2. It is thus natural to ask whether global rates can be obtained for stochastic gradient in the nonconvex setting. For gradient descent, we know that one can guarantee (e.g. using a constant stepsize) that for any $K \in \mathbb{N}$,

$$\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\| \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

which is sometimes established as

$$\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\|^2 \leq \mathcal{O}\left(\frac{1}{K}\right).$$

As we will see, the guarantees that one can establish in the stochastic setting are affected by noise.

We begin by deriving a result in the context of constant stepsizes.

Theorem 2.3.3 *Let Assumptions 2.3.1 and 2.3.2 hold. Suppose that Algorithm 1 is run with a constant stepsize $\alpha_k = \alpha > 0$ where $\alpha \in \left(0, \frac{1}{L}\right]$. Then, for any $K \in \mathbb{N}$,*

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1} \|\nabla f(\boldsymbol{w}_k)\|^2\right] \leq \frac{\alpha L \sigma^2}{\mu} + \frac{2(f(\boldsymbol{w}_0) - f^*)}{\alpha K}. \tag{2.3.7}$$

As in the strongly convex case, we see that the noise prevents from guaranteeing (as in the gradient descent case) that the sum of squared gradients remains finite. Note that the second term on the right-hand side of (2.3.7) corresponds to the usual (sublinear) convergence rate of gradient descent. Note also that the result of Theorem 2.3.3 indicates that Algorithm 1 spends increasingly more time in regions for which the gradient norm is small, up to a certain level determined by the noise.

In the case of decreasing stepsizes, we can provide the following guarantee.

Theorem 2.3.4 *Let Assumptions 2.3.1 and 2.3.2 hold. Suppose that Algorithm 1 is run with a decreasing stepsize sequence $\{\alpha_k\}$ such that $\alpha_k \in (0, \frac{1}{L}]$ for every k, and the sequence satisfies*

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \textit{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 = \infty.$$

Then, we have

$$\mathbb{E}\left[\frac{1}{\sum_{k=0}^{K-1}\alpha_k}\sum_{k=0}^{K-1} \alpha_k \|\nabla f(\boldsymbol{w}_k)\|^2\right] \to 0 \quad \textit{as} \quad K \to \infty. \tag{2.3.8}$$

Theorem 2.3.4 shows that a weighted sum of squared gradients converges to zero regardless of the noise level (which could not be guaranteed with a constant stepsize). We can then derive the following corollaries.

Corollary 2.3.1 *Let the assumptions of Theorem 2.3.4 hold. For any $K \in \mathbb{N}$, let $k(K)$ be a random index chosen such that*

$$\mathbb{P}\left(k(K) = k\right) = \frac{\alpha_k}{\sum_{k=0}^{K-1} \alpha_k} \quad \forall k = 0, \dots, K-1.$$

Then, $\|\nabla f(\boldsymbol{w}_{k(K)})\| \to 0$ in probability as $K \to \infty$, i.e.

$$\forall \epsilon > 0, \ \mathbb{P}\left(\|\nabla f(\boldsymbol{w}_{k(K)})\| \geq \epsilon\right) \to 0 \quad \textit{as} \quad K \to \infty.$$

Note that results similar to those above can be derived using a batch approach, with appropriate changes in the noise level. Those lead to the same observations as in the strongly convex case.

2.4 Concluding remarks

To end this section, we discuss several follow-ups on our analysis, both theoretical and practical.

Conditioning All our results heavily depend on the Lipschitz constant for the gradient (and, in the strongly convex case, on the strong convexity constant). Ill-conditioned problems, for which the ratio $\frac{L}{\mu}$ is much larger than 1, pose a significant challenge for stochastic gradient, as the speed of the method depends on this ratio. Similarly, a large Lipschitz constant enforces restrictions on the stepsize that may lead to small, and thus inefficient steps. Rescaling techniques, that transform the objective (either locally or globally), can improve these constants and lead to dramatic speed-ups in practice.

Sharpness The rates achieved by stochastic gradient are sharp, in that under the same set of assumptions (and access to a stochastic gradient oracle), there does not exist a method that has a better rate than stochastic gradient (for instance, $\mathcal{O}(1/k)$ with a decreasing step size). Under additional assumptions on the problem, however, methods with improved complexity bounds can be developed.

Extension to the online setting We have presented the analysis in the case of the finite-sum problem (2.1.1), but it is also possible to generalize the analysis to stochastic optimization problem involving an expected value. Under relatively little assumptions about the problem and the method, one typically obtain convergence rates in expectation for convex and nonconvex problems.

Chapter 3

Advanced concepts in stochastic gradient methods

Chapter 4

Conclusion

From a pure optimization perspective, stochastic gradient methods may not seem so attractive, as they only rely on partial information from the gradient and possess worse convergence guarantees than gradient descent. However, they have encountered tremendous success in data-related applications, where computing gradients involves looking at the entire data and is thus too prohibitive. On the contrary, using stochastic gradient estimates represents a significantly cheaper cost per iteration; in a data science setting, where there can be redundancies (or even underlying randomness) in the data, such updates do not necessarily hinder the progress of the algorithm, but rather lead to faster convergence in practice.

Bibliography

[1] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.*, 60:223–311, 2018.

[2] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, Cambridge, United Kingdom, 2019.

[3] J. C. Duchi. Introductory lectures on stochastic optimization. In A. C. Gilbert M. W. Mahoney, J. C. Duchi, editor, *The mathematics of data*, number 25 in IAS/Park City Mathematics Series. AMS, IAS/Park City Mathematics Institute, and Society for Industrial and Applied Mathematics, Princeton, 2018.

[4] R. M. Gower. Lecture notes on stochastic gradient. M2 IASD, 2019.

[5] S. J. Wright. Optimization algorithms for data analysis. In A. C. Gilbert M. W. Mahoney, J. C. Duchi, editor, *The mathematics of data*, number 25 in IAS/Park City Mathematics Series. AMS, IAS/Park City Mathematics Institute, and Society for Industrial and Applied Mathematics, Princeton, 2018.

Appendix A

Basics in probability and statistics

A.1 Probability theory

The concept of probability originates from measure theory. All results in probability and statistics implicitly rely on probability spaces, i.e. triplets $(\Omega, \mathcal{A}, \mathbb{P}())$, where

- Ω is a set of possible values, or outcomes;
- $\mathcal{A}$ is a family of subsets of Ω called set of events, that satisfy certain properties that make it a σ-algebra;
- $\mathbb{P}(\,) : \mathcal{A} \to [0,1]$ is a probability measure, that satisfies in particular $\mathbb{P}(()\emptyset) = 0$ et $\mathbb{P}(()\Omega) = 1$.

Given this definition, a random variable is a mapping from a probability space to another space that induces a new probability measure on the latter. The term *random variable* is often used for scalar quantities, thus we will make a distinction between random variables and random vectors defined as follows:

- random variables z defined on a probability space $(\mathbb{R}, \mathcal{B}(\mathbb{R}), \mathbb{P})$ by
$$\forall B \in \mathcal{B}(\mathbb{R}), \quad \mathbb{P}(z \in B) = \mathbb{P}(B);$$
- random vectors $\boldsymbol{z} = \begin{bmatrix} z_1 \\ \cdots \\ z_d \end{bmatrix}$ of size d, defined on the probability space $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d), \mathbb{P})$.

In both case, the set of events will be the Borel σ-algebra $\mathcal{B}(\mathbb{R}^d)$.

A.2 Random variables

Although a generic study of random variables can be performed by considering them as taking a continuum of values, we begin by providing the more elementary definition of discrete random variables.

Definition A.2.1 (Discrete random variable) *A discrete random variable z is defined by*

- *A discrete set of possible values* $\mathcal{Z} = \{z_i\} \subset \mathbb{R}$;
- *An associated set of probabilities* $p = \{p_i\}$ *such that* $p_i \geq 0$, $\sum_i p_i = 1$ *and*

$$\forall \mathcal{S} \subset \mathcal{Z}, \quad \mathbb{P}(z \in \mathcal{S}) = \sum_{z_i \in \mathcal{S}} p_i.$$

Definition A.2.2 (Continuous random variable) *A continuous random variable* z *is defined by*

- *A continuous set of possible values* $\mathcal{Z} \subset \mathbb{R}$;
- *An associated probability density* $p : \mathcal{Z} \to \mathbb{R}^+$ *such that* $\int_{\mathbb{R}} p(z)\, dz = 1$ *and*

$$\forall \mathcal{S} \subset \mathcal{Z}, \quad \mathbb{P}(z \in \mathcal{S}) \; = \; \int_{z \in \mathcal{S}} p(z)\, dz.$$

For both continuous and discrete random variables, we will say that z follows a distribution characterized by $(p, \mathcal{Z})$, or simply p when the set of possible values is implicit from the definition of p.

To understand the behavior of random variables, one can look at the moments of their distribution (provided they are well defined). The most classical moment is the mean of a random variable.

Definition A.2.3 (Expected value/Mean) *Let* z *be a random variable with a distribution* $(p, \mathcal{Z})$*, which we indicate as* $z \sim p$*. The* **expected value of** z *is defined by*

$$\mathbb{E}[z] = \mathbb{E}_z[z] = \begin{cases} \sum_{z_i \in \mathcal{Z}} z_i\, p(z = z_i) & \text{(discrete case)} \\ \int_{\mathcal{Z}} z\, p(z)\, dz & \text{(continuous case).} \end{cases}$$

The expected value has several desirable properties that facilitate its use, especially the following.

Proposition A.2.1 *The expected value is a linear operator: that is, for every random variable* z *and every* $\alpha, \beta \in \mathbb{R}$*, one has:*

$$\mathbb{E}[\alpha\, z + \beta] = \alpha\, \mathbb{E}[z] + \beta;$$

The expected

Definition A.2.4 (Variance and standard deviation) *Let* z *be a random variable.*

- *The variance of* z *is defined by*

$$\mathrm{Var}[z] = \mathbb{E}\left[z^2\right] - \mathbb{E}[z]^2.$$

- *The standard deviation of* z *is the square root of the variance.*

Lemma A.2.1

- *If* z *is a discrete random variable, then* $\mathrm{Var}[z] = \sum_i p_i z_i^2 - [\sum_i p_i z_i]^2$;
- *If* z *has zero mean, i.e.* $\mathbb{E}[z] = 0$*, then* $\mathrm{Var}[z] = \mathbb{E}\left[z^2\right]$.

A.3 Pair of random variables

When two random variables possess the same distribution on the same probability space, we say that those variables are **identically distributed**. In a general setting, one can study the distribution of the pair formed by two random variables.

Definition A.3.1 (Joint distribution (discrete case)) *Let z and w be two discrete random variables taking values in $\mathcal{Z} = \{z_i\}$ and $\mathcal{W} = \{w_j\}$, respectively. The distribution of the pair of random variables (z, w) is defined by*

- *The set of possible values $\mathcal{Z} \times \mathcal{W} = \{(z_i, w_j)\}$;*
- *The discrete probability density $p = \{p_{i,j}\}$, where*

$$p_{i,j} = \mathbb{P}(z = z_i, w = w_j).$$

Definition A.3.2 (Joint distribution (continuous case)) *Let z and w be two continuous random variables taking values in $\mathcal{Z}$ and $\mathcal{W}$. The distribution of the pair of random variables (z, w) is defined by*

- *The set of possible values $\mathcal{Z} \times \mathcal{W}$;*
- *The continuous probability density $p : \mathcal{Z} \times \mathcal{W} \to \mathbb{R}^+$ such that*

$$\int_z \int_w p(z, w)\, dz\, dw \;=\; 1.$$

In the above definitions, we started from two random variables to obtain the joint distribution of the pair formed by these variables. It is also possible to go the other way around, by defining marginal laws.

Definition A.3.3 (Marginal laws (discrete case)) *Let z and w be two discrete random variables taking values in $\mathcal{Z} = \{z_i\}$ and $\mathcal{W} = \{w_j\}$, respectively. Let $\{p_{i,j}\}$ be the joint distribution of (z, w).*

- *The marginal law of z is given by $\{p_{i\bullet}\}_i$, where*

$$p_{i\bullet} := \mathbb{P}(z = z_i) = \sum_{j | w_j \in \mathcal{W}} \mathbb{P}(z = z_i,\ w = w_j) = \sum_j p_{i,j}.$$

- *Similarly, the marginal law of w is given by $\{p_{\bullet j}\}_j$, where*

$$p_{\bullet j} := \mathbb{P}(w = w_j) = \sum_{i | z_i \in \mathcal{Z}} \mathbb{P}(z = z_i,\ w = w_j) = \sum_i p_{i,j}.$$

Definition A.3.4 (Marginal laws (continuous case)) *Let z and w be two continuous random variables taking values in $\mathcal{Z}$ and $\mathcal{W}$, respectively. Let $p : (z, w) \mapsto p(z, w)$ be the joint density of (z, w).*

- *The marginal law of z, denoted by p_z or $p(z,\bullet)$, is the function $p_z : \mathcal{Z} \to \mathbb{R}^+$ given by*

$$\forall z \in \mathcal{Z}, \quad p_z(z) = \int_{\mathcal{W}} p(z,w)\, dw.$$

- *The marginal law of w, denoted by p_w or $p(\bullet,w)$, is the function $p_w : \mathcal{W} \to \mathbb{R}^+$ given by*

$$\forall w \in \mathcal{W}, \quad p_w(w) = \int_{\mathcal{Z}} p(z,w)\, dz.$$

Definition A.3.5 (Covariance and correlation) *Let z and w be two random variables. The covariance of z and w is defined by*

$$\mathrm{Cov}\left[z,w\right] = \mathbb{E}_{z,w}\left[\left(z - \mathbb{E}\left[z\right]\right)\left(w - \mathbb{E}\left[w\right]\right)\right].$$

The correlation of z and w is

$$\mathrm{Corr}\left[z,w\right] = \frac{\mathrm{Cov}\left[z,w\right]}{\sqrt{\mathrm{Var}_z\left[z\right]}\sqrt{\mathrm{Var}_w\left[w\right]}}.$$

Independent random variables Independence is widely used in statistics, where it is often combined with the notion of identically distributed variables: we then say that the random variables are **i.i.d.**, which stands for "independent, identically distributed".

Definition A.3.6 (Independent variables) *Let z and w be two random variables with distributions $(p_z, \mathcal{Z})$ and $(p_w, \mathcal{W})$, respectively. The variables z and w are called* **independent** *if the pair (z,w) satisfies*

$$\forall \mathcal{S} \times \mathcal{T} \subset \mathcal{Z} \times \mathcal{W}, \quad \mathbb{P}\left(z \in \mathcal{S}, w \in \mathcal{T}\right) \;=\; \mathbb{P}\left(z \in \mathcal{S}\right)\mathbb{P}\left(w \in \mathcal{T}\right).$$

Independence allows an easy characterization of the joint distribution, as illustrated by the following result.

Proposition A.3.1 *Let z and w be two independent random variables. Then, their joint distribution is obtained as the product of the marginal distributions. We thus have*

$$\begin{cases} p_{ij} = p_{i\bullet} \times p_{\bullet j} & \textit{(discrete case)} \\ p(z,w) = p_z(z) \times p_w(w) & \textit{(continuous case).} \end{cases}$$

Proposition A.3.2 *Let z and w be two independent random variables. Then, these values are decorrelated, i. e. $\mathrm{Cov}\left[z,w\right] = \mathrm{Corr}\left[z,w\right] = 0$.*

A.4 Multidimensional statistics

Most of the previous results on random variables can be extended to the case of **random vectors**, i.e. multidimensional random quantities. We provide below the basic concepts.

Definition A.4.1 (Law of a random vector) *Let $\boldsymbol{z} = [z_i]_i$ be a random vector in $\mathbb{R}^n$: the law (or the distribution) of $\boldsymbol{z}$ is given by the joint distribution of its components. In particular, we define the following moments of this distribution:*

- *the expected value of $\boldsymbol{z}$ is the vector of the expected values of each component:*

$$\mathbb{E}[\boldsymbol{z}] = \{\mathbb{E}[z_i]\}_i \in \mathbb{R}^n;$$

  where the expected value is taken with respect to $\boldsymbol{z}$;

- *the* **covariance** *matrix of $\boldsymbol{z}$, denoted by $\mathrm{Var}[\boldsymbol{z}]$ or $\boldsymbol{\Sigma}_{\boldsymbol{z}}$ is the matrix of the covariances between each component*

$$\forall 1 \leq i, j \leq n, \quad [\boldsymbol{\Sigma}_{\boldsymbol{z}}]_{i,j} := \mathbb{E}[(z_i - \mathbb{E}[z_i])(z_j - \mathbb{E}[z_j])].$$

Note that the covariance matrix can be written as

$$\boldsymbol{\Sigma}_{\boldsymbol{z}} = \mathbb{E}\left[(\boldsymbol{z} - \mathbb{E}[\boldsymbol{z}])(\boldsymbol{z} - \mathbb{E}[\boldsymbol{z}])^{\mathrm{T}}\right] \in \mathbb{R}^{n \times n}.$$

Lemma A.4.1 *If the components of a random vector are independent, then its covariance matrix is diagonal.*

A.5 Markov's inequality

Many statistical results rely on providing bounds on probability levels, or moments of a given random variable. One of the most prominent results in this respect, due to Markov, is given below.

Theorem A.5.1 (Markov's inequality) *Let x be a nonnegative random variable and $\epsilon > 0$. Then,*

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