Deep Approximation and Stochastic Gradient Descent

Peio Ibarrondo Murguialday

Máster en Matemáticas y Aplicaciones

MÁSTERES DE LA UAM

2019 - 2020

Facultad de Ciencias

Universidad Autónoma de Madrid

Departamento de Matemáticas, Facultad de Ciencias
Universidad Autónoma de Madrid

Deep Approximation and Stochastic Gradient Descent

Master's Thesis

Master in Mathematics and Applications

Author: Peio Ibarrondo Murguialday
Tutor: Davide Barbieri

Course 2019-2020

Abstract

In the last decade, deep neural networks have become remarkably popular due to the great result that they perform in practice. However, why they are so empirically efficient is uncertain so far. In this thesis, we try to clarify the black box of deep learning. For this purpose, we first study the approximation ability of neural networks in terms of their depth and the number of weights to deduce that deep neural networks can perform faster convergence than shallow networks. Then, we focus on one of the most used learning process named Stochastic Gradient Descent. We approximate the learning iteration by a stochastic process in order to analyse its asymptotic behaviour. Finally, we conclude that this method implicitly performs variational inference and regularization, which are quite desired properties in supervised learning.

Contents

CHAPTER 1

Introduction

In this age of digitalization that we are living, the amount of data that each of us generates has increased extremely. In order to deal with such quantities of information and thanks to the computation power that we have nowadays, different artificial intelligence tasks have been developed such as image classification and speech recognition.

In the last decades, deep learning has been the main tool that has improved the-state-of-the-art in this subject. This technique arises from the idea of mimicking the neural activity of human brain by a computer. In the middle of 20th century, neuropsychologist D. Hebb (1904-1985) created a learning hypothesis based on the mechanism of neuronal plasticity that became known as Hebbian learning. The main idea of this theory was that when two neurons or systems of neurons are repeatedly activated at the same time, they will tend to become associated so that activity in one facilitates activity in the other.

Inspired by this research, F. Rosenblatt constructed the electronic device named Perceptron in 1958, which showed the ability to learn in accordance with associationism. He introduced the perceptron as a binary classifier within the context of vision system. Its mechanism can be described as follows:

- Consider an input data given by a vector $x \in \mathbb{R}^n$.
- Apply an affine transformation $T : \mathbb{R}^n \to \mathbb{R}$, defined by $T(x) = \sum_{i=1}^{n} w_i x_i + w_0$, where the elements w_i of the matrix associated to T are called *weights*.
- Finally, an *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$, which is a non-linear function, is used in order to classify the data. Initially, a common choice was the Heaviside function, but also smoother functions such as tanh and the logistic function were considered.

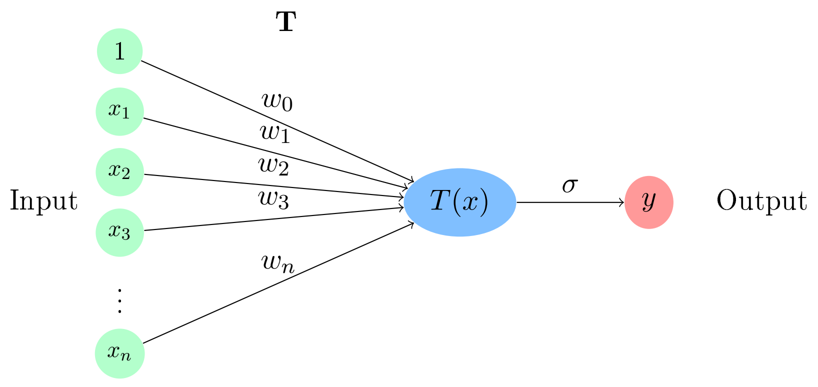

Figure 1.1: Visualization of perceptron.

The learning process that the perceptron performs is based on considering the weights as parameters to be chosen in order to obtain a better classification. Therefore, given a training sample, those weights are modified progressively by minimization of the error of classification. This leads to an early implementation of so-called supervised classification.

The perceptron is a linear classifier, which means that it is useful only if the data can be appropriately separated by a hyperplane. However, this limitation was overcome with the appearance of the neural networks. The main idea was just to put several perceptrons together in parallel in order to get a vector-valued perceptron, and in series, in order to obtain a network made of many "layers" of neurons.

In this case, the number of affine transformations $T_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ is equal to the number of layers of the network and the activation function is applied in each neuron of the layers. Since we want to generalize the functions that can be constructed with this architecture, we do not use the activation function in the output. Notice that if we want to obtain a binary classifier, it suffices to apply a threshold $\mathbb{I}_{\{x>a\}}$ to the output.

Therefore, for a fixed vector of weights W, the neural network represents a function $\eta_W : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ given by the composition of affine transformations and the activation function:

$$\eta_W(x) = T_L \circ \sigma \circ T_{L-1} \circ \cdots \circ \sigma \circ T_1(x).$$

Let remark that all layers in between the input and the output are called "hidden". The power of this new tool was shown by the Universal Approximation Theorem, proved by G. Cybenko in 1989. The theorem states that any continuous function with compact support on $\mathbb{R}^n$ can be approximated with arbitrarily small error by a 2-layer network (that is, a network with 1 hidden layer) with a sigmoid-type activation function. Nevertheless, the width of the hidden layer will increase exponentially as the error get smaller, which represents a strong limitation on any computational implementation of this network.

As we will see in Chapter 2, neural networks with many layers, that is, deep neural networks, are able to approximate continuous functions with optimal rate of convergence. The aim of the first part of this thesis will be to prove this statement given by D. Yarotsky in 2018, which suggest that deep networks can be much more efficient than shallow networks.

On the other hand, the main issue that has been discussed about neural networks is the learning process. Once we have that the architecture of the neural network is good enough, how can we choose the weights W in order to get the desired function? This is the essential question about this topic that we will try to answer. Notice that in image recognition for example, the function that we are looking for is an underlying classifier of images which is unknown, so we desire to get the optimal values of the weights based on a sample of images previously classified.

The most common process of choosing, or "learning", the weights is called *Backpropagation*, announced by D. Rumelhart, G. Hinton and R. Williams in 1986. As we will show in Chapter 3, it is a method based on gradient descent, but instead of updating all the parameters with respect to the error, backpropagation first propagates the error term from output layer back to the layer at which parameters need to be updated and then uses standard gradient descent to update parameters with respect to the propagated error. Intuitively, the derivation of backpropagation is about organizing the terms when the gradient is expressed with the chain rule.

This classic method leads us to an open problem: How can we ensure that optimizing the weights does not reach a local minimum of the error which is not global? Convexity provides an answer to this question, but in high-dimensional spaces convexity properties are quite difficult to obtain. That is an instance of what is known as the curse of dimensionality. Moreover, we cannot compute the error of the whole training sample at once and then look for the best weights, since

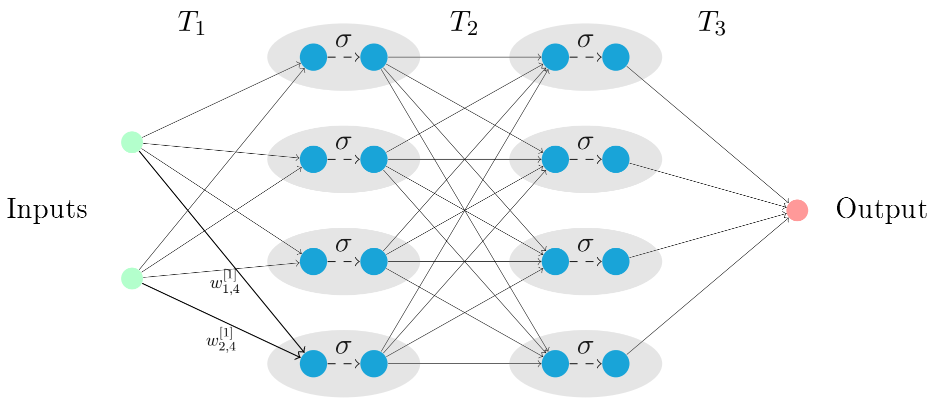

Figure 1.2: Fully-connected neural network with 3 layers, or with 2 hidden layers.

the computational cost would be prohibitively expensive. In consequence, the method that we study is a slight modification of the classic method which is called *Stochastic Gradient Descent* (SGD).

As we will see in Chapter 3, the underlying idea is to implement some randomness in the choice of elements of the training sample to be used in each step of the optimization. In this way, the evaluation of the error, and hence the update of the weights, does not have any order with respect to the training sample, which will be the key to avoid local minimums.

Introducing randomness in the backpropagation leads us to consider the framework of stochastic processes. In Chapter 4, we will present fundamental concepts about this theory and Itô's calculus. There, the Fokker-Planck equation will play an important role, since it will describe the probability density function of those stochastic processes.

Then, in Chapter 5, we will present a model of the Stochastic Gradient Descent in terms of stochastic differential equations given by Li, Tai and Weinan in 2017. The basic idea is to define a continuous-time stochastic model whose Euler discretization can be seen as an approximation of the SGD iteration rule. Consequently, we will describe what is considered as approximation in this problem and we will show a proof that the considered model satisfies those conditions.

Following the results of P. Chaudhari and S. Soatto in 2018, in the last chapter we will prove that SGD implicitly performs variational inference, as is often claimed informally in literature. Furthermore, we will show that SGD maximizes an entropic term which can be considered as an implicit regularization. This would explain why SGD has such generalization properties in practice.

Thus, in this thesis we will present the mathematical basis of deep neural network and its approximation and learning properties.

CHAPTER 2

Deep approximation

As it was announced in the introduction, the aim of this chapter is to show the approximation power of deep neural networks. With this purpose, we will show the benefits of choosing the Rectified Linear Unit as activation function and we will prove upper bounds for the approximation error given by estimates of the Vapnik-Chervonenkis dimension of networks. But first, let us present some basic concepts of learning complexity.

2.1. Learning complexity

Learning algorithms

A *network* N is a machine capable of taking on a number of “states”, each of which represents a function computable by the machine. They give functions from X to Y, that normally are $X \subset \mathbb{R}^d$ and $Y = \{0, 1\}$. Nevertheless, all the statements can be easily generalised to $Y = \mathbb{R}$.

Let Ω be the set of states, then N give us a function $F : \Omega \times X \to Y$. For any $\omega \in \Omega$, $h_\omega : X \to Y$ is the function *represented by state* ω, given by $h_\omega(x) = F(\omega, x)$.

The set of *functions computable by* N is $H = \{h_\omega : \omega \in \Omega\}$.

We consider a training sample of length m, $z = ((x_1, y_1), \ldots, (x_m, y_m)) \in (X \times Y)^m$ and we denoted by P a probability distribution on $X \times Y$. Then, given a function $h \in H$, the *error of h with respect to* P is defined as follows

$$\mathrm{er}_P(h) = P\{(x, y) \in X \times Y : h(x) \neq y\}.$$

Similarly, the *observed error* on the sample z is

$$\hat{\mathrm{er}}_z(h) = \frac{1}{m} |\{i : 1 \leq i \leq m \text{ and } h(x_i) \neq y_i\}|.$$

We define the *approximation error* of the class H as the minimum error that a function in H can give, that is,

$$\mathrm{opt}_P(H) = \inf_{h \in H} \mathrm{er}_P(h).$$

The aim of the learning process is, given $\epsilon > 0$, produce an $h \in H$ such that

$$\mathrm{er}_P(h) < \mathrm{opt}_P(H) + \epsilon.$$

In order to construct a more general model which can consider noise in the data, we formalize the concept of learning as a process that can satisfy this condition above with high probability for sufficiently large training samples.

Definition 2.1. Suppose that H is a class of functions that map from a set X to $\{0,1\}$. A *learning algorithm* L for H is a function

$$L : \bigcup_{m=1}^{\infty} (X \times Y)^m \to H$$

with the following property:

Given $\epsilon, \delta \in (0,1)$, there exists $m_0(\epsilon,\delta)$ such that if $m \geq m_0(\epsilon,\delta)$ then, for any probability distribution P, if z is a sample of length m drawn according to P^m, we have

$$\text{er}_P(L(z)) < \text{opt}_P(H) + \epsilon,$$

with probability $1-\delta$, that is

$$P^m\{z : \text{er}_P(L(z)) - \text{opt}_P(H) < \epsilon\} > 1 - \delta.$$

Equivalently, there exists a function $\epsilon_0(m,\delta)$ such that for all m, δ and P, with probability at least $1-\delta$ over $z \in (X \times Y)^m$ choosing accordingly with P^m,

$$\text{er}_P(L(z)) < \text{opt}_P(H) + \epsilon_0(m,\delta)$$

and for all $\delta \in (0,1)$, $\epsilon_0(m,\delta)$ approaches to zero as m tends to infinity.

Definition 2.2. We say that H is *learnable* if there is a learning algorithm for H.

One measure of efficiency of a learning algorithm is the minimum sample size $m_0(\epsilon,\delta)$ sufficient for (ϵ,δ)-learning.

Definition 2.3. We define the *sample complexity* function $m_L(\epsilon,\delta)$ of L as

$$m_L(\epsilon,\delta) = \min\{m : m \text{ is a sufficient sample size for } (\epsilon,\delta)\text{-learning H}\}$$

It is also useful to define the *inherent sample complexity* $m_H(\epsilon,\delta)$ of the learning problem for H:

$$m_H(\epsilon,\delta) = \min_L m_L(\epsilon,\delta),$$

where the minimum is taken over all learning algorithms for H.

Analogously, we define the *estimation error* $\epsilon_L(m,\delta)$ of L to be the smallest possible estimation error bound.

Vapnik-Chervonenkis dimension

Consider a finite subset $S \subset X$. For a class of functions H from X to $\{0,1\}$, the restriction of H to the set S is denoted by $H_{|S}$. If $|H_{|S}| = 2^{|S|}$ then H contains all the classifiers of S, therefore that is a good way to measure the classification complexity of H with respect to S.

Definition 2.4. The *growth function* of H, $\Pi_H : \mathbb{N} \to \mathbb{N}$, is defined as

$$\Pi_H(m) = \max\{|H_{|S}| : S \subset X \text{ and } |S| = m\}.$$

Notice that $\Pi_H(m) \leq 2^m$ for all m and if H is finite then $\Pi_H(m) \leq |H|$ with equality for a sufficiently large m. Thus, this function can be considered to be a refinement of the notion of cardinality that is applicable to infinite sets of functions.

Definition 2.5. Given a finite set $S \subset X$, we say that H *shatters* S if H contains all the dichotomies over S, that is, if $|H_{|_S}| = 2^{|S|}$.

Definition 2.6. (Vapnik-Chervonenkis dimension). We define the *VC-dimension* of H, a set of functions from X to $\{0,1\}$, as

$$\mathrm{VCdim}(H) = \max\{|S| : H \ \text{ shatters } \ S \subset X\},$$

or equivalently,

$$\mathrm{VCdim}(H) = \max\{m : \Pi_H(m) = 2^m\}.$$

If H is a class of real-valued functions, we set $\mathrm{VCdim}(H) :=\mathrm{VCdim}(\Theta(H))$, where

$$\Theta(H) = \{\Theta(h) : h \in H\}$$

and $\Theta(x) = 1$ if $x > 0$, $\Theta(h) = 0$ if $x \leq 0$.

The relation between VC-dimension and the sample complexity is very tight. For ϵ and δ small($<1/40$), we show this relation by the following inequalities from [Anthony-Barlett] for any learning algorithm L for H,

$$\frac{C_1}{\epsilon^2}\left(\mathrm{VCdim}(H) + \log\left(\frac{1}{\delta}\right)\right) \leq m_L(\epsilon,\delta) \leq \frac{C_2}{\epsilon^2}\left(\mathrm{VCdim}(H) + \log\left(\frac{1}{\delta}\right)\right) \tag{2.1}$$

Moreover, a direct relationship between VC-dimension and learnability is summarized by the following theorem of [Anthony-Barlett, Theorem 5.5].

Theorem 2.7. *For a class H of functions mapping from X to $\{0,1\}$, the following statements are equivalent:*

i) *H is learnable.*

ii) $m_H(\epsilon,\delta) = \mathcal{O}\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right)$

iii) *$VCdim(H) < +\infty$*

iv) *$\Pi_H(m)$ is bounded by a polynomial in m.*

A class of functions H of special interest is the one which ReLU networks can represent, as we will discuss in the next section. Here we recall a recent result by [Barlett et al.], which provides a sharp estimate on VC-dimension.

Theorem 2.8. *Let $VCdim(W,L)$ be defined as the largest VC-dimension of ReLU network with W weights and L layers. Then, the following inequality holds*

$$cWL\log(W/L) \leq VCdim(W,L) \leq CWL\log(W). \tag{2.2}$$

2.2. ReLU Calculus

In this section we focus on ReLU networks. We will show that they are equivalent to any network with a continuous piecewise linear activation function. However, this equivalence is not free of charge, we will have to increase the number of units, weights and layers, and we will try to bound this growth. But first, let us define ReLU networks.

Definition 2.9. For any number of layers L, input and output dimensions, n_0 and n_L respectively, a $\mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ ReLU network is given by a sequence of $L-1$ numbers $n_1, \dots, n_{L-1}$ representing widths of hidden layers, a set of $L-1$ affine transformations $T_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ and a linear transformation $T_L : \mathbb{R}^{n_{L-1}} \to \mathbb{R}^{n_L}$ corresponding to the weights of the hidden layers. This L-layer ReLU network represents the function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ defined as

$$f = T_L \circ \sigma \circ T_{L-1} \circ \cdots \circ T_2 \circ \sigma \circ T_1, \tag{2.3}$$

where $\sigma(x) = (\max\{0, x_1\}, \dots, \max\{0, x_n\})$. For a ReLU network, we defined its *size* as $n_1 + \cdots + n_{L-1}$, its *depth* as L and its *width* as $\max\{n_1, \dots, n_{L-1}\}$.

Let us show some basic results about ReLU networks that will be useful.

Lemma 2.10. *Let $f_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ be represented by a ReLU network with depth L_1 and size s_1, and let $f_2 : \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$ be represented by a ReLU network with depth L_2 and size s_2. Then, $f_2 \circ f_1$ can be represented by a ReLU network with depth $L_1 + L_2 - 1$ and size $s_1 + s_2$.*

Proof. The result follows from (2.3),

$$\begin{aligned} f_2 \circ f_1 &= (T^2_{L_2} \circ \sigma \circ T^2_{L_2-1} \circ \cdots \circ T^2_2 \circ \sigma \circ T^2_1) \circ (T^1_{L_1} \circ \sigma \circ T^1_{L_1-1} \circ \cdots \circ T^1_2 \circ \sigma \circ T^1_1) \\ &= T^2_{L_2} \circ \sigma \circ T^2_{L_2-1} \circ \cdots \circ T^2_2 \circ \sigma \circ T' \circ \sigma \circ T^1_{L_1-1} \circ \cdots \circ T^1_2 \circ \sigma \circ T^1_1, \end{aligned}$$

where $T' = T^2_1 \circ T^1_{L_1}$. ■

Lemma 2.11. *Let $f_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ be represented by a ReLU network with depth L and size s_1, and let $f_2 : \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$ be represented by a ReLU network with depth L and size s_2. Then, $f_1 + f_2$ can be represented by a ReLU network with depth L and size $s_1 + s_2$.*

Proof. We just put the two ReLUs in parallel with the same inputs and combine the outputs to get the sum.

■

Lemma 2.12. *Let $f_1, \cdots, f_m : \mathbb{R}^n \to \mathbb{R}$ be functions that can be represented by ReLU networks with depth L_i and size s_i, $i = 1, \dots, m$. Then, the function $f = \max\{f_1, \dots, f_m\}$ can be represented by a ReLU network of depth at most $\max\{L_1, \dots, L_m\} + \lceil \log(m) \rceil$ and at most size $s_1 + \cdots + s_m + 4(2m-1)$. The result is analogous to $f = \min\{f_1, \dots, f_m\}$.*

Proof. We prove it by induction on m. The case $m = 1$ is trivial.

For $m \geq 2$, consider $g_1 := \max\{f_1, \dots, f_{\lfloor \frac{m}{2} \rfloor}\}$ and $g_2 := \max\{f_{\lfloor \frac{m}{2} \rfloor + 1}, \dots, f_m\}$. By the induction hypothesis and since $\lfloor \frac{m}{2} \rfloor, \lceil \frac{m}{2} \rceil < m$ when $m \geq 2$, g_1 and g_2 can be represented by ReLU networks of depths at most $\max\{L_1, \dots L_{\lfloor \frac{m}{2} \rfloor}\} + \lceil \log(\lfloor \frac{m}{2} \rfloor) \rceil$ and $\max\{L_{\lfloor \frac{m}{2} \rfloor + 1}, \dots, L_m\} + \lceil \log(\lfloor \frac{m}{2} \rfloor) \rceil$ respectively.

Moreover, both networks will have at most size $s_1 + \cdots + s_{\lfloor \frac{m}{2} \rfloor} + 4(2\lfloor \frac{m}{2} \rfloor - 1)$ and $s_{\lfloor \frac{m}{2} \rfloor + 1} + \cdots + s_m + 4(2\lfloor \frac{m}{2} \rfloor - 1)$, respectively.

Therefore, the function $G : \mathbb{R}^n \to \mathbb{R}^2$ given by $G(x) = (g_1(x), g_2(x))$ can be implemented by a ReLU network with depth at most $\max\{L_1, \ldots, L_m\} + \lceil \log(\lceil \frac{m}{2} \rceil) \rceil$ and at most size $s_1 + \cdots + s_m + 4(2m-2)$.

Let us show know how to represent the function $T : \mathbb{R}^2 \to \mathbb{R}$ defined as $T(x, y) = \max\{x, y\} = \frac{x+y}{2} + \frac{|x-y|}{2}$ by a 2-layer ReLU with size 4.

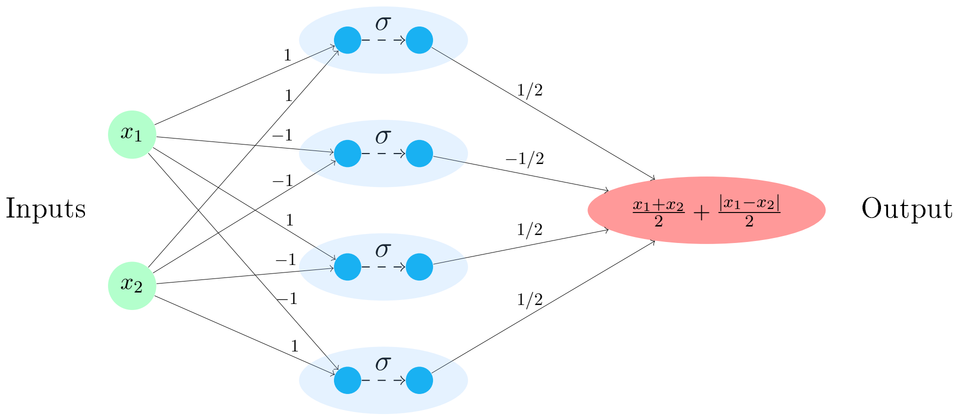

Figure 2.1: Maximum function represented by a ReLU network.

Finally, the result follows from the fact that $f = T \circ G$ and Lemma 2.10.

■

By [Yarotsky 2017], we have the following result involving approximation of continuous piecewise linear functions by ReLU networks.

Proposition 2.13. *Let $\tau : \mathbb{R} \to \mathbb{R}$ be any continuous piecewise linear function with $1 \leq M < \infty$ breakpoints. Let ξ be a network with the activation function τ, having length L, W weights and size s. Then there exists a ReLU network η that has depth L, not more than $(M+1)^2 W$ weights and size not larger than $(M+1)s$, that computes the same function as ξ.*

Proof. Let $a_1 < \ldots < a_M$ be the break points of τ. Notice that we can express τ via the ReLU function σ, as a linear combination

$$\tau(x) = c_0 \sigma(a_1 - x) + \sum_{m=1}^{M} c_m \sigma(x - a_m) + h$$

for some specific coefficients $\{c_i\}_{m=0}^{M}$ and h. We use this representation to rewrite the computation performed by a single τ-unit,

$$x_1, \ldots, x_N \mapsto \tau \left(\sum_{k=1}^{N} w_k x_k + b \right),$$

as a linear combination of $M+1$ σ-units,

$$x_1, \ldots, x_N \mapsto \begin{cases} \sigma\left(\sum_{k=1}^{N} w_k x_k + b - a_m\right), & m = 1, \ldots, M, \\ \sigma\left(a_1 - b - \sum_{k=1}^{N} w_k x_k\right), & m = 0. \end{cases}$$

Thus, we can replace one-by-one all the τ-units without incrementing the depth, just by increasing the size to $(M+1)s$ with at most $(M+1)^2 W$ connections.

■

Now, we consider this equivalence by increasing the depth. Let us prove a strong result about upper bound of ReLu network's depth given by [Arora et al.].

Theorem 2.14. *Every piecewise linear function $\mathbb{R}^n \to \mathbb{R}$ can be represented by a ReLU network with at most $\lceil \log(n+1) \rceil$ depth.*

Proof. First, notice that any continuous piecewise linear function can be represented as a linear combination of piecewise linear convex functions. More precisely, by [Wang-Sun], we have that any continuous piecewise linear function $f : \mathbb{R}^n \to \mathbb{R}$, there exists a finite set of affine linear functions $\ell_1, \dots, \ell_k$ and subsets of indexes $I_1, \dots, I_p \subseteq \{1, \dots, k\}$ (not necessarily disjoint) with cardinality at most $n+1$, such that

$$f = \sum_{j=1}^{p} s_j \left(\max_{i \in I_j} \ell_i \right),$$

where $s_j \in \{-1, +1\}$ for all $j = 1, \dots, p$. Observe that $\max_{i \in I_j} \ell_i$ is a piecewise linear convex function with at most $n+1$ pieces, since $|I_j| \leq n+1$. By Lemma 2.12, it holds that for each $\max_{i \in I_j} \ell_i$ there exists a ReLU network with depth at most $\lceil \log(n+1) \rceil$ which represents it. Furthermore, using Lemmas 2.10 and 2.11, it follows that neither the sum nor the composition of functions increases the depth of the ReLU network which represents each operation. Hence, we conclude that for any continuous piecewise linear function there exists a ReLU network with at most $\lceil \log(n+1) \rceil$.

■

2.3. Deep approximation

In the previous section we have shown the relevance of the ReLU activation function, in this section we focus on the approximation properties of ReLU networks. In particular, we are interested in the minimal requirements that a ReLU network must satisfy to achieve a desired performance. Namely, we will prove some lower bounds for the number of weights and layers that a ReLU network needs to approximate a function with error ϵ. The class of functions that we want to approximate is the unit ball of the Sobolev space $W^{n,\infty}([0,1]^d)$, which we denote by

$$F_{d,n} = \{u \in W^{n,\infty}([0,1]^d) : \|u\|_{W^{n,\infty}} \leq 1\}.$$

In this setup, the error is given in the norm of $W^{n,\infty}([0,1]^d)$ and some lower bounds on the network complexity can be obtained as a consequence of existing upper bounds on VC-dimension of networks. The first result that we present is [Yarotsky 2017, Theorem 4].

Theorem 2.15. *Fix d, n. For any $\epsilon \in (0,1)$, a ReLU network architecture capable of approximating any function $f \in F_{d,n}$ with error ϵ must have at least $c\epsilon^{-d/2n}$ weights, with some constant $c = c(d,n) > 0$.*

Proof. Given a positive integer N, let choose S a set of N^d points $\mathbf{x}_1, \dots, \mathbf{x}_{N^d}$ in the cube $[0,1]^d$ such that distance between any of them is not less than $1/N$. For any assignment of values $y_1, \dots, y_{N^d} \in \mathbb{R}$, we can define a smooth function f satisfying $f(\mathbf{x}_m) = y_m$ for every m by

$$f(\mathbf{x}) = \sum_{m=1}^{N^d} y_m \phi(N(\mathbf{x} - \mathbf{x}_m)),$$

where $\phi \in C_c^\infty(\mathbb{R}^d)$ is some cut-off function such that $\phi(\mathbf{0}) = 1$ and $\phi(\mathbf{x}) = 0$ if $|\mathbf{x}| > 1/2$. Notice that for any $\mathbf{x} \in \mathbb{R}^d$ the value $\phi(N(\mathbf{x}-\mathbf{x}_m))$ can be non-zero at most for a single $m \in \{1, \dots, N^d\}$, since $|\mathbf{x}_m - \mathbf{x}_k| \geq 1/N$ for every $m \neq k$.

In order to ensure that f belongs to $F_{d,n}$ we give the following condition. For any multi-index $\alpha = (\alpha_1, \dots, \alpha_d)$ with $|\alpha| \leq n$, it holds that

$$\max_{\mathbf{x}} |D^\alpha f(\mathbf{x})| \leq N^{|\alpha|} \max_m |y_m| \max_{\mathbf{x}} |D^\alpha \phi(\mathbf{x})|,$$

therefore, if

$$\max_m |y_m| \leq c_4 N^{-n}, \tag{2.4}$$

with $c_4 = \|\phi\|^{-1}_{W^{n,\infty}}$, then $\|f\|_{W^{n,\infty}} \leq 1$, that is $f \in F_{d,n}$.

Now, set

$$\epsilon = \frac{c_4}{3} N^{-n}. \tag{2.5}$$

Let suppose that there is a ReLU network architecture η that can approximate, by adjusting its weights, any $f \in F_{d,n}$ with error less than ϵ. We denote by $\eta(\mathbf{x}, \mathbf{w})$ the output of the network for the input $\mathbf{x}$ and the weights $\mathbf{w}$.

Consider any assignment $\mathbf{z}$ of Boolean values $z_1, \dots, z_{N^d} \in \{0, 1\}$. Set

$$y_m = z_m c_4 N^{-n},$$

and let f be given as before. Thus, condition (2.4) holds and hence $f \in F_{d,n}$. The objective is now to obtain a lower bound on the complexity of a network that can approximate such an f.

By assumption over η, there exists a vector of weights $\mathbf{w_z}$ such that for every $\mathbf{x} \in [0,1]^d$, $\eta(\mathbf{x}, \mathbf{w_z})$ approximates $f(\mathbf{x})$ with an error smaller than ϵ. In particular, for all m we have $|\eta(\mathbf{x}_m, \mathbf{w_z}) - y_m| \leq \epsilon$. Therefore, if $z_m = 1$, then

$$\eta(\mathbf{x}_m, \mathbf{w_z}) \geq c_4 N^{-n} - \epsilon = \frac{2}{3}\, c_4\, N^{-n} > \frac{c_4 N^{-n}}{2}.$$

On the other hand, if $z_m = 0$, then

$$\eta(\mathbf{x}_m, \mathbf{w_z}) \leq \epsilon < \frac{c_4 N^{-n}}{2}.$$

Hence, the threshold network $\eta_1 = \mathbb{I}(\eta > c_4 N^{-n}/2)$ has outputs

$$\eta_1(\mathbf{x}_m, \mathbf{w_z}) = z_m, \quad \forall m \in \{1, \dots, N^d\}.$$

Since the Boolean values z_m were arbitrary, we conclude that the subset S is shattered by the class of function given by the ReLU network architecture η_1. That is, any dichotomy of S can be represented by η_1 with some specific weights. Hence,

$$\mathrm{VCdim}(\eta_1) \geq N^d.$$

Expressing N through ϵ with (2.5), we obtain

$$\mathrm{VCdim}(\eta_1) \geq \left(\frac{3\epsilon}{c_4}\right)^{-d/n} \tag{2.6}$$

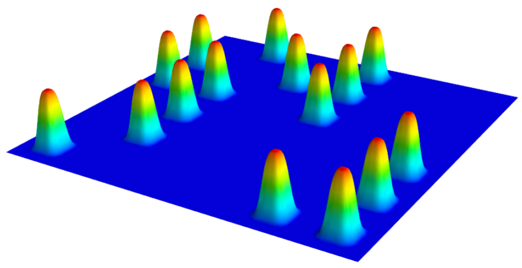

Figure 2.2: Function f considered in the proof for $d = 2$. Figure from [Yarotsky 2017].

In order to conclude the proof, we combine this inequality with the following upper bound of VCdim(η_1) given by [Anthony-Barlett, Theorem 8.6],

$$\mathrm{VCdim}(\eta_1) \leq c_3 W^2.$$

Notice that W is the number of weights, which is the same as in η since we do not consider the threshold parameter as a weight. By (2.6) it holds that

$$\left(\frac{3\epsilon}{c_4}\right)^{-d/n} \leq c_3 W^2,$$

or equivalently

$$W \geq c\epsilon^{-d/(2n)},$$

with $c = (c_3(3/c_4)^{d/n})^{-1/2}$, as we want to prove.

■

This Theorem states that a ReLU network with W weights generally cannot provide approximation in $\mathcal{F}_{n,d}$ with accuracy better than $O(W^{-2n/d})$, since

$$W \geq c\epsilon^{-d/2n} \Rightarrow CW^{-2n/d} \leq \epsilon.$$

The ReLU network architecture that is presented in Theorem 1 of [Yarotsky 2017] is able to approximate with error $O(W^{-n/d} \log^{n/d} W)$. Up to the logarithmic factor, this approximation reaches the optimal rate over all models under the assumption of continuous parameter selection.

However, this rate is quite far from the optimal rate that we have obtain in Theorem 2.15, since the gap between the powers $\frac{2n}{d}$ and $\frac{n}{d}$ is considerably large.

Therefore, the main matter now is how to bridge this gap of rates of approximation. We will present results that go in this direction, developed for $W^{1,\infty}([0,1]^d)$. The phase diagram for approximation rates that we will prove involves the modulus of continuity ω_f of the function f, which is defined as follows

$$\omega_f(r) = \max\{|f(\mathbf{x}) - f(\mathbf{y})| : \mathbf{x}, \mathbf{y} \in [0,1]^d, |\mathbf{x} - \mathbf{y}| \leq r\},$$

where $|\mathbf{x}|$ is the Euclidean norm.

Let $\tilde{f} : [0,1]^d \to \mathbb{R}$ be the approximation of a continuous function f by a network architecture η_W with d inputs and W weights. The relevant question is which powers $p \in \mathbb{R}$ can be achieved, by choosing the architecture and the weights for the following inequality:

$$\|f - \tilde{f}\|_\infty \leq a\omega_f(cW^{-p}), \quad \forall f \in C([0,1]^d), \tag{2.7}$$

with some constant a and c.

Notice that if the equality above holds for some p, then it also holds for any smaller p, since if $r < R$ we have that

$$\omega_f(r) \leq \omega_f(R).$$

A first answer to this question can be given in the $p = \frac{1}{d}$ phase in which the approximation can be obtained using a standard piecewise linear interpolation. This represents only a minor improvement on the previously mentioned $\mathcal{O}(W^{-n/2}\log^{n/2}(W))$ result.

Proposition 2.16. *There exists a network architecture η_W with W weights and, for each W, a weight assignment linear in f such that Equation (2.7) is satisfied with $p = \frac{1}{d}$. The network architecture can be chosen as $O(W)$ parallel blocks each having the same architecture that only depends on d.*

Proof. The proof follows from a construction of an approximating function of linear combination of spike function as in the proof of Theorem 2.15. For the details see [Yarotsky 2018].

We turn now to the region $p > \frac{1}{d}$. As we will see, the results concerning this region are given by the tight relation between VC dimension bounds and approximation theory that we have been showing along this section.

Theorem 2.17. *Let $f \in \mathcal{F}_{d,1}$.*

i) (Feasibility) Approximation rate (2.7) cannot be achieved with $p > \frac{2}{d}$.

ii) (Inherent depth) If approximation rate (2.7) is achieved with some $p \in (\frac{1}{d}, \frac{2}{d}]$ by an architecture η_W with W weights, then η_W must have depth $L \geq cW^{pd-1}/\log W$, with $c > 0$ possibly depending on d and p.

Proof. First, let us see that if the approximation rate (2.7) holds for some p, then all $f \in F_{d,1}$ can be approximated in $W^{1,\infty}$ by architectures η_W with accuracy

$$\epsilon_W = c_1 W^{-p}. \tag{2.8}$$

This holds since if we consider $x_0, y_0 \in [0,1]^d$ that maximize $|f(x_0) - f(y_0)|$ and satisfy $|x_0 - y_0| \leq cW^{-p}$, we get

$$\begin{aligned} a\omega_f(cW^{-p}) &= a|f(x_0) - f(y_0)| \\ &\leq aL_f|x_0 - y_0| \\ &\leq c_1 W^{-p}. \end{aligned}$$

Notice that c_1 does not depend on f despite L_f is its Lipschitz constant. This happens due to the fact that f is in $F_{d,1}$ and then $L_f \leq 1$.

Now, let us prove each part of the Theorem.

i) This statement is a consequence of Theorem 2.15, where it is proved that any architecture which approximates all $f \in F_{d,1}$ with accuracy ϵ_W must satisfy $W \geq c_2 \epsilon_W^{-d/2}$ for some constant c_2. Thus, (2.8) implies that

$$W \geq c_2 \left(c_1 W^{-p}\right)^{-d/2} \Rightarrow W \geq cW^{pd/2},$$

and hence

$$\frac{pd}{2} \leq 1 \Rightarrow p \leq \frac{2}{d}.$$

ii) The second part can be obtained by combining arguments of Theorem (2.7) with the recently established tight upper bound (2.2) for the VC dimension,

$$\mathrm{VCdim}(W, L) \leq CWL \log W, \tag{2.9}$$

where C is a global constant.

Suppose that η is a network architecture which can approximate all $f \in F_{d,1}$ with accuracy ϵ_W. Then, by inequality (2.6) we obtain

$$c_2 \epsilon_W^{-d} \leq \mathrm{VCdim}(\eta_W).$$

Now, using Equation (2.8) and (2.9), it holds that

$$c_3 W^{pd} \leq \mathrm{VCdim}(\eta_W) \leq CWL \log W,$$

and we conclude that

$$c_3 W^{pd} \leq CWL \log W \quad \Rightarrow \quad L \geq c \frac{W^{pd-1}}{\log W}.$$

■

Thus, we have proved that the best rate of approximation that we can achieve is $p = \frac{2}{d}$, so the natural question that arises is which architecture can obtain this rate. In [Yarotsky 2018], it is shown that deep neuronal networks with constant number of neurons per layer can reach this optimum rate. Namely, he is able to construct a network with constant width at least $2d + 10$, whose number of weights increases by rising the depth and weight assignment is discontinuous that satisfies this approximation property.

In practice, this results motivates the use of deep neuronal networks instead of shallow networks, since even if the Universal Approximation Theorem claims that shallow networks can approximates with an arbitrary error, the weight growth will be exponential and the convergence will be slower than deep networks.

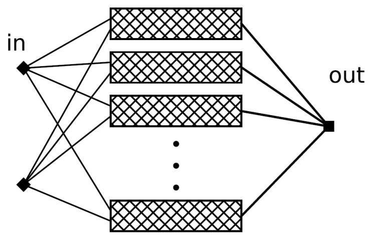

Figure 2.3: The parallel, constant depth network architecture implementing piecewise linear interpolation ensures approximation rate $p = \frac{1}{d}$. Figure from [Yarotsky 2018].

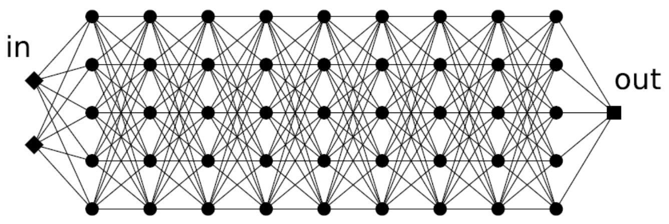

Figure 2.4: An example of narrow fully-connected network architecture with constant width. These architectures provides the optimal approximation rate $p = \frac{2}{d}$. Figure from [Yarotsky 2018].

CHAPTER 3

Supervised learning

Once we have seen the approximation ability of deep neural networks, let focus on the learning process. From now on, we will study how to choose the weights of a given architecture in order to get the underlying function. For instance, this desired function could be an image classifier which recognizes two different features in images such as if there appears a cat or a dog.

The aim is to let deep neural networks learn how to classify data from a given training sample. In our case, we will work in the framework of supervised learning where the training sample is already classified and hence it is possible to contrast the output of the network with the available data. Nevertheless, it should be noted that there exist other setups for deep learning problems, such as the unsupervised learning, where the main idea is to classify into groups or clusters the data whose characteristics are similar, according to an appropriate notion of similarity.

In this chapter we will present one of the most frequently used methods of weights assignments in supervised learning, which is called Stochastic Gradient Descent. It is a modification of the common Gradient Descent, where we introduce some randomness in order to avoid reaching local minimums and saddle points. Moreover, we will show explicitly how to update the weights by Backpropagation method. The key of this procedure is that the error is propagated from the output layer back to hidden layers, providing update rules to the weights of each layer.

First, let us recall how neural networks are constructed. Let us consider a neural network with L layers. For each layer $l \in \{1, \dots, L\}$, we use n_l to denote the number of neurons in it. So the network maps from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{n_L}$. We use $W^{[l]} \in \mathcal{M}_{n_l \times n_{l-1}}(\mathbb{R})$ to represent the matrix of *weights* at layer l. More precisely, $w_{jk}^{[l]}$ is the weight that neuron j at layer l applies to the output of neuron k at layer $l-1$. Similarly, $b^{[l]} \in \mathbb{R}^{n_l}$ is the vector of *biases* for layer l, therefore neuron j at layer l uses the bias $b_j^{[l]}$. Thus, the affine transformation $T_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ which maps from layer $l-1$ to layer l is given by

$$T_l(x) = W^{[l]}x + b^{[l]}.$$

As we have seen previously, after applying each affine transformation we use an activation function. We define the *activation function* as a non-linear function, $\sigma : \mathbb{R} \to \mathbb{R}$ similar to the step function which mimics the behaviour of a neuron in the brain. This means that it gives positive output if the input is large and 0 otherwise. Notice that this functions is useful to distinguish the layers due to the non-linearity, since if this was not the case we could consider deep networks as single-hidden-layer networks.

We use $a_j^{[l]}$ to denote the result of the activation function applied to the output of the neuron j in the layer l, what we will simply call *activation*. We also define $a^{[l]} := (a_1^{[l]}, \dots, a_{n_l}^{[l]})^T \in \mathbb{R}^{n_l}$ as the *activation vector* of the layer l. Hence, we can describe the feed-forward algorithm as follows.

Feed-forward algorithm. Giving an input $\mathbf{x} \in \mathbb{R}^{n_0}$, we may summarize how the network works.

$$\begin{aligned} a^{[0]} &= \mathbf{x} \in \mathbb{R}^{n_0} \\ a^{[l]} &= \sigma\left(W^{[l]} a^{[l-1]} + b^{[l]}\right) \in \mathbb{R}^{n_l}, \quad \text{for } \ l = 1, 2, \dots, L. \end{aligned}$$

This algorithm will feed the input $\mathbf{x}$ forward through the network in order to produce the output vector $a^{[L]} \in \mathbb{R}^{n_l}$.

The weights and biases that we have defined before are considered parameters which the network readjust after working at a training sample $\{(\mathbf{x}^{\{i\}}, \mathbf{y}^{\{i\}})\}_{i=1}^{N} \subset \mathbb{R}^{n_0} \times \mathbb{R}^{n_L}$. Therefore, we define an error function depending on weights and biases that we want to be minimized by the network. In order to show weights and biases as the variables of the error function it is convenient to consider these matrices and vectors as a unique vector $p \in \mathbb{R}^s$ which is composed of each component of them.

Definition 3.1. Given a training sample $\{\left(\mathbf{x}^{\{i\}}, \mathbf{y}^{\{i\}}\right)\}_{i=1}^{N} \subset \mathbb{R}^{n_1} \times \mathbb{R}^{n_L}$ and an architecture specified by $p = \{(W^{[l]}, b^{[l]})\}_{l=1}^{L} \in \mathbb{R}^s$, we define the *error function* $f : \mathbb{R}^s \to \mathbb{R}$ by

$$f(p) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \|\mathbf{y}^{\{i\}} - a^{[L]}(\mathbf{x}^{\{i\}})\|_2^2.$$

We also define f_i as the *error function for a single sample* $(\boldsymbol{x}^{\{i\}}, \boldsymbol{y}^{\{i\}})$ by

$$f_i(p) = \frac{1}{2} \|\mathbf{y}^{\{i\}} - a^{[L]}(\mathbf{x}^{\{i\}})\|_2^2.$$

Observe that the relationship between both definition is

$$f(p) = \frac{1}{N} \sum_{i=1}^{N} f_i(p).$$

The classical approach of supervised learning of neural networks is the minimization of this error function. Note that the architecture is fixed, and hence the only objective of such a minimization is the value of weights and biases.

3.1. Stochastic Gradient

The most common learning technique is based on the iteration given by the gradient descent method. This method which was first proposed by Cauchy in 1847 [Cauchy] and its description can be found on any introductory textbook on optimization, such as [Nesterov]. Here we will only briefly sketch the idea.

Let $f \in C^{1,1}(\mathbb{R}^s)$, so that $\exists L > 0$:

$$|f(y) - f(x) - < \nabla f(x), y - x > | \leq \frac{L}{2} |x - y|^2 \quad \forall x, y \in \mathbb{R}^s.$$

The method of gradient descent consist of evaluating first ∇f at a point $p \in \mathbb{R}^s$, and then performing the update

$$p \to p - \eta \nabla f(p), \tag{3.1}$$

where η is a real number called the *learning rate*. In more complex training methods this parameter can be dependent on the iteration, namely, the learning rate will be large in the first steps and it will be small when it approaches to the minimum, see [Nesterov, Section 1.2.3].

In our case, where we consider η fixed and sufficiently small, this iteration guarantees that

$$f(p - \eta \nabla f(\eta)) \leq f(p) - \eta \left(1 - \frac{\eta L}{2}\right) |\nabla f(p)|^2,$$

which corresponds to a decrease of f.

We would like to apply this method for the error function of the whole training sample, but normally we have a large number of parameters and a large number of training points, that makes the computation prohibitively expensive. A frequently used strategy to overcome this problem is the following.

Stochastic gradient method. A single step may be summarized as

1. Choose an integer $i \in \{1, \dots, N\}$ uniformly random.
2. Update
$$p \to p - \eta \nabla f_i(p).$$

This is the simplest form of these type of methods where one randomly chosen training point is used to represent the full training set. As the iteration proceeds, the method choose different points, so there is some hope that this reduction of cost-per-iteration will be worthwhile overall.

Stochastic gradient for mini-batch. As the method before tries to approximate the mean over all training points by a single sample, it is natural to consider instead a small sample average. For some $\mathscr{b} \ll N$ we could take steps as follows.

1. Choose $\mathscr{b}$ integers uniformly random, $k_1, \dots, k_{\mathscr{b}} \in \{1, \dots, N\}$.
2. Update
$$p \to p - \frac{\eta}{\mathscr{b}} \sum_{i=1}^{\mathscr{b}} \nabla f_i(p).$$

The purpose of Chapters 5 and 6 is to provide a quantitative analysis of this method.

3.2. Backpropagation

We are now interested in applying the stochastic gradient method in order to train the network. For this purpose we will present the backpropagation process which was announced by [Rumelhart et al.] in 1986 . The point is that we compute the chain rule of the gradient of the loss function in order to propagate the error from the output layer back to the hidden layers. Therefore, we switch from the general vector of parameters p to the entries in the weight matrixes and biases.

Namely, our goal is to compute the gradient of the error function with respect to each $w_{jk}^{[l]}$ and $b_j^{[l]}$. As the error function f is a linear combination of f_i for every training point and the partial derivatives are linear operators we just focus on computing the partial derivatives of f_i.

Hence, for a fixed training point we will make a slight abuse of notation and use f to represent f_i and simply write

$$f = \frac{1}{2}\|y - a^{[L]}\|_2^2.$$

We also define the *weighted input* for neuron j at layer l as follows

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]} \in \mathbb{R}^{n_l}, \quad \text{for } \ l = 1, 2, \ldots, L.$$

Notice that the relation between weighted inputs and the activation which propagates the information through the network is

$$a^{[l]} = \sigma\left(z^{[l]}\right), \quad \text{for } \ l = 1, 2, \ldots, L.$$

Definition 3.2. Let *local gradient*, $\delta^{[l]} \in \mathbb{R}^{n_l}$, be defined by

$$\delta_j^{[l]} = \frac{\partial f}{\partial z_j^{[l]}}, \quad \text{for } \ 1 \le j \le n_l \ \text{ and } \ 1 \le l \le L.$$

The importance of this object is that the error function can only be at a minimum if all components of $\delta^{[l]}$ are zero.

At this stage we also define the *Hadamard product* of two vectors. If $u, v \in \mathbb{R}^n$, then $u \circ v := (u_1v_1, \ldots, u_nv_n)^T \in \mathbb{R}^n$. Now, we conclude this chapter presenting the explicit computation of the elements of backpropagation given by [Higham].

Lemma 3.3. *Let σ be piecewise differentiable. With the above notation we have*

$$\delta^{[L]} = \sigma'\left(z^{[L]}\right) \circ \left(a^{[L]} - y\right), \tag{3.2}$$

$$\delta^{[l]} = \sigma'\left(z^{[l]}\right) \circ \left(W^{[l+1]}\right)^T \delta^{[l+1]}, \quad \textit{for } \ 1 \le l \le L-1, \tag{3.3}$$

$$\frac{\partial f}{\partial b_j^{[l]}} = \delta_j^{[l]}, \quad \textit{for } \ 1 \le l \le L, \tag{3.4}$$

$$\frac{\partial f}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} a_k^{[l-1]}, \quad \textit{for } \ 1 \le l \le L. \tag{3.5}$$

Proof. We begin by proving (3.2). Notice that by the chain rule we obtain

$$\delta_j^{[L]} = \frac{\partial f}{\partial z_j^{[L]}} = \frac{\partial f}{\partial a_j^{[L]}} \frac{\partial a_j^{[L]}}{\partial z_j^{[L]}}, \quad \text{for } \ 1 \le j \le n_l.$$

As we write before we know that $a^{[L]} = \sigma\left(z^{[L]}\right)$, so

$$\frac{\partial a_j^{[L]}}{\partial z_j^{[L]}} = \sigma'\left(z_j^{[L]}\right).$$

On the other hand, we can compute the partial derivative of the error function by its definition,

$$\frac{\partial f}{\partial a_j^{[L]}} = \frac{\partial}{\partial a_j^{[L]}}\left(\frac{1}{2}\|y - a^{[L]}\|_2^2\right) = -(y_j - a_j^{[L]}).$$

Hence,

$$\delta_j^{[L]} = \left(a_j^{[L]} - y_j\right) \sigma'\left(z_j^{[L]}\right),$$

which is the componentwise form of (3.2).

To prove (3.3), recall that $z_k^{[l+1]}$ and $z_j^{[l]}$ are related by

$$z_k^{[l+1]} = \sum_{s=1}^{n_l} w_{ks}^{[l+1]} \sigma\left(z_s^{[l]}\right) + b_k^{[l+1]}.$$

Therefore,

$$\frac{\partial z_k^{[l+1]}}{\partial z_j^{[l]}} = w_{kj}^{[l+1]} \sigma'\left(z_j^{[l]}\right).$$

Now, we can apply these results to compute the local gradient of a hidden layer l,

$$\delta_j^{[l]} = \frac{\partial f}{\partial z_j^{[l]}} = \sum_{k=1}^{n_{l+1}} \frac{\partial f}{\partial z_k^{[l+1]}} \frac{\partial z_k^{[l+1]}}{\partial z_j^{[l]}} = \sum_{k=1}^{n_{l+1}} \delta_k^{[l+1]} w_{kj}^{[l+1]} \sigma'\left(z_j^{[l]}\right) = \sigma'\left(z_j^{[l]}\right) \left((W^{[l+1]})^T \delta^{[l+1]}\right)_j.$$

In order to prove 3.4, we remark how are $z_j^{[l]}$ and $b_j^{[l]}$ connected,

$$z_j^{[l]} = \left(W^{[l]} \sigma\left(z^{[l-1]}\right)\right)_j + b_j^{[l]}.$$

Since $z^{[l-1]}$ does not depend on $b_j^{[l]}$, we obtain

$$\frac{\partial f}{\partial b_j^{[l]}} = \frac{\partial f}{\partial z_j^{[l]}} \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} = \delta_j^{[l]}.$$

Finally, to show (3.5), we start with the componentwise version of the definition of $z^{[l]}$,

$$z_j^{[l]} = \sum_{k=1}^{n_{l-1}} w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]},$$

which gives

$$\frac{\partial z_j^{[l]}}{\partial w_{jk}^{[l]}} = a_k^{[l-1]} \quad \text{and} \quad \frac{\partial z_s^{[l]}}{\partial w_{jk}^{[l]}} = 0, \quad \text{for } s \neq j.$$

It holds these results because the jth neuron at layer l uses the weights from only the jth row of $W^{[l]}$, and applies them linearly. Then,

$$\frac{\partial f}{\partial w_{jk}^{[l]}} = \sum_{s=1}^{n_l} \frac{\partial f}{\partial z_s^{[l]}} \frac{\partial z_s^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial f}{\partial z_j^{[l]}} \frac{\partial z_j^{[l]}}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} a_k^{[l-1]}.$$

■

Hence, the learning scheme can then be summarized by the following algorithm:

1. Initialize W and b.
2. Choose $x \in X$ at random.
3. Compute $\frac{\partial f}{\partial w_{ij}}$ and $\frac{\partial f}{b_j}$ using backpropagation Lemma 3.3.
4. Update W and b by gradient descent (3.1).
5. Loop steps 2., 3. and 4.

Observe that this algorithm can be easily modified for the mini-batch. However, convergence of the scheme above is not granted, and depends not only on the architecture but also on the training data. In terms of concepts introduced in Chapter 2, the problem may not even be learnable.

Moreover, even when some kind of convergence is reached, the learned parameters W and b (and the quality of the result) may depend on the initialization. Therefore, in practice, it is often the case that the learning algorithm is performed several times and either a selection or a combination of the results is kept.

CHAPTER 4

Stochastic Differential Equations

Before we start studying the behaviour of SGD, it is required to present some basic concepts about stochastic processes. Namely, we will follow [Evans] to show some results of Itô's calculus and to see how they can be used in stochastic differential equations. We will assume some basic preliminary notions of stochastic calculus such as that of Itô integral. The main objective of this chapter is to provide the connection between the probability density functions of the solutions to those equations and the Fokker-Planck equation, following [Schuss].

Let us start by considering a probability space $(\Omega, \mathcal{U}, P)$ and the 1-dimensional Brownian motion $W(\cdot)$ defined on it. As usual, given a random variable X, we denote by $\mathcal{U}(X)$ the smallest sub-σ-algebra of $\mathcal{U}$ with respect to which X is measurable. Recall also that the Brownian motion is a collection of random variables $\{W(t) | t \geq 0\}$ such that

i) $W(0) = 0$,

ii) $W(t) - W(s) \sim N(0, t-s) \quad \forall t \geq s \geq 0$,

iii) for all $0 < t_1 < \ldots < t_n$ the random variables

$$W(t_1), W(t_2) - W(t_1), \ldots, W(t_n) - W(t_{n-1})$$

are independents.

Then, we present the following definitions:

Definition 4.1. i) The σ-algebra $\mathcal{W}(t) = \mathcal{U}(W(s) | 0 \leq s \leq t)$ is called the *history* of the Brownian motion up to time t.

ii) The σ-algebra $\mathcal{W}^+(t) = \mathcal{U}(W(s) - W(t) | s \geq t)$ is called the *future* of the Brownian motion beyond time t.

Definition 4.2. A family $\mathcal{F}(\cdot)$ of σ-algebras contained in $\mathcal{U}$ is called a *filtration* (or nonanticipating with respect to $W(\cdot)$) if

i) $\mathcal{F}(t) \supseteq \mathcal{F}(s)$ for all $t \geq s \geq 0$.

ii) $\mathcal{F}(t) \supseteq \mathcal{W}(t)$ for all $t \geq 0$.

iii) $\mathcal{F}(t)$ is independent of $\mathcal{W}^+(t)$ for all $t \geq 0$.

We could think of $\mathcal{F}(t)$ as the σ-algebra which contains all information available to us at time t.

From this definition, we will say that a real-valued stochastic process $G(\cdot)$ is *adapted* if for each time $t \geq 0$, $G(t)$ is $\mathcal{F}(t)$-measurable with respect to a filtration $\mathcal{F}$. The idea is that for each $t \geq 0$, the random variable $G(t)$ depends on the information available in $\mathcal{F}(t)$ uniquely.

Actually, we will need a stronger condition, namely that $G(\cdot)$ be *progressively measurable*. This means that for every time t, the map $[0,t] \times \Omega \to \mathbb{R}$ defined by $(s,\omega) \to X(s,\omega)$ is $\mathcal{B}(\mathbb{R}) \otimes \mathcal{F}(t)$-measurable. This implies $G(\cdot)$ is $\mathcal{F}(t)$-adapted.

Definition 4.3. i) We denote by $\mathbb{L}^2(0,T)$ the space of all real-valued and progressively measurable stochastic processes $G(\cdot)$ such that

$$\mathbb{E}\left(\int_0^T G^2 dt\right) < \infty.$$

Notice that this condition means that the variance of the random variable $\int_0^T G dW$ is finite, as we will see below, in the Lemma 4.5.

ii) On the other hand, we denote by $\mathbb{L}^1(0,T)$ the space of all real-valued and progressively measurable stochastic process $G(\cdot)$ such that

$$\mathbb{E}\left(\int_0^T |G| dt\right) < \infty.$$

In order to achieve some properties about the integrals of process in $\mathbb{L}^2(0,T)$, we will prove them for step processes in that space and then we will use approximation just as it is done for the Lebesgue integral.

Definition 4.4. A process $G \in \mathbb{L}^2(0,T)$ is called a *step process* if there exists a partition $\{0 = t_0 < t_1 < \ldots < t_m = T\}$ and a family $\{G_k\}_{k=0}^{m-1}$ of random variables such that

$$G(t) \equiv G_k \quad \text{for} \quad t_k \leq t < t_{k+1}, \qquad \text{where} \quad k = 0, 1, \ldots, m-1.$$

The main feature that makes this definition useful is that Itô integral of a step process G on the interval $(0,T)$ is given by

$$\int_0^T G dW = \sum_{k=0}^{m-1} G_k(W(t_{k+1}) - W(t_k)). \tag{4.1}$$

Lemma 4.5. *Let $G \in \mathbb{L}^2(0,T)$ be an adapted step process. Then:*

(i)

$$\mathbb{E}\left(\int_0^T G dW\right) = 0.$$

(ii)

$$\mathbb{E}\left(\left(\int_0^T G dW\right)^2\right) = \mathbb{E}\left(\int_0^T G^2 dt\right).$$

Proof. *(i)* By 4.1 we have

$$\mathbb{E}\left(\int_0^T G dW\right) = \sum_{k=0}^{m-1} \mathbb{E}\left(G_k(W(t_{k+1}) - W(t_k))\right).$$

Recall that G_k is $\mathcal{F}(t_k)$-measurable and $\mathcal{F}(t_k)$ is independent of $\mathcal{W}^+(t_k)$. Since $W(t_{k+1})-W(t_k)$ is $\mathcal{W}^+(t_k)$-measurable, we obtain that G_k and $W(t_{k+1})-W(t_k)$ are independent. Therefore,

$$\mathbb{E}\left(G_k(W(t_{k+1})-W(t_k))\right)=\mathbb{E}(G_k)\mathbb{E}(W(t_{k+1})-W(t_k))=0,$$

due to the fact that $\mathbb{E}(W(t_{k+1})-W(t_k))=0$ by definition of the Brownian motion.

(ii) Again using (4.1) we have

$$\mathbb{E}\left(\left(\int_0^T Gdw\right)^2\right)=\sum_{k,j=1}^{m-1}\mathbb{E}(G_kG_j(W(t_{k+1})-W(t_k))(W(t_{j+1})-W(t_j))).$$

Let us first see that the terms with $j\neq k$ vanish. Without lost of generality, we can assume $j<k$. Then, by definition of filtration, it holds,

$$\mathcal{F}(t_k)\supseteq\mathcal{F}(t_j)\quad\text{and}\quad\mathcal{W}^+(t_k)\ \text{independent of}\ \mathcal{F}(t_k).\tag{4.2}$$

Since G is adapted, then G_kG_j is $F(t_k)$-measurable, so it is independent of $W(t_{k+1})-W(t_k)$, which is $\mathcal{W}^+(t_k)$-measurable. Also, by the independence of the increments of the Brownian motion, $W(t_{k+1})-W(t_k)$ is independent of $W(t_j+1)-W(t_j)$.

This implies, in particular, that

$$\mathbb{E}(G_kG_j(W(t_{k+1})-W(t_k))(W(t_{j+1})-W(t_j)))=\mathbb{E}(G_kG_j(W(t_{j+1})-W(t_j)))\mathbb{E}(W(t_{k+1})-W(t_k)).$$

Thus, since $G\in\mathbb{L}^2(0,T)$, we use Cauchy-Schwarz inequality to obtain

$$\begin{aligned}&|\mathbb{E}(G_kG_j(W(t_{k+1})-W(t_k))(W(t_{j+1})-W(t_j)))|\\&=|\mathbb{E}(G_kG_j(W(t_{j+1})-W(t_j)))|\,|\mathbb{E}(W(t_{k+1})-W(t_k))|\\&\leq\mathbb{E}(G_k^2)^{1/2}\,\mathbb{E}(G_j^2(W(t_{j+1})-W(t_j))^2)^{1/2}\,|\mathbb{E}(W(t_{k+1})-W(t_k))|\\&=\underbrace{\mathbb{E}(G_k^2)^{1/2}}_{<\infty}\underbrace{\mathbb{E}(G_j^2)^{1/2}}_{<\infty}\underbrace{(\mathbb{E}(W(t_{j+1})-W(t_j))^2)^{1/2}}_{=(Var(W(t_{j+1})-W(t_j)))^{1/2}<\infty}\underbrace{|\mathbb{E}(W(t_{k+1})-W(t_k))|}_{=0}=0,\end{aligned}$$

where we have used that G_j and $W(t_{j+1})-W(t_j)$ are independent , since $\mathcal{F}(t_j)$ and $\mathcal{W}^+(t_j)$ are.

Hence, keeping only the diagonal terms in the sum, we get

$$\begin{aligned}\mathbb{E}\left(\left(\int_0^T Gdw\right)^2\right)&=\sum_{j=0}^{m-1}\mathbb{E}(G_j^2(W(t_{j+1})-W(t_j))^2)\\&=\sum_{j=0}^{m-1}\mathbb{E}(G_j^2)\mathbb{E}((W(t_{j+1})-W(t_j))^2)\\&=\sum_{j=0}^{m-1}\mathbb{E}(G_j^2)(t_{j+1}-t_j)\\&=\mathbb{E}\left(\int_0^T G^2dt\right),\end{aligned}$$

where the second identity is the variance of the Brownian motion.

■

As we have said before, we can extend those properties to every process in $\mathbb{L}^2(0,T)$ by the following lemma.

Lemma 4.6. (Approximation by step processes). *If $G \in \mathbb{L}^2(0,T)$, there exists a sequence of bounded step processes $G^n \in \mathbb{L}^2(0,T)$ such that*

$$\mathbb{E}\left(\int_0^T |G - G^n|^2 dt\right) \to 0.$$

4.1. Itô's formula

The aim of this section is to introduce a fundamental result for SDE, known as Itô's formula.

Definition 4.7. (i) Let $\mathbf{W}(\cdot) = (W^1(\cdot), \dots, W^m(\cdot))$ be the *m-dimensional Brownian motion.*

(ii) An $\mathbb{M}^{n\times m}$-valued stochastic process $\mathbf{B} = ((b^{ij}))$ belongs to $\mathbb{L}^2_{n\times m}(0,T)$ if

$$b^{ij} \in \mathbb{L}^2(0,T) \quad \text{with} \quad i = 1,\dots,n; \quad j = 1,\dots,m.$$

(iii) An $\mathbb{R}^n$-valued stochastic process $\mathbf{a} = (a^1,\dots,a^n)$ belongs to $\mathbb{L}^1_n(0,T)$ if

$$a^i \in \mathbb{L}^1(0,T) \quad \text{where} \quad i = 1,\dots,n$$

Definition 4.8. If $\mathbf{B} \in \mathbb{L}^2_{n\times m}(0,T)$, then

$$\int_0^T \mathbf{B}\, d\mathbf{W}$$

is an $\mathbb{R}^n$-valued random variable, whose i-th component is

$$\sum_{j=1}^m \int_0 b^{ij}\, dW^j \quad \text{where} \quad i = 1,\dots,n.$$

We observe at this point that, by approximation with step processes as in Lemma 4.6, by Lemma 4.5 we obtain the following result which will be useful in the next section.

Lemma 4.9. *If $\mathbf{B} \in \mathbb{L}^2_{n\times m}(0,T)$, then*

$$\mathbb{E}\left(\int_0^T \mathbf{B}\, d\mathbf{W}\right) = 0,$$

and

$$\mathbb{E}\left(\left|\int_0^T \mathbf{B}\, d\mathbf{W}\right|^2\right) = \mathbb{E}\left(\int_0^T |\mathbf{B}|^2 dt\right),$$

where $|\mathbf{B}|^2 = \sum_{i=1}^m \sum_{j=1}^n |b^{ij}|^2$.

In order to introduce Itô's formula, we recall the definition of stochastic differential.

Definition 4.10. If $\mathbf{X}(\cdot) = (X^1(\cdot),\dots,X^n(\cdot))$ is an $\mathbb{R}^n$-valued stochastic process such that

$$\mathbf{X}(r) = \mathbf{X}(s) + \int_s^r \mathbf{a}\, dt + \int_s^r \mathbf{B}\, d\mathbf{W}, \tag{4.3}$$

for some $\mathbf{a} \in \mathbb{L}^1_n(0,T)$, $\mathbf{B} \in \mathbb{L}^2_{n\times m}(0,T)$ and all $0 \le s \le r \le T$. Then, we say $\mathbf{X}(\cdot)$ has the *stochastic differential*

$$d\mathbf{X} = \mathbf{a}\, dt + \mathbf{B}\, d\mathbf{W}, \quad \mathbf{X}(s) = x. \tag{4.4}$$

Equivalently, each component of X satisfies the SDE

$$dX^i = a^i\, dt + \sum_{j=1}^m b^{ij}\, dW^j \quad \text{for} \quad i = 1,\dots,n.$$

Theorem 4.11. (Itô's formula). *Let $\mathbf{X}$ be an $\mathbb{R}^n$-valued stochastic process satisfying* (4.4). *Let $f : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ be continuous , with continuous partial derivations $\frac{\partial f}{\partial t}$, $\frac{\partial f}{\partial x_i}$, $\frac{\partial^2 f}{\partial x_i \partial x_j}$, when $i,j = 1,\dots,n$. Then*

$$d(f(\mathbf{X}(t),t)) = \frac{\partial f}{\partial t}dt + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} dX^i + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j} \sum_{k=1}^{m} b^{ik} b^{jk}\, dt.$$

For a proof of this theorem we refer to [Evans]. An useful formulation of this result can be obtained by introducing explicitly (4.7) and isloating the Brownian motion. To this end, define

$$\sigma^{ij}(\mathbf{X}(t),t) = \frac{1}{2}\sum_{k=1}^{m} b^{ik}(\mathbf{X}(t),t) b^{jk}(\mathbf{X}(t),t),$$

as the components of the *diffusion matrix*

$$\sigma(\mathbf{X}(t),t) = \frac{1}{2}\mathbf{B}(\mathbf{X}(t),t)\, \mathbf{B}^T(\mathbf{X}(t),t), \tag{4.5}$$

and set the *backward Kolmogorov operator*

$$L^* f(\mathbf{X}(t),t) = \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma^{ij}(\mathbf{X}(t),t) \frac{\partial^2 f(\mathbf{X}(t),t)}{\partial x_i \partial x_j} + \sum_{i=1}^{n} a^i(\mathbf{X}(t),t) \frac{\partial f(\mathbf{X}(t),t)}{\partial x_i}. \tag{4.6}$$

Corollary 4.12. *With the same hypothesis stated in the previous theorem, the Itô's formula can be expanded as*

$$\begin{aligned} d(f(\mathbf{X}(t),t)) &= \left[\frac{\partial f}{\partial t} + L^* f\right] dt + \sum_{i=1}^{n}\sum_{j=1}^{m} b^{ij}\, \frac{\partial f}{\partial x_i} dW^j \\ &= \left[\frac{\partial f}{\partial t} + \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma^{ij} \frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{i=1}^{n} a^i \frac{\partial f}{\partial x_i}\right] dt + \sum_{i=1}^{n}\sum_{j=1}^{m} b^{ij}\, \frac{\partial f}{\partial x_i} dW^j. \end{aligned}$$

4.2. The Fokker-Planck equation

We are now interested in finding the probability density function of the solution to a special case of (4.4). Namely, let $\mathbf{a} \in \mathbb{L}^1(\mathbb{R}^n \times [0,T], \mathbb{R}^n)$, $\mathbf{B} \in \mathbb{L}^2(\mathbb{R}^n \times [0,T], \mathbb{R}^{n,m})$, $x \in \mathbb{R}^n$ and consider the stochastic process $\mathbf{Y}$ satisfying

$$\begin{cases} d\mathbf{Y} = \mathbf{a}(\mathbf{Y},t)dt + \mathbf{B}(\mathbf{Y},t)d\mathbf{W}, & t > s \\ \mathbf{Y}(s) = x \end{cases} \tag{4.7}$$

Denote by $p(y,t|x,s)$ its probability density. This function satisfies two different PDEs, one with respect to the forward variables (y,t) and one with respect to the backward variables (x,s). The former is called the *Fokker-Planck equation* as can be seen in [Schuss, Section 4.5], and it will be discussed in this section.

Definition 4.13. (The Fokker-Planck operator). Let σ be as in (4.5). The partial differential operator L given by

$$L\, p = \sum_{i=1}^{n} \frac{\partial}{\partial y_i}\left[\sum_{j=1}^{n} \frac{\partial(\sigma^{ij}(\mathbf{Y},t)\, p)}{\partial y_j} - a^i(\mathbf{Y},t)p\right] \tag{4.8}$$

is called the Fokker-Planck operator for the SDE (4.7). Equivalently, we can rewrite this operator in its divergence form, which is the form that we will use in Chapter 6,

$$L\,p = \nabla \cdot (\nabla \cdot (\sigma\, p) - a\, p).$$

Note that this operator L is the adjoint of the operator L^* defined in (4.6), with respect to the $L^2(\mathbb{R}^n)$ inner product $(\cdot,\cdot)_{L^2}$. This means that for every $f, g \in C_c^\infty(\mathbb{R}^n)$ it holds that

$$(Lf, g)_{L^2} = (f, L^* g)_{L^2}.$$

Theorem 4.14. (The Fokker-Planck equation, FPE). *Let* $\mathbf{Y}$ *be an* $\mathbb{R}^n$*-valued stochastic process satisfying* (4.7). *Then, the probability density function* $p(y,t|x,s)$ *is the fundamental solution to the following problem.*

$$\text{(FPE)} \qquad \begin{cases} \partial_t p(y,t|x,s) = Lp(y,t|x,s) \quad \text{for } x, y \in \mathbb{R}^n, \quad t > s \\ \\ \lim\limits_{t\to s} p(y,t|x,s) = \delta(x-y). \end{cases}$$

Proof. Let $f \in C_c^\infty(\mathbb{R}^n)$. By integrating the Itô's formula for $f(\mathbf{Y}(t))$, we get

$$\begin{aligned} f(\mathbf{Y}(t)) - f(\mathbf{Y}(s)) &= \int_s^t d(f(\mathbf{Y}(\tau))) \\ &= \int_s^t \left[\sum_{i=1}^n a^i(\mathbf{Y}(\tau),\tau) \frac{\partial f(\mathbf{Y}(\tau))}{\partial y_i} + \sum_{i=1}^n \sum_{j=1}^n \sigma^{ij}(\mathbf{Y}(\tau),\tau) \frac{\partial^2 f(\mathbf{Y}(\tau))}{\partial y_i \partial y_j} \right] d\tau \\ &\quad + \sum_{j=1}^m \int_s^t \left[\sum_{i=1}^n b^{ij}(\mathbf{Y}(\tau),\tau)\, \frac{\partial f(\mathbf{Y}(\tau))}{\partial y_i} \right] dW^j(\tau). \end{aligned}$$

Now we compute the expectation of $f(\mathbf{Y}(\tau))$ conditioned to $\mathbf{Y}(s) = x$ in order to have an equation with respect to $p(y,t|x,s)$. Notice that by Lemma 4.9, we obtain that the expectation of the second addend vanishes:

$$\begin{aligned} &\mathbb{E}\left\{ \sum_{j=1}^m \int_s^t \left[\sum_{i=1}^n b^{ij}(\mathbf{Y}(\tau),\tau)\, \frac{\partial f(\mathbf{Y}(\tau))}{\partial y_i} \right] dW^j(\tau) \right\} \\ &\quad = \sum_{j=1}^m \mathbb{E}\left\{ \int_s^t \left[\sum_{i=1}^n b^{ij}(\mathbf{Y}(\tau),\tau)\, \frac{\partial f(\mathbf{Y}(\tau))}{\partial y_i} \right] dW^j(\tau) \right\} = 0. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbb{E}\left[f(\mathbf{Y}(t)) | \mathbf{Y}(s) = x \right] = f(x) + \mathbb{E}\Bigg\{ \int_s^t \Bigg[&\sum_{i=1}^n a^i(\mathbf{Y}(\tau),\tau) \frac{\partial f(\mathbf{Y}(\tau))}{\partial y_i} \\ &+ \sum_{i=1}^n \sum_{j=1}^n \sigma^{ij}(\mathbf{Y}(\tau),\tau) \frac{\partial^2 f(\mathbf{Y}(\tau))}{\partial y_i \partial y_j} \Bigg] d\tau \mid \mathbf{Y}(s) = x \Bigg\}. \end{aligned}$$

Which means that

(4.9)

$$\begin{aligned}
&\int_{\mathbb{R}^n} f(y)p(y,\tau|x,s)dy \\
&= f(x) + \int_\Omega \int_s^t \left[\sum_{i=1}^n a^i(\mathbf{Y}(\omega),\tau)\frac{\partial f(\mathbf{Y}(\omega))}{\partial y_i} + \sum_{i=1}^n\sum_{j=1}^n \sigma^{ij}(\mathbf{Y}(\omega),\tau)\frac{\partial^2 f(\mathbf{Y}(\omega))}{\partial y_i \partial y_j} \right] d\tau\, dP(\omega) \\
&= f(x) + \int_s^t \int_\Omega \left[\sum_{i=1}^n a^i(\mathbf{Y}(\omega),\tau)\frac{\partial f(\mathbf{Y}(\omega))}{\partial y_i} + \sum_{i=1}^n\sum_{j=1}^n \sigma^{ij}(\mathbf{Y}(\omega),\tau)\frac{\partial^2 f(\mathbf{Y}(\omega))}{\partial y_i \partial y_j} \right] dP(\omega) d\tau \\
&= f(x) + \int_s^t \int_{\mathbb{R}^n} \left[\sum_{i=1}^n a^i(y,\tau)\frac{\partial f(y)}{\partial y_i} + \sum_{i=1}^n\sum_{j=1}^n \sigma^{ij}(y,\tau)\frac{\partial^2 f(y)}{\partial y_i \partial y_j} \right] p(y,\tau|x,s)dyd\tau.
\end{aligned}$$

Equation (4.9) is the weak form of the PDE that the probabilistic density function satisfies because we have considered that equation for all test function in $C_c^\infty(\mathbb{R}^n)$.

In order to obtain the PDE for $p(y,t|x,s)$, we integrate by parts in eq. (4.9) and we change the order of integration by Fubini.

$$\begin{aligned}
\int f\, p\, dy &= f(x) + \int_s^t \left(\int \sum_{i=1}^n a^i\, p\, \frac{\partial f(y)}{\partial y_i} + \sum_{i=1}^n\sum_{j=1}^n \sigma^{ij}\, p\, \frac{\partial^2 f(y)}{\partial y_i \partial y_j} dy \right) d\tau \\
&= f(x) + \int_s^t \left(\int -\sum_{i=1}^n \frac{\partial (a^i\, p)}{\partial y_i} f(y) + \sum_{i=1}^n\sum_{j=1}^n \frac{\partial^2 (\sigma^{ij}\, p)}{\partial y_i \partial y_j} f(y) dy \right) d\tau \\
&= \int f(y)\delta(x-y)dy + \int f(y) \left(\int_s^t -\sum_{i=1}^n \frac{\partial (a^i\, p)}{\partial y_i} + \sum_{i=1}^n\sum_{j=1}^n \frac{\partial^2 (\sigma^{ij}\, p)}{\partial y_i \partial y_j} d\tau \right) dy \\
&= \int f(y)\delta(x-y)dy + \int f(y) \left(\int_s^t \sum_{i=1}^n \frac{\partial}{\partial y_i} \left[\sum_{j=1}^n \frac{\partial^2 (\sigma^{ij}\, p)}{\partial y_i \partial y_j} - a^i\, p \right] d\tau \right) dy.
\end{aligned}$$

Thus, in the sense of distributions, we have obtained the following equation

$$p(y,t|x,s) = \delta(x-y) + \int_s^t \sum_{i=1}^n \frac{\partial}{\partial y_i} \left[\sum_{j=1}^n \frac{\partial^2 (\sigma^{ij}(y,\tau)\, p(y,\tau|x,s))}{\partial y_i \partial y_j} - a^i(y,\tau)\, p(y,\tau|x,s) \right] d\tau.$$

$$= \delta(x-y) + \int_s^t L\, p(y,\tau|x,s)d\tau, \tag{4.10}$$

which is the equation for the fundamental solution (FPE).

■

For the previous theorem we have considered a formal derivation of a parabolic equation without assuming any regularity on the coefficients. The classical theory for **a** and **B** Hölder continuous can be found in [Friedman]. Moreover, the notion of fundamental solution and its relation with the (distributional) initial problem considered here, also commonly called fundamental solution, for parabolic equations, can be found in [Friedman, Chapter 1, Section 7] and it is frequently called Duhamel's principle.

CHAPTER 5

Continuous-time SGD

Since we have presented the basis of stochastic processes in Chapter 5, we can start now using these tools to study how SGD behaves. The aim of this chapter is to show that the SGD learning iteration can be approximated by a stochastic process characterized by a particular Stochastic Differential Equation. Then, following the inverse path of the so-called Euler discretization to obtain a continuous-time SGD, we will be able to analyse this learning process by means of studying the associated Fokker-Planck Equation. This approach was introduced in [Li et al.].

As observed in Chapter 3, Stochastic Gradient Descent tries to minimize a loss function

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{5.1}$$

where, for neural networks, $x \in \mathbb{R}^d$ are the weights and each $f_i : \mathbb{R}^d \to \mathbb{R}$ for $i = 1, \dots, n$ is the loss function associated to the ith sample (n is the size of the whole training sample). We shall call f the total loss function.

Solving (5.1) using the standard *Gradient Descent* (GD), given by

$$x_{k+1} = x_k - \eta\nabla f(x_k),$$

requires prohibitively expensive computations when $n \gg 1$. Therefore, we try the alternative method of *Stochastic Gradient Descent*, where we replace the full gradient ∇f by a gradient of a loss function of a randomly chosen sample i. In its simplest form, the SGD iteration is given by

$$x_{k+1} = x_k - \eta\nabla f_{\gamma_k}(x_k), \tag{5.2}$$

where $k \geq 0$ represents the iteration of the learning process and $\{\gamma_k\}$ are i.i.d. uniform random variables taking values in $\{1, \dots, n\}$. Unlike GD, the computational complexity of SGD per iteration is independent of n.

Let us rewrite the SGD iteration rule (5.2) as follows,

$$x_{k+1} = x_k - \eta\nabla f(x_k) + \sqrt{\eta}V_k, \tag{5.3}$$

where $V_k = V_k(x_k) = \sqrt{\eta}(\nabla f(x_k) - \nabla f_{\gamma_k}(x_k))$ is a d-dimensional random vector.

We want to approximate (5.3) with a simpler stochastic evolution of the form

$$X_{k+1} = X_k - \eta b(X_k) + \sqrt{\eta}\sigma(X_k)Z_k, \tag{5.4}$$

where $\{Z_k\}$ are i.i.d. random variables with distribution $N_d(0, 1)$ and $\sigma(X_k)$ is the covariance matrix of $V_k(X_k)$. Observe that for $\eta = \Delta t$, (5.4) is the Euler discretization of following SDE

$$dX_t = b(X_t)dt + \sigma(X_t)dW_t, \quad X_0 = x_0. \tag{5.5}$$

In the next section, we will then provide conditions on b that make 5.5 an approximation of (5.2) with the identification $t = k\eta$.

5.1. Stochastic approximation

It is now important to discuss which is the meaning of "approximation". Following [Li et al.], we will consider an approximation in the weak sense below.

Definition 5.1. Let $0 < \eta < 1$, $T > 0$ and set $N = \lfloor T/\eta \rfloor$. Let us define the set of *functions with polynomial growth*,

$$G := \{g : \mathbb{R}^d \to \mathbb{R} \mid \exists K, k > 0 \quad \text{s.t.} \quad |g(x)| < K(1 + |x|^k) \quad \forall x \in \mathbb{R}^d\}.$$

Then, we say that the SDE (5.5) is an *order α weak approximation* to the SGD (5.2) if for every $g \in G$, there exists $C > 0$, independent of η, such that for all $k = 0, 1, \ldots, N$,

$$\left|\mathbb{E}\left[g(X_{k\eta})\right] - \mathbb{E}\left[g(x_k)\right]\right| < C\eta^\alpha.$$

Notice that we are considering SDE to be a weak approximation of SGD just by studying the expectation of certain class of test functions applied to them. This is a standard definition in numerical analysis of SDEs, since weak approximation are close to the original process not in terms of individual sample paths, but their distributions.

Now, we show two theorems of approximation associated with what we will call *Stochastic Modified Equations* (SME) for the SGD iterations. This theorem from [Li et al., Theorem 1] allows us to use the SME to deduce distributional properties of the SGD, but with an important difference with respect to usual convergence studies. Namely, it describes dynamical behaviour and is derived without convexity assumptions on f or f_i.

Theorem 5.2. (Stochastic modified equations). *Let $0 < \eta < 1$, $T > 0$ and set $N = \lfloor T/\eta \rfloor$. Let $x_k \in \mathbb{R}$, $0 \le k \le N$ denote a sequence of SGD iterations defined by (5.2). Define $X_t \in \mathbb{R}^d$ as the stochastic process satisfying the SDE*

$$\text{(SME)} \qquad \begin{cases} dX_t = -\nabla f(X_t)dt + \sqrt{\eta D(X_t)}dW_t, \\ X_0 = x_0, \end{cases}$$

where $D(x) = \frac{1}{n}\sum_{i=1}^n (\nabla f(x) - \nabla f_i(x))(\nabla f(x) - \nabla f_i(x))^T$. Fix some test function $g \in G$ as before. Suppose also that the following conditions are met:

(i) ∇f, ∇f_i satisfying a Lipschitz condition: there exists $L > 0$ such that

$$|\nabla f(x) - \nabla f(y)| + \sum_{i=1}^n |\nabla f_i(x) - \nabla f_i(y)| \le L|x - y| \quad \forall x, y \in \mathbb{R}^d.$$

(ii) f, f_i and its partial derivatives up to order 7 belong to G.

(iii) ∇f, ∇f_i satisfy a growth condition: there exists $M > 0$ such that

$$|\nabla f(x)| + \sum_{i=1}^n |\nabla f_i(x)| \le M(1 + |x|) \quad \forall x \in \mathbb{R}^d.$$

(iv) g and its partial derivatives up to order 6 belong to G.

Then, the equation (SME) is an order 1 weak approximation of the SGD iterations.

It can be also constructed a SME which is higher order weak approximation of the SGD, as we can see in the following theorem.

Theorem 5.3. *Let consider the same conditions that we assume in the Theorem 5.2. Let define the following stochastic process $X_t \in \mathbb{R}^d$ given by the SDE*

$$\begin{cases} dX_t = -\nabla(f(X_t) + \frac{1}{4}\eta|\nabla f(X_t)|^2)dt + (\eta D(X_t))^{1/2}dW_t, \\ X_0 = x_0, \end{cases} \tag{5.6}$$

where $D(x) = \frac{1}{n}\sum_{i=1}^n(\nabla f(x) - \nabla f_i(x))(\nabla f(x) - \nabla f_i(x))^T$, as before. Then, we have that (5.6) is an order 2 weak approximation of the SGD iterations.

5.2. Stochastic approximation - Proofs

In order to prove these theorems we start by showing that a one-step approximation has order 3 error. After that we will use the general result given by Milstein (1986), to show that the overall global error is of order $\alpha = 1$ and $\alpha = 2$ for Theorem 5.2 and Theorem 5.3, respectively.

In the following proofs we will make repeated use of Taylor expansions in powers of η. To simplify presentation, we introduce a modified use for the notation $\mathcal{O}(\eta^\alpha)$, which will means that there exists a function $K(x) \in G$ such that the error terms are bounded by $K(x)\eta^\alpha$. For example, we write

$$b(x+\eta) = b_0(x) + \eta b_1(x) + \mathcal{O}(\eta^2),$$

to mean that there exists $K \in G$ such that

$$|b(x+\eta) - b_0(x) - \eta b_1(x)| \leq K(x)\eta^2.$$

Since the noise we are trying to model is small, we may assume from the outset that $b(x) = \mathcal{O}(1)$ but $\sigma(x) = \mathcal{O}(\eta^{1/2})$. For brevity, we will simply denote the noise term of the SDE by $\eta^{1/2}\sigma$. We will denote by ∂_i the partial derivative with respect to the i-th component.

First step to define the approximating SDE (5.4) is to compute expectation and covariance of V_k.

Lemma 5.4. *Let V_k be defined as before. Then, $\mathbb{E}[V_k] = 0$ and*

$$Var[V_k] = \eta D(x_k),$$

where $D(x) := \frac{1}{n}\sum_{i=1}^n(\nabla f(x) - \nabla f_i(x))(\nabla f(x) - \nabla f_i(x))^T$.

Proof. First, we calculate the expectation of V_k.

$$\begin{aligned} \mathbb{E}(V_k) &= \sqrt{\eta}\,\nabla f(x_k) - \sqrt{\eta}\,\mathbb{E}(\nabla f_{\gamma_k}(x_k)) = \sqrt{\eta}\,\nabla f(x_k) - \sqrt{\eta}\sum_{i=1}^n P(\gamma_k = i)\nabla f_i(x_k) \\ &= \sqrt{\eta}\,\nabla f(x_k) - \sqrt{\eta}\frac{1}{n}\sum_{i=1}^n \nabla f_i(x_k) = 0. \end{aligned}$$

In order to know the covariance matrix of V_k, denoted by $\mathrm{Var}[V_k]$, let compute the generic term $\mathrm{Var}[V_k]_{l,j}$. Notice that,

$$\mathrm{Cov}\left[V_k^l, V_k^j\right] = \mathbb{E}\left[(V_k^l - \mathbb{E}(V_k^l))(V_k^j - \mathbb{E}(V_k^j))\right] = \mathbb{E}\left[V_k^l V_k^j\right].$$

Therefore, simplifying notation $f = f(x_k)$ we obtain

$$\begin{aligned}
\mathrm{Cov}\left[V_k^l, V_k^j\right] &= \mathbb{E}\Big[\sqrt{\eta}(\partial_l f - \partial_l f_{\gamma_k})\sqrt{\eta}(\partial_j f - \partial_j f_{\gamma_k})\Big] \\
&= \eta\mathbb{E}\Big[\partial_l f \partial_j f - \partial_l f \partial_j f_{\gamma_k} - \partial_l f_{\gamma_k}\partial_j f + \partial_l f_{\gamma_k}\partial_j f_{\gamma_k}\Big] \\
&= \eta\Big[\partial_l f \partial_j f - \partial_l f \mathbb{E}(\partial_j f_{\gamma_k}) - \mathbb{E}(\partial_l f_{\gamma_k})\partial_j f + \mathbb{E}(\partial_l f_{\gamma_k}\partial_j f_{\gamma_k})\Big] \\
&= \eta\left[\partial_l f \partial_j f - \partial_l f \frac{1}{n}\sum_{i=1}^n \partial_j f_i - \frac{1}{n}\sum_{i=1}^n \partial_l f_i \partial_j f + \frac{1}{n}\sum_{i=1}^n \partial_l f_i \partial_j f_i\right] \\
&= \eta\frac{1}{n}\sum_{i=1}^n [\partial_l f \partial_j f - \partial_l f \partial_j f_i - \partial_l f_i \partial_j f + \partial_l f_i \partial_j f_i] \\
&= \eta\frac{1}{n}\sum_{i=1}^n (\partial_l f - \partial_l f_i)(\partial_j f - \partial_j f_i).
\end{aligned}$$

Hence, we conclude that the matrix expression for the covariance is

$$\mathrm{Var}\left[V_k\right] = \eta\frac{1}{n}\sum_{i=1}^n (\nabla f(x_k) - \nabla f_i(x_k))(\nabla f(x_k) - \nabla f_i(x_k))^T = \eta D(x_k).$$

Moreover, let us remark that $D(x)$ is a positive defined matrix, since $\eta D(x)$ is a covariance matrix and $\eta > 0$.

■

Next, we will need a lemma regarding a general fact about moments of SDEs with small noise, see [Li et al., Lemma 1].

Lemma 5.5. *Let $0 < \eta < 1$. Consider a stochastic process X_t, $t \geq 0$ satisfying the SDE*

$$dX_t = b(X_t)dt + \eta^{1/2}\sigma(X_t)dW_t, \tag{5.7}$$

with $X_0 = x \in \mathbb{R}^d$ and b, σ together with their derivatives belong to G. Define the one-step difference $\Delta = X_\eta - x$, then we have

(i) $\mathbb{E}[\Delta_i] = b_i(x)\eta + \frac{1}{2}\left(\sum_{j=1}^d b_j(x)\partial_j b_i(x)\right) + \mathcal{O}(\eta^3)$.

(ii) $\mathbb{E}\left[\Delta_i\Delta_j\right] = \left(b_i(x)b_j(x) + \sigma\sigma_{ij}^T\right)\eta^2 + \mathcal{O}(\eta^3)$.

(iii) $\mathbb{E}\left[\prod_{j=1}^s \Delta_{i_j}\right] = \mathcal{O}(\eta^3)$ *for all $s \geq 3$, $i_j = 1, \ldots, d$.*

All functions above are evaluated at x.

An equivalent result holds for one SGD iteration, for which we give a full proof.

Lemma 5.6. *Let $0 < \eta < 1$. Consider x_k, $k \geq 0$ satisfying the SGD iterations*

$$x_{k+1} = x_k - \eta \nabla f_{\gamma_k}(x_k), \tag{5.8}$$

with $x_0 = x \in \mathbb{R}^d$. Define the one-step difference $\bar{\Delta} = x_1 - x = -\eta \nabla f_{\gamma_0}(x)$, then we have

(i) $\mathbb{E}[\bar{\Delta}_i] = -(\partial_i f(x))\eta$

(ii) $\mathbb{E}[\bar{\Delta}_i \bar{\Delta}_j] = (\partial_i f(x) \partial_j f(x))\, \eta^2 + D_{ij}(x)\eta^2$.

(iii) $\mathbb{E}\left[\prod_{j=1}^{s} \bar{\Delta}_{i_j}\right] = \mathcal{O}(\eta^3)$ *for all* $s \geq 3$, $i_j = 1, \dots, d$.

where $D(x) = \frac{1}{n}\sum_{i=1}^{n} (\nabla f(x) - \nabla f_i(x))(\nabla f(x) - \nabla f_i(x))^T$.

Proof. By (5.8), we have that $\Delta_i = -\eta\, \partial_i f_{\gamma_0}(x)$ which implies (i).

In order to prove (ii), we recall $V := V_0 = \sqrt{\eta}(\nabla f - \nabla f_{\gamma_0})$ defined in (5.3). It is clear that $\bar{\Delta} = \sqrt{\eta} V - \eta \nabla f$, then by Lemma (5.4) and using that ∇f is deterministic we obtain that

$$\mathrm{Var}[\bar{\Delta}] = \mathrm{Var}[\sqrt{\eta} V - \eta \nabla f] = \eta\, \mathrm{Var}[V] = \eta^2 D.$$

We also know that

$$\mathrm{Var}\big[\bar{\Delta}\big] = \mathbb{E}\big[\bar{\Delta}\bar{\Delta}^T\big] - \mathbb{E}\big[\bar{\Delta}\big]\mathbb{E}\big[\bar{\Delta}\big]^T.$$

Therefore, by (i) we conclude

$$\mathbb{E}\big[\bar{\Delta}\bar{\Delta}^T\big] = \eta^2 (\nabla f)(\nabla f)^T + \eta^2 D,$$

which implies (ii).

Finally, we prove (iii) for $s = 3$:

$$\mathbb{E}\left[\bar{\Delta}_i \bar{\Delta}_j \bar{\Delta}_k\right] = \mathbb{E}\left[-\eta^3 \partial_i f_{\gamma_0} \partial_j f_{\gamma_0} \partial_k f_{\gamma_0}\right] = \eta^3 \mathbb{E}\left[-\partial_i f_{\gamma_0} \partial_j f_{\gamma_0} \partial_k f_{\gamma_0}\right] = \mathcal{O}(\eta^3),$$

where the last equality holds because $f_{\gamma_0}(x)$ is not related to η and the expectation of its derivatives belongs to G. If $s > 3$, by a similar argument we get higher orders in η.

■

Now, we will need a key result linking one-step approximations to global approximations given by [Milstein, Theorem 2, Lemma 5]. We reproduce the theorem tailored to our problem.

Theorem 5.7. (Milstein). *Let α be a positive integer and let the assumptions in Theorem 5.2 hold. Assume, in addition, that there exist $K_1, K_2 \in G$ so that*

$$\left|\mathbb{E}\Big[\prod_{j=1}^{s} \Delta_{i_j}\Big] - \mathbb{E}\Big[\prod_{j=1}^{s} \bar{\Delta}_{i_j}\Big]\right| \leq K_1(x)\eta^{\alpha+1},$$

for $s = 1, 2, \dots, 2\alpha + 1$ and

$$\mathbb{E}\Big[\prod_{j=1}^{2\alpha+2} |\bar{\Delta}_{i_j}|\Big] \leq K_2(x)\eta^{\alpha+1}.$$

Then, there exists a constant C such that for all $k = 0, 1, \dots, N$ and all $g \in G$ we have

$$\big|\mathbb{E}[g(X_{k\eta})] - \mathbb{E}[g(x_k)]\big| \leq C\eta^{\alpha}$$

Thus, we have all the tools to prove Theorem 5.2.

Proof of Theorem 5.2. We want to show that the hypotheses of Theorem 5.7 are satisfied by (SME). We just compute $\mathbb{E}[\prod_{j=1}^{s}\Delta_{i_j}]$ and $\mathbb{E}[\prod_{j=1}^{s}\bar{\Delta}_{i_j}]$ using Lemma 5.5 and Lemma 5.6 and compare their difference.

- Case $s=1$:

$$\mathbb{E}[\Delta_i] = -\partial_i f\eta + \frac{1}{2}\eta^2\sum_{j=1}^{d}\partial_j f\partial_j\partial_i f + \mathcal{O}(\eta^3)$$
$$\mathbb{E}[\bar{\Delta}_i] = -\partial_i f\eta.$$

  Therefore,

$$\mathbb{E}[\Delta_i] - \mathbb{E}[\bar{\Delta}_i] = \mathcal{O}(\eta^2)$$

- Case $s=2$:

$$\mathbb{E}[\Delta_i\Delta_j] = \eta^2[\partial_i f\partial_j f + D_{ij}] + \mathcal{O}(\eta^3)$$
$$\mathbb{E}[\bar{\Delta}_i\bar{\Delta}_j] = \eta^2[\partial_i f\partial_j f + D_{ij}]$$

  Therefore,

$$\mathbb{E}[\Delta_i\Delta_j] - \mathbb{E}[\bar{\Delta}_i\bar{\Delta}_j] = \mathcal{O}(\eta^3).$$

  Since $0<\eta<1$ by assumption, we obtain that

$$\mathbb{E}[\Delta_i\Delta_j] - \mathbb{E}[\bar{\Delta}_i\bar{\Delta}_j] = \mathcal{O}(\eta^2).$$

- Case $s\geq 3$:
  As we know that $\mathbb{E}[\prod_{j=1}^{s}\Delta_{i_j}] = \mathcal{O}(\eta^3)$ and $\mathbb{E}[\prod_{j=1}^{s}\bar{\Delta}_{i_j}] = \mathcal{O}(\eta^3)$, it holds that $\mathbb{E}[\prod_{j=1}^{s}\Delta_{i_j}] - \mathbb{E}[\prod_{j=1}^{s}\bar{\Delta}_{i_j}] = \mathcal{O}(\eta^3)$. Thus,

$$\mathbb{E}[\prod_{j=1}^{s}\Delta_{i_j}] - \mathbb{E}[\prod_{j=1}^{s}\bar{\Delta}_{i_j}] = \mathcal{O}(\eta^2).$$

- The last hypothesis that we have to ensure is the case $2\alpha+2=4$:

$$\mathbb{E}[\prod_{j=1}^{4}|\bar{\Delta}_{i_j}|] = \mathbb{E}[\prod_{j=1}^{4}\eta|\partial_{i_j}f_{\gamma_0}|] = \eta^4\mathbb{E}[\prod_{j=1}^{4}|\partial_{i_j}f_{\gamma_0}|] = \mathcal{O}(\eta^4)$$

  Therefore,

$$\mathbb{E}[\prod_{j=1}^{4}|\bar{\Delta}_{i_j}|] = \mathcal{O}(\eta^2)$$

Hence, we can apply Theorem 5.7with $\alpha = 1$ to conclude.

■

Proof of Theorem 5.3. We will follow the same steps as before. But first, we make some calculations in order to simplify the proof.

Recall that in this Theorem we consider $b(x) = -\nabla(f(x) + \frac{1}{4}|\nabla f(x)|^2)$, so

$$b_i = -\partial_i f - \frac{1}{4}\eta\partial_i(|\nabla f|^2) = -\partial_i f - \frac{1}{2}\eta\sum_{k=1}^{d}\partial_k f\partial_i\partial_k f$$

$$\partial_j b_i = -\partial_j\partial_i f - \frac{1}{2}\eta\sum_{k=1}^{d}(\partial_j\partial_k f\partial_i\partial_k f + \partial_k f\partial_j\partial_i\partial_k f)\,.$$

Now, let us show that conditions of Theorem 5.7 are satisfies by using Lemma 5.5.

- Case $s = 1$:

$$\begin{aligned}\mathbb{E}[\Delta_i] &= -\eta\partial_i f - \frac{1}{2}\eta^2\sum_{k=1}^{d}\partial_k f\partial_i\partial_k f + \frac{1}{2}\eta^2\sum_{j=1}^{d}[b_j\partial_j b_i] + \mathcal{O}(\eta^3)\\ &= -\eta\partial_i f - \frac{1}{2}\eta^2\sum_{k=1}^{d}\partial_k f\partial_i\partial_k f + \frac{1}{2}\eta^2\sum_{j=1}^{d}\partial_j f\partial_i\partial_j f + \mathcal{O}(\eta^3)\\ &= -\eta\partial_i f + \mathcal{O}(\eta^3),\end{aligned}$$

  since $\frac{1}{2}\eta^2\sum_{j=1}^{d}[b_j\partial_j b_i] = \frac{1}{2}\eta^2\sum_{j=1}^{d}[(-\partial_j f)(-\partial_j\partial_i f)] + \mathcal{O}(\eta^3)$.

  Therefore,

$$\mathbb{E}[\Delta_i] - \mathbb{E}[\bar{\Delta}_i] = \mathcal{O}(\eta^3).$$

- Case $s = 2$:

$$\mathbb{E}[\Delta_i\Delta_j] = \eta^2\,[b_i b_j + D_{ij}] + \mathcal{O}(\eta^3) = \eta^2\,[\partial_i f\partial_j f + D_{ij}] + \mathcal{O}(\eta^3),$$

  since $\eta^2 b_i b_j = \eta^2\partial_i f\partial_j f + \mathcal{O}(\eta^3)$.

  Then,

$$\mathbb{E}[\Delta_i\Delta_j] - \mathbb{E}[\bar{\Delta}_i\bar{\Delta}_j] = \mathcal{O}(\eta^3).$$

- Case $s \geq 3$: We obtain directly from the lemmas that

$$\mathbb{E}[\prod_{j=1}^{s}\Delta_{i_j}] - \mathbb{E}[\prod_{j=1}^{s}\bar{\Delta}_{i_j}] = \mathcal{O}(\eta^3).$$

- The last hypothesis that we have to check is the case $2\alpha + 2 = 6$:

$$\mathbb{E}[\prod_{j=1}^{6}|\bar{\Delta}_{i_j}|] = \mathbb{E}[\prod_{j=1}^{6}\eta|\partial_{i_j}f_{\gamma_0}|] = \eta^6\,\mathbb{E}[\prod_{j=1}^{6}|\partial_{i_j}f_{\gamma_0}|] = \mathcal{O}(\eta^6)$$

  Therefore,

$$\mathbb{E}[\prod_{j=1}^{6}|\bar{\Delta}_{i_j}|] = \mathcal{O}(\eta^3).$$

Hence, we conclude that (5.6) is an order 2 weak approximation of SGD iterations by applying Theorem 5.7.

■

5.3. SGD for mini-batch

In this section, we will generalize the results of this chapter in SGD for mini-batch. Recall from section 3.1 that iterations of this learning process are described as

$$x_{k+1} = x_k - \nabla f_{\mathcal{B}}(x_k), \tag{5.9}$$

where $\nabla f_{\mathcal{B}}(x_k)$ is the average gradient over a mini-batch $\mathcal{B}$,

$$\nabla f_{\mathcal{B}}(x) = \frac{1}{\mathcal{B}} \sum_{k \in \mathcal{B}} \nabla f_{\gamma_k}(x).$$

It should be pointed out that we use notation $\mathcal{B}$ for both the randomly chosen set of examples in a mini-batch and its size. As a way to know more about this object, let us compute its expectation and covariance matrix.

Lemma 5.8. *Let $\nabla f_{\mathcal{B}}(x)$ be defined as before. Then, $\mathbb{E}[\nabla f_{\mathcal{B}}(x)] = \nabla f(x)$ and*

$$\mathit{Var}[\nabla f_{\mathcal{B}}(x)] = \frac{D(x)}{\mathcal{B}},$$

with $D(x) = \frac{1}{n}\sum_{i=1}^{n}(\nabla f(x) - \nabla f_i(x))(\nabla f(x) - \nabla f_i(x))^T$, as in Lemma 5.4.

Proof. First, notice that

$$\mathbb{E}[\nabla f_{\mathcal{B}}] = \mathbb{E}\left[\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}} \nabla f_{\gamma_k}\right] = \frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}} \mathbb{E}\left[\nabla f_{\gamma_k}\right] = \frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\sum_{i=1}^{n}\frac{1}{n}\nabla f_i = \frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\nabla f = \nabla f.$$

On the other hand,

$$\mathrm{Var}[\nabla f_{\mathcal{B}}] = \mathrm{Var}\left[\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\nabla f_{\gamma_k}\right] = \mathbb{E}\left[\left(\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\nabla f_{\gamma_k} - \nabla f\right)\left(\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\nabla f_{\gamma_k} - \nabla f\right)^T\right].$$

Since for $k \neq j$ we have that γ_k and γ_j are independent, it holds that

$$\mathrm{Cov}[\nabla f_{\gamma_k}, \nabla f_{\gamma_j}] = \mathbb{E}[\nabla f_{\gamma_k}\nabla f_{\gamma_j}^T] - \nabla f \nabla f^T = \mathbb{E}[\nabla f_{\gamma_k}]\mathbb{E}[\nabla f_{\gamma_j}^T] - \nabla f\nabla f^T = 0.$$

Thus, we can apply the following formula

$$\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + \mathrm{Cov}[X,Y] + \mathrm{Cov}[Y,X],$$

in order to obtain

$$\mathrm{Var}\left[\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\nabla f_{\gamma_k}\right] = \frac{1}{\mathcal{B}^2}\sum_{k\in\mathcal{B}}\mathrm{Var}\left[\nabla f_{\gamma_k}\right].$$

By Lemma 5.4, we have that

$$\eta D = \mathrm{Var}\left[\sqrt{\eta}\,(\nabla f - \nabla f_{\gamma_k})\right] = \eta \mathrm{Var}[\nabla f_{\gamma_k}],$$

which implies

$$\mathrm{Var}[\nabla f_{\gamma_k}] = D.$$

Hence, we conclude that

$$\mathrm{Var}[\nabla f_{\mathcal{B}}] = \mathrm{Var}\left[\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\nabla f_{\gamma_k}\right] = \frac{1}{\mathcal{B}^2}\sum_{k\in\mathcal{B}}\mathrm{Var}\left[\nabla f_{\gamma_k}\right] = \frac{D}{\mathcal{B}}.$$

∎

Now, we can follow the same steps as in Theorem 5.2 in order to obtain a continuous-time representation of the SGD for mini-batch.

Corollary 5.9. *Let $0 < \eta < 1$, $T > 0$ and set $N = \lfloor T/\eta \rfloor$. Let $x_k \in \mathbb{R}$, $0 \leq k \leq N$ denote a sequence of SGD for mini-batch iterations defined by (5.9). Define $X_t \in \mathbb{R}^d$ as the stochastic process satisfying the SDE*

$$\begin{cases} dX_t = -\nabla f(X_t)dt + \sqrt{\frac{\eta}{\mathscr{b}} D(X_t)}dW_t, \\ X_0 = x_0. \end{cases} \tag{5.10}$$

Then, assuming the same conditions of the Theorem 5.2, it holds that (5.10) is an order 1 weak approximation of the SGD for mini-batch iteration.

Proof. We can rewrite the iteration process (5.9) as

$$x_{k+1} = x_k - \eta \nabla f(x_k) + \sqrt{\eta}\tilde{V}_k,$$

where $\tilde{V}_k = \sqrt{\eta}\,(\nabla f(x_k) - \nabla f_{\mathscr{b}}(x_k))$.

Notice that $\mathbb{E}[\tilde{V}_k] = 0$, since we have that $\mathbb{E}[\nabla f_{\mathscr{b}}(x_k)] = \nabla f(x_k)$ by Lemma 5.8. Moreover, we know that $\mathrm{Var}[\nabla f_{\mathscr{b}}(x_k)] = \frac{1}{\mathscr{b}} D(x_k)$, therefore

$$\mathrm{Var}[\tilde{V}_k] = \eta \mathrm{Var}[\nabla f_{\mathscr{b}}(x_k)] = \frac{\eta}{\mathscr{b}} D(x_k).$$

Thus, the proof is equivalent to the one given in Theorem 5.2 just dividing D by the constant $\mathscr{b}$.

■

In conclusion, we have approximated the learning iteration for mini-batch SGD as a stochastic process described in (5.10). Therefore, we can obtain results about its behaviour by studying the associated stochastic process and its probability density function.

CHAPTER 6

Asymptotic behaviour of SGD

As it has been studied so far in the literature, two main problems emerge when training deep neural networks. On the one hand, it is required that the neural network minimizes a loss function with respect to a given training sample by choosing suitable weights. As we have previously commented, optimizing a loss function in a high-dimensional space is a very difficult problem, since global minima may be elusive. On the other hand, we would like the neural network to avoid overfitting, that is, it has to be able to generalize from the training sample. This is a crucial task in deep learning since we do not want a neural network which is only capable of memorizing the training data, indeed the neural network should learn from the training sample and perform good results for new data.

The challenge of obtaining both properties is the so-called bias-variance tradeoff, which has been tried to resolve by implementing an explicit regularization term in the loss function. In this way, we penalized weights with large norms in the learning process and hence we avoid overfitting. However, in practice, it has been shown that SGD performs better generalization results than other methods even if they used some regularization. Therefore, is widely believed that SGD perform implicit regularization when used to train deep neural networks, but the precise manner in which this occurs has thus far been elusive.

In this final chapter, we follow [Chaudhari-Soatto] in order to prove that this fact about SGD is well founded. Indeed, we prove that SGD minimizes an average potential over the posterior distribution of weights along with an entropic regularization term. Moreover, we will show in which situations this potential is the loss function and hence SGD performs variational inference.

Concretely, for a loss function $f(x)$ with weights $x \in \mathbb{R}^d$, if ρ^{ss} is the steady-state distribution over the weights estimated by SGD, we prove that

$$\rho^{ss} = \arg\min_{\rho} \mathbb{E}_{x\sim\rho}\Big[\Phi(x)\Big] - \frac{\eta}{2b}H(\rho),$$

where $H(\rho)$ is the entropy of the distribution ρ and η and b are the learning rate and the batch-size, respectively. We next prove that the implicit potential $\Phi(x)$ is equal to our chosen loss $f(x)$ if and only if the noise in mini-batch gradients is isotropic. However, this condition is not always satisfied by deep neural networks and hence SGD implicitly discovers locations where $\nabla\Phi(x) = 0$, which are different from $\nabla f(x) = 0$. Thanks to the special architecture of deep networks, SGD helps itself to a potential Φ with properties that lead to both generalization and acceleration.

6.1. Framework

In this section, we present the theoretical background and the assumptions that are necessary for understanding Theorem 6.10, which is the main result of this chapter.

Functionals over probability density functions

We will consider the space of probability measures over $\Omega \subset \mathbb{R}^d$ which are absolutely continuous with respect to the Lebesgue measure. Nevertheless, this theory can be generalized to the whole space of probability measures over Ω without difficulty.

We are particularly interested in the probability distribution functions of those probability measures. Thus, we define the following space.

Definition 6.1. Let $\mathcal{D}(\Omega)$ denote the space of all probability density functions over Ω, that is

$$\mathcal{D}(\Omega) = \{\rho \in L^1(\Omega) \quad : \quad \|\rho\|_1 = 1 \quad \text{and} \quad \rho(x) \geq 0 \ \text{ a.e. } \ x \in \Omega\}.$$

Notice that this space is complete with respect to the L^1-norm, but it is not a Banach space since it is not a vector space.

Let us see some interesting functionals defined over this space.

Definition 6.2. (Entropy). We denote the entropy of a given probability density function by the functional $H : \mathcal{D}(\Omega) \to \mathbb{R} \cup \{\infty\}$, which is defined as follows:

$$H(\rho) = -\int_\Omega \rho(x) \log(\rho(x)) dx.$$

This is a very popular functional used with different meanings in many areas such as thermodynamics and information theory. In deep learning framework, the entropy of the distribution of the weights is considered a regularization term, since maximizing the entropy implies that values of the weights are evenly distributed and hence overfitting is prevented. From the point of view of information theory, this effect can be seen as a way to avoid the information to be stored in few weights as can be seen in [Achille et al.]. Otherwise, we could obtain a deep neural network which has memorized the training sample but is incapable to generalize.

Let us present a concept which is closely related to entropy.

Definition 6.3. (Kullback-Leibler divergence). Let P and Q be two probability measures over a set Ω, satisfying that P is absolutely continuous with respect to Q. Then, the Kullback-Leibler divergence (KL-divergence) is defined as follows,

$$KL(P \,\|\, Q) = \int_\Omega \log\left(\frac{dP}{dQ}\right) dP$$

where $\frac{dP}{dQ}$ denotes the Radon-Nikodym derivative of P with respect to Q.

Moreover, if P and Q are probability distributions of two continuous random variables with p and q probability density functions respectively, we define

$$KL(p \,\|\, q) = \int_\Omega \log\left(\frac{p(x)}{q(x)}\right) p(x) dx.$$

Then, given a function $\rho_0 \in \mathcal{D}(\Omega)$ we can define the functional $KL(\cdot \,\|\, \rho_0) : \mathcal{D}(\Omega) \to \mathbb{R} \cup \{\infty\}$ by

$$KL(\rho \,\|\, \rho_0) = \int_\Omega \rho(x) \log\left(\frac{\rho(x)}{\rho_0(x)}\right) dx.$$

Notice that by splitting the logarithm we obtain

$$\begin{aligned} KL(\rho \,\|\, \rho_0) &= -\int_\Omega \rho(x)\log(\rho_0(x))dx + \int_\Omega \rho(x)\log(\rho(x))dx \\ &= \mathbb{E}_\rho\big[-\log(\rho_0)\big] - H(\rho). \end{aligned} \tag{6.1}$$

The KL-divergence can be interpreted as a measure of discrepancy between probability density functions as the following lemma shows:

Lemma 6.4. (Information inequality). *Let p and q be two probability density functions, then $KL(p \,\|\, q) \geq 0$ with equality if and only if $p = q$.*

Proof. We first recall the Jensen's inequality, which states that, if $(\Omega, \mathcal{A}, P)$ is a probability space and g is a real-valued function, then for any convex function ϕ on the real line it holds that

$$\phi\left(\int_\Omega g\, dP\right) \leq \int_\Omega \phi \circ g\, dP.$$

Now, let us use this statement in the definition of KL-divergence with $\phi = -\log$ and $g = \frac{q}{p}$,

$$\begin{aligned} KL(p \,\|\, q) &= \int_\Omega \log\left(\frac{p}{q}\right) p\, dx = \int_\Omega -\log\left(\frac{q}{p}\right) p\, dx \\ &\geq -\log\left(\int_\Omega \frac{q}{p} p\, dx\right) = -\log\left(\int_\Omega q\, dx\right) \\ &= -\log(1) = 0. \end{aligned}$$

On the other hand, notice that

$$KL(p \,\|\, p) = \int_\Omega \log\left(\frac{p}{p}\right) p\, dx = \int_\Omega \log(1)\, p\, dx = 0.$$

■

In order to analyse these functionals, we now pass to the computation of the first variations. Since they are only defined in the convex set $\mathcal{D}(\Omega)$, we prefer to give a more specific definition.

Definition 6.5. (First Variation). Given a functional $F : \mathcal{D}(\Omega) \to \mathbb{R} \cup \{\infty\}$, we call $\frac{\delta F}{\delta \rho}(\rho)$, if it exists, any measurable function satisfying

$$\frac{d}{d\epsilon} F(\rho + \epsilon\chi)\bigg|_{\epsilon=0} = \int_\Omega \frac{\delta F}{\delta \rho}(\rho)\, \chi dx$$

for any perturbation $\chi = \tilde{\rho} - \rho$ with $\tilde{\rho} \in L^\infty_c \cap \mathcal{D}(\Omega)$.

Observe that from the fact that $\int d\chi = 0$, it is clear that $\frac{\delta F}{\delta \rho}(\rho)$ is defined up to additive constants. On the other hand, up to this invariance, we have uniqueness. Furthermore, we state the following lemma from [Santambrogio, Section 7.2]:

Lemma 6.6. *Let $F : \mathcal{D}(\Omega) \to \mathbb{R} \cup \{\infty\}$ be an integral functional defined by $F(\rho) = \int_\Omega f(\rho(x))dx$. If $f \in C^1$ and f' satisfies suitable bounds, typically f and f' must have polynomial growth, then it holds that $\frac{\delta F}{\delta \rho}(\rho) = f'(\rho)$.*

We can apply this lemma to the functionals described in this section.

Corollary 6.7. *The first variation of the entropy is given by*

$$\frac{\delta H}{\delta \rho} = -\log(\rho) - 1.$$

Moreover, for a given $\rho_0 \in \mathcal{D}(\Omega)$, the first variation of $KL(\rho \,\|\, \rho_0)$ is

$$\frac{\delta KL(\rho \,\|\, \rho_0)}{\delta \rho} = \log\left(\frac{\rho}{\rho_0}\right) + 1.$$

Proof. By definition of entropy and using Lemma 6.6, we only have to focus on $f(x) = -x\log(x)$. It is clear that $f \in C^1$ and f' have polynomial growth, so we obtain that

$$\frac{\delta H}{\delta \rho}(\rho) = f'(\rho) = -\log(\rho) - 1.$$

For the KL functional, by linearity of the first variation and using (6.1), we only have to compute the first variation of $E(\rho) = \int_\Omega -\rho \log(\rho_0) dx$ which can be done directly,

$$\begin{aligned} \frac{d}{d\epsilon} E(\rho + \epsilon\chi)\bigg|_{\epsilon=0} &= \lim_{t\to 0} \int_\Omega -\frac{1}{t}\left[(\rho + (\epsilon + t)\chi) - (\rho + \epsilon\chi)\right]\log(\rho_0) dx \\ &= \int_\Omega -\log(\rho_0)\chi dx. \end{aligned}$$

Hence,

$$\begin{aligned} \frac{\delta KL(\rho \,\|\, \rho_0)}{\delta \rho} = \frac{\delta E}{\delta \rho} - \frac{\delta H}{\delta \rho} &= -\log(\rho_0) + \log(\rho) + 1 \\ &= \log\left(\frac{\rho}{\rho_0}\right) + 1. \end{aligned}$$

■

Assumptions for Theorem 6.10

We now state the natural assumptions for Theorem 6.10. Before doing that, we summarize in a lemma some essential results from Chapter 4 and Chapter 5, involving the stochastic differential equation for the SGD updates, Corollary 5.9, and the associated Fokker-Planck equation for its probability density function, Theorem 4.14.

Lemma 6.8. *The continuous-time SGD is given by*

$$dx(t) = -\nabla f(x)dt + \sqrt{2\beta^{-1}D(x)}dW(t), \tag{6.2}$$

where D is the diffusion matrix described in Chapter 5 and $\beta^{-1} = \frac{\eta}{2b}$ is the temperature. The probability density function of the weights $\rho(x,t)$ evolves according to the Fokker-Planck equation:

$$\partial_t \rho = \nabla \cdot \left(\nabla f(x)\rho + \beta^{-1}\nabla \cdot (D(x)\rho)\right) \tag{FP}$$

where the divergence operator is applied column-wise to matrices such as $D(x)$.

Note that β^{-1} completely captures the magnitude of the noise in SGD that depends only on the learning rate η and the mini-batch size $\mathscr{b}$. This noise is crucial to improve SGD behaviour around saddle points for non-convex losses, therefore optimizing β^{-1} is an a relevant problem, as can be seen in [Nakamura et al.], for instance.

Let us present the first assumption concerning solutions for the elliptic equation associated to (FP).

Assumption 1 (Steady-state distribution exists and is unique). We assume that the steady-state probability density function of the Fokker-Planck equation (FP) exists and is unique. It is denoted by $\rho^{ss}(x)$ and satisfies

$$0 = \partial_t \rho^{ss} = \nabla \cdot \left(\nabla f(x) \rho^{ss} + \beta^{-1} \nabla \cdot (D(x) \rho^{ss}) \right). \tag{6.3}$$

This assumption is quite natural since we are considering ρ^{ss} as a solution of the linear equation (6.3). Therefore, under smoothness and growth condition over the loss function and the diffusion matrix this assumption will be satisfied. For more details see [Bogachev et al.].

Now, let us implicitly define the potential $\Phi(x)$ using the steady-state probability density function ρ^{ss}:

$$\Phi(x) = -\beta^{-1} \log \left(\rho^{ss}(x) \right) + C.$$

Thus, we can express ρ^{ss} in terms of the potential using a normalizing constant $Z(\beta) = \int_\Omega e^{-\beta\Phi}$ as

$$\rho^{ss}(x) = \frac{1}{Z(\beta)} e^{-\beta\Phi(x)}.$$

It can be easily proved that it is also the steady-state solution of

$$dx = \beta^{-1} \nabla \cdot D(x) dt - D(x) \nabla \Phi(x) dt + \sqrt{2\beta^{-1} D(x)} dW(t), \tag{6.4}$$

just by direct substitution in (FP).

Note that this remark suggests that if $\nabla f(x)$ can be written in terms of the diffusion matrix and a gradient term $\nabla\Phi(x)$, the steady-state probability density function of this SDE can be easily obtained. We leverage this observation to define the difference between $-\nabla f(x)$ and the drift term in (6.4):

$$j(x) = -\nabla f(x) + D(x) \nabla \Phi(x) - \beta^{-1} \nabla \cdot D(x). \tag{6.5}$$

We now make an important assumption on $j(x)$ which has its origins in thermodynamics.

Assumption 2 (Force j(x) is conservative). We assume that

$$\nabla \cdot j(x) = 0.$$

In physics, the Fokker-Planck equation typically models a system which exchanges energy with an external environment. In our case, this physical system is the gradient dynamics $\nabla\cdot(\nabla f\,\rho)$ and the interaction with the environment is through the term involving temperature $\beta^{-1}\nabla\cdot(\nabla\cdot(D\rho))$. This assumption is motivated by the second law of thermodynamics which states that the entropy of a system can never decrease.

An important consequence of this assumption is given by the following lemma:

Lemma 6.9. *Under Assumption 2, it holds that $j(x)$ is orthogonal to $\nabla\rho^{ss}(x)$ for any $x \in \Omega$.*

Proof. Using the definition of $j(x)$, we can rewrite the Fokker-Planck equation (FP) as

$$\begin{aligned} 0 = \partial_t \rho^{ss} &= \nabla \cdot (-j\rho^{ss} + D\nabla\Phi\rho^{ss} - \beta^{-1}(\nabla \cdot D)\rho^{ss} + \beta^{-1}\nabla \cdot (D\rho^{ss})) \\ &= \nabla \cdot J^{ss}. \end{aligned}$$

From (6.4), we also have that

$$0 = \partial_t \rho^{ss} = \nabla \cdot (D\nabla\Phi\rho^{ss} + \beta^{-1}D\nabla\rho^{ss}),$$

and consequently,

$$\begin{aligned} 0 &= \nabla \cdot (j\rho^{ss}) \\ &= (\nabla \cdot j)\rho^{ss} - j \cdot \nabla\rho^{ss}. \end{aligned}$$

Thus, by Assumption 2 we conclude that $j \cdot \nabla\rho^{ss} = 0$.

■

6.2. SGD performs variational inference

We are now in a position to state the main result of this chapter, which is given by [Chaudhari-Soatto].

Theorem 6.10. *The functional*

$$F(\rho) = \beta^{-1} KL(\rho \,\|\, \rho^{ss}) \tag{6.6}$$

decreases monotonically along the trajectories of the Fokker-Planck equation (FP)*. and converge to its minimum, which is zero, at steady-state. Moreover, we also have an energetic-entropic split*

$$F(\rho) = \mathbb{E}_\rho\big[\Phi(x)\big] - \beta^{-1}H(\rho). \tag{6.7}$$

where the potential can be defined implicitly by $\Phi(x) = -\beta^{-1}\log(\rho^{ss}(x))$.

This theorem shows that SGD implicitly minimizes a functional which is a combination of an energetic term and an entropic term. Note that the first is the average of a potential (not necessary equal to the loss function) over a distribution ρ. That is, the steady-states is such that it places most of its probability mass in regions where Φ has small values over the weights space. The second term shows that SGD has an implicit bias towards solutions that maximize the entropy of ρ. This is closely associated with the implicit regularization of SGD which is widely believed by empirical results, see [Zhang et al.].

Proof. By Lemma 6.4, we know that $F(\rho) \geq 0$ with equality if and only if $\rho = \rho^{ss}$.

The proof of this theorem is based on showing that $\frac{dF(\rho)}{dt} \leq 0$ with equality only at $\rho = \rho^{ss}$ when $F(\rho)$ reaches its minimum and the Fokker-Planck equation achieves its steady-state.

STEP I: We compute the first variation of the functional $F(\rho) = \beta^{-1}\int_\Omega \rho(x)\log\left(\frac{\rho}{\rho^{ss}}\right)$. Using Corollary 6.7 and the linearity of the first variation we obtain

$$\frac{\delta F}{\delta \rho}(\rho) = \beta^{-1}\left(\log\left(\frac{\rho}{\rho^{ss}}\right) + 1\right).$$

Note that we can rewrite it in terms of the potential $\Phi(x) = -\beta^{-1}\log(\rho^{ss}(x))$ as

$$\frac{\delta F}{\delta \rho}(\rho) = \Phi + \beta^{-1}(\log(\rho) + 1). \tag{6.8}$$

STEP II: Let us try to rewrite the Fokker-Planck equation (FP), in terms of the first variation of F and the force j described in (6.5).

From equation (6.8), we obtain that

$$\nabla\left(\frac{\delta F}{\delta\rho}\right) = \beta^{-1}\frac{1}{\rho}\nabla\rho + \nabla\Phi$$

and

$$-j\rho = \rho\nabla f - \rho D\,\nabla\Phi + \beta^{-1}\rho\,\nabla\cdot D.$$

Thus,

$$\begin{aligned}-j\rho + \rho\,D\,\nabla\left(\frac{\delta F}{\delta\rho}\right) &= \rho\nabla f - \rho D\,\nabla\Phi + \beta^{-1}\rho\,\nabla\cdot D + \beta^{-1}D\nabla\rho + \rho D\nabla\Phi \\ &= \rho\nabla f + \beta^{-1}\rho\,\nabla\cdot D + \beta^{-1}D\nabla\rho \\ &= \rho\nabla f + \beta^{-1}\nabla\cdot(D\rho),\end{aligned}$$

since $\nabla\cdot(D\rho) = (\nabla\cdot D)\rho + D\nabla\rho$. Then, we can rewrite (FP) as follows,

$$\partial_t\rho = \nabla\cdot\left(-j\,\rho + \rho\,D\,\nabla\left(\frac{\delta F}{\delta\rho}\right)\right). \tag{6.9}$$

STEP III: Let us compute the derivative of F with respect to the time t.

$$\begin{aligned}\frac{dF(\rho)}{dt} &= \beta^{-1}\int_\Omega \frac{d}{dt}\left(\rho\log\left(\frac{\rho}{\rho^{ss}}\right)\right)dx = \beta^{-1}\int_\Omega \partial_t\rho\log\left(\frac{\rho}{\rho^{ss}}\right) + \partial_t\rho\,dx \\ &= \int_\Omega \partial_t\rho\left(\beta^{-1}\log\left(\frac{\rho}{\rho^{ss}}\right) + \beta^{-1}\right)dx = \int_\Omega \partial_t\rho\,\frac{\delta F}{\delta\rho}\,dx\end{aligned}$$

Then, we replace $\partial_t\rho$ by its representation in (6.9) in order to obtain that

$$\begin{aligned}\frac{dF(\rho)}{dt} &= \int_\Omega \partial_t\rho\,\frac{\delta F}{\partial\rho}\,dx \\ &= \int_\Omega \frac{\delta F}{\delta\rho}\nabla\cdot(-j\;\rho)\,dx + \int_\Omega \frac{\delta F}{\delta\rho}\nabla\cdot\left(\rho\,D\,\nabla\left(\frac{\delta F}{\delta\rho}\right)\right)dx\end{aligned}$$

Let us show that the first term in the right hand side vanishes. Note that by Assumption 2 we have $\nabla\cdot(-j\rho) = -\nabla\cdot j\;\rho - j\cdot\nabla\rho = -j\cdot\nabla\rho$, therefore it suffices to prove

$$\int_\Omega \frac{\partial F}{\delta\rho}j\cdot\nabla\rho dx = 0. \tag{6.10}$$

With this purpose, we use definition of $\frac{\delta F}{\delta\rho}$ and integration by parts to obtain

$$\begin{aligned}\int_\Omega \frac{\delta F}{\delta\rho}j\cdot\nabla\rho &= \beta^{-1}\int_\Omega j\cdot\nabla\rho + \beta^{-1}\int_\Omega \log(\rho)j\cdot\nabla\rho - \beta^{-1}\int_\Omega \log(\rho^{ss})j\cdot\nabla\rho \\ &= \beta^{-1}\int_\Omega j\cdot\nabla\rho - \beta^{-1}\int_\Omega \nabla\cdot(\log(\rho)j)\rho + \beta^{-1}\int_\Omega \nabla\cdot(\log(\rho^{ss})j)\rho \\ &= \beta^{-1}\int_\Omega j\cdot\nabla\rho - \beta^{-1}\int_\Omega \nabla\rho\cdot j - \beta^{-1}\int_\Omega \log(\rho)\rho\,\nabla\cdot j + \beta^{-1}\int_\Omega \nabla\cdot(\log(\rho^{ss})j)\rho \\ &= -\beta^{-1}\int_\Omega \log(\rho)\rho\,\nabla\cdot j + \beta^{-1}\int_\Omega \frac{\rho}{\rho^{ss}}\nabla\rho^{ss}\cdot j + \beta^{-1}\int_\Omega \log(\rho^{ss})\rho\nabla\cdot j\end{aligned}$$

Thus, applying Assumption 2 and Lemma 6.9 we obtain (6.10). This lead us to express time derivative of $F(\rho)$ by

$$\frac{dF(\rho)}{dt} = \int_\Omega \frac{\delta F}{\delta \rho} \nabla \cdot \left(\rho\, D\, \nabla \left(\frac{\delta F}{\delta \rho} \right) \right) dx.$$

STEP IV: In order to conclude the proof we have to show $\frac{dF(\rho)}{dt} \leq 0$ and $\frac{dF(\rho^{ss})}{dt} = 0$.

Under suitable boundary condition on the Fokker-Planck equation which ensures that no probability mass flows across the boundary of the domain $\partial\Omega$ as in Step III, we obtain the following result using integration by parts:

$$\begin{aligned} \frac{dF(\rho)}{dt} &= -\int_\Omega \nabla \left(\frac{\delta F}{\delta \rho} \right) \cdot \left(\rho\, D\, \nabla \left(\frac{\delta F}{\delta \rho} \right) \right) dx \\ &= -\int_\Omega \rho \, \left(D\, \nabla \left(\frac{\delta F}{\delta \rho} \right) \right) \cdot \nabla \left(\frac{\delta F}{\delta \rho} \right) dx \\ &\leq 0, \end{aligned}$$

since $\rho \geq 0$ and D is positive-defined, that is, $Du \cdot u \geq 0$ for any $u \in \mathbb{R}^d$. Furthermore, for ρ^{ss} it holds that

$$\frac{\delta F}{\delta \rho}(\rho^{ss}) = \beta^{-1} + \beta^{-1} \log \left(\frac{\rho^{ss}}{\rho^{ss}} \right) = \beta^{-1}.$$

Thus, $\nabla \left(\frac{\delta F}{\delta \rho}(\rho^{ss}) \right) = 0$, which implies that

$$\frac{dF(\rho^{ss})}{dt} = 0.$$

■

A natural question arises from Theorem 6.10: *Which is the potential Φ that is minimized in the functional $F(\rho)$?* Indeed, we are interested in the relation between Φ and the loss function f. For this purpose, let us show the following lemma:

Lemma 6.11. *If the diffusion matrix $D(x)$ is isotropic, i.e., a positive constant multiple of the identity, the implicit potential is the original loss itself*

$$D(x) = cI_{d\times d} \quad \Leftrightarrow \quad \Phi(x) = \frac{1}{c} f(x).$$

Proof. The forward implication is given by substituting $\rho^{ss}(x) \propto e^{-\frac{\beta}{c} f(x)}$ in the Fokker-Planck equation (FP),

$$\nabla \cdot (\nabla f\, e^{-\frac{\beta}{c} f} + \beta^{-1} \nabla \cdot (cI\, e^{-\frac{\beta}{c} f})) = \nabla \cdot \left(\nabla f\, e^{-\frac{\beta}{c} f} - \nabla f\, e^{-\frac{\beta}{c} f} \right) = 0.$$

The reverse implication is true since otherwise it would contradict Lemma 6.9.

■

6.3. Conclusion and further discussion

The main result of this chapter states that SGD does perform variational inference using a new potential which is not necessarily equal to the loss function. Indeed, if the diffusion matrix is non-isotropic, as it happens usually in deep neural networks, the difference between them will be considerable. In [Chaudhari-Soatto], they give an explicit computation of this new potential by comparing the A-type SDE and Itô's SDE associated to the same Fokker-Planck equation.

Despite this difference between Φ and f may seem problematic, it opens a new way to study SGD learning. A remarkable point of view proposed in [Soatto et al.] is that minimizing the potential Φ instead of the loss function lead SGD to approach wide minimums of f. As it has been seen in practice, avoiding sharp minimums is quite desirable, since they often cause overfitting. On the other hand, the entropic term of the functional $F(\rho)$ also provides the implicit regularization of SGD that has been commonly believed. Note that if $\beta^{-1} \to 0$, the entropic term vanishes, which it implies that $\beta^{-1} = \frac{\eta}{2\beta}$ should not be small.

Additionally, Theorem 6.10 also can be studied in the framework of Gradient Flows, where we consider Fokker-Planck equation as an infinte-dimensional ODE, see [Santambrogio]. Observe that if the diffusion matrix is isotropic, this theorem is equivalent to the celebrated JKO functional in optimal transportation [Jordan et al.].

From the point of view of information theory, the functional $F(\rho)$ is equivalent to the information bottleneck principle [Tishby et al.]. Moreover, minimizing this functional explicitly has been shown to lead to invariant representation. Theorem 6.10 proves that SGD implicitly contains this bottleneck and therefore begets these properties, naturally.

An interesting open question that we propose is implementing in SGD other noises instead of white noise which lead us to Brownian motion. We suggest to consider this variation which may lead us to the novel framework of fractional Laplacian.

In conclusion, we have presented a global view of the modern theory about deep neural networks, describing their approximation properties and analysing the commonly used learning process, SGD. We also have described a line of investigation about this subject showing many open problems.

Bibliography

[Achille et al.] A. Achille, G. Paolini and S. Soatto. Where is the Information in a Deep Neural Network?. *arXiv preprint arXiv:1905.12213*, 2019.

[Arora et al.] R. Arora, A. Basu, P. Mianjy and A. Mukherjee. Understanding Deep Neural Networks with Rectified Linear Units. *arXiv preprint arXiv:1611.01491*, 2018.

[Anthony-Barlett] M. Anthony and P.L. Barlett. *Neuronal network learning: Theoretical foundations.* Cambridge university press, 2009.

[Barlett et al.] P.L. Barlett, N. Harvey, C.Liaw and A. Mehrabian. *Nearly-tight VC-dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks.* Journal of Machine Learning Research 20(63):1-17, 2019.

[Brezis] H. Brezis. *Functional analysis, Sobolev spaces and partial differential equations.* Springer, New York, 2011.

[Bogachev et al.] V.I. Bogachev, N.V. Krylov, M. Röckner and S.V. Shaposhnikov. *Fokker-Planck-Kolmogorov Equations.* American Mathematical Society, 2015.

[Cauchy] A. Cauchy. *Methodes generales pour la resolution des systemes dequations simultanees.* C.R. Acad. Sci. Par., 25:536-538, 1847.

[Chaudhari-Soatto] P. Chaudhari and S. Soatto. Stochastic Gradient Descent performs variational inference, converges to limit cycles for Deep Networks. *arXiv preprint arXiv:1710.11029*, 2018.

[Evans] L.C. Evans. *An introduction to stochastic differential equations.* American Mathematical Society, 2013.

[Friedman] A. Friedman. *Partial differential equations of parabolic type.* Prentice-Hall, 1964.

[Higham] C. F. Higham and D.J. Higham. Deep learning: an introduction for applied mathematicians. *arXiv preprint arXiv:1801.05894*, 2018.

[Jordan et al.] R. Jordan, D. Kinderlehrer and F. Otto. *Free energy and the Fokker-Planck equation.* Physica D:Nonlinear Phenomena, 107(2-4):265-271, 1997.

[Li et al.] Q. Li, C. Tai and E. Weinan. *Stochastic modified equations and adaptative stochastic gradient algorithms.* In ICML, pages 2101-2110, 2017.

[Milstein] GN. Milstein. *Weak approximation of solutions of systems of stochastic differential equations.* Theory of Probability & Its Applications, 30(4):750-766, 1986.

[Nakamura et al.] K. Nakamura, S. Soatto and B. Hong. Stochastic batch size for adpative regularization in deep network optimization. *arXiv preprint arXiv:2004.06341,* 2020.

[Nesterov] Y. Nesterov. *Introductory lectures on convex optimization. A basic course.* Springer Science & Business Media, 2003.

[Rumelhart et al.] D. E. Rumelhart, G.E. Hinton and R.J. Williams. *Learning representations by back-propagating error.* Nature 323(6088):533-536.

[Santambrogio] F. Santambrogio. *Optimal Transport for Applied Mathematicians.* Birkuser, NY, 2015.

[Schuss] Z. Schuss. *Theory and Applications of Stochastic Differential Equations,* volume 170 of *Applied Mathematical Science.* Springer, New York, 2010.

[Soatto et al.] S. Soatto, P. Chaudhari, A. Choromanska, Y.LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun and R. Zecchina. Entropy-SGD: biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838,* 2016.

[Tishby et al.] N. Tishby, F. C. Pereira and W. Bialek. *The information bottleneck method.* In Proc of 37-th Annual Allerton Conference on Communication, Control and Computing, pages 368-377, 1999.

[Wang-Sun] S. Wang and X. Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

[Yarotsky 2017] D. Yarotsky. Error bounds for approximations with deep ReLU networks. Neural Networks, 94:103-114, 2017.

[Yarotsky 2018] D. Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. *arXiv prepint arXiv:1802.03620*, 2018.

[Zhang et al.] C.Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.