5 Stochastic gradient methods

An algorithm must be seen to be believed.

Donald Knuth

The calculation on the computer of an effective solution to the minimization problem $W_* \approx \text{argmin}\ J(W)$ is of supreme importance for real problems [34]. This activity is called the **training** of the parameters. Most of the successes of Neural Networks and Machine Learning are direct consequences of the fact that the training can be efficiently performed with dedicated softwares.

Hereafter we discuss and interpret (mostly in the context of ordinary differential equations ODEs) some issues which concern the minimization of the objective function written as

$$W \mapsto J(W) = \sum_{(x,y)\in\mathcal{D}} \varphi_y(f(x,W)) \tag{5.1}$$

here φ_y is some cost function (in our case, the least square function or cross-entropy function), y is the output, and the control parameter W is distributed through a neural network. The dataset $\mathcal{D}$ is fixed (perhaps it has been constructed with some methods described in the previous Chapter). The parameter W is the unknown of the minimization problem. We consider that W is restricted in a bounded set $\Omega \subset \mathbb{R}^q$ because it is the case in practice. A basic Theorem is the following.

Theorem 5.0.1. *Let $J : \Omega \to \mathbb{R}$ be a continuous function over a* **closed bounded set** *in finite dimension $\Omega \subset \mathbb{R}^q$. Then there exists a minimizer $W_* \in \Omega$ such that $J(W_*) \le J(W)$ for all $W \in \Omega$.*

The goal of the algorithms explained below is to calculate an approximation of W_*. All algorithms can be written as variations around the main theme, which is the **steepest gradient method**. Even if the steepest gradient method is convenient as a theoretical introduction, it cannot be sufficient for applications. In what follows, we also consider four developments of the steepest gradient method

- Momentum methods which deal with non convex functions J.
- Batches which deal with large or very large datasets, that is $\#(\mathcal{D}) \gg 1$.
- Initialization of the algorithms.
- The case where J has no second order derivative.

The combination of the three first techniques constitute the algorithmic foundations of what are **stochastic gradient methods**.

5.1 Steepest gradient method

The discrete form of the steepest gradient method can be written as

$$W^{n+1} = W^n - \Delta t \ \nabla J(W^n), \qquad \Delta t > 0 \tag{5.2}$$

where $\Delta t > 0$ is some gradient length which is written as a time step because it is convenient to for the interpretation of the algorithms. The continuous version of (5.2) introduces a fictitious continuous time variable $t > 0$

$$\begin{cases} W'(t) = -\nabla J(W(t)), & t > 0, \\ W(t) = W_0. \end{cases} \tag{5.3}$$

One has formally

$$\frac{d}{dt} J(W(t)) = - \left|\nabla J(W(t))\right|^2 \leq 0,$$

so the value of the functional decreases.

The Cauchy-Lipschitz Theorem assesses the well-posedness of the continuous problem.

Theorem 5.1.1 (Cauchy-Lipschitz theorem)**.** *Take $W_0 \in \Omega$ where the domain $\Omega \subset \mathbb{R}^q$ is an open set. Assume $\nabla J : \Omega \to \mathbb{R}$ is Lipschitz*

$$\left|\nabla^2 J(W)\right| \leq C \text{ for all } W \in \Omega.$$

Then there exists a maximal time interval $I = [0, T)$ such that the Cauchy problem has a unique solution $W \in C^0(I : \Omega)$.

Let us make the stronger assumptions that J is a convex functional with a bounded and positive Hessian matrix $\nabla^2 J \in C^0(\mathbb{R}^q : \mathcal{M}_q(\mathbb{R}))$

$$\beta \geq \nabla^2 J \geq \alpha > 0 \tag{5.4}$$

Here the domain is $\Omega = \mathbb{R}^q$. If the number of hidden layers is $p \geq 1$, then the convexity assumption is not realistic. If J is assembled with ReLU functions (or alike) as activation functions, then J cannot be twice differentiable which is another reason why the assumptions are not realistic. Nevertheless we continue the presentation of the steepest gradient method in this context.

Let us consider that W_* is an extremal point

$$\nabla J(W_*) = 0. \tag{5.5}$$

Then there exists $\mu \in \mathbb{R}$ such that

$$J(W) = J(W_*) + \langle \nabla J(W_*), W - W_* \rangle \tag{5.6}$$
$$+\frac{1}{2}\left\langle \nabla^2 J(W_* + \mu(W - W_*))(W - W_*), W - W_* \right\rangle .$$

It yields that

$$J(W_*) \le J(W_*)+\frac{\alpha}{2}\left|W - W_*\right|^2 \le J(W) \le J(W_*)+\frac{\beta}{2}\left|W - W_*\right|^2 \text{ for all } W \in \mathbb{R}^q. \tag{5.7}$$

So W_* is a minimizer and moreover it is the unique minimizer.

Lemma 5.1.2. *Under assumptions (5.4-5.5), one has* $|W(t) - W_*| \le e^{-\alpha t}\left|W(0) - W_*\right|$.

Proof. Indeed

$$\frac{d}{dt}\left|W(t) - W_*\right|^2 = 2\left\langle W'(t), W(t) - W_* \right\rangle = -2\left\langle \nabla J(W(t)), W(t) - W_* \right\rangle$$
$$= -2\left\langle \nabla J(W(t)) - \nabla J(W_*), W(t) - W_* \right\rangle .$$

A classical estimate for the convex function J which satisfies (5.4) is

$$\alpha\left|W_1 - W_2\right|^2 \le \langle \nabla J(W_1) - \nabla J(W_2), W_1 - W_2 \rangle \le \beta\left|W_1 - W_2\right|^2 . \tag{5.8}$$

So $\frac{d}{dt}\left|W(t) - W_*\right|^2 \le -2\alpha\left|W(t) - W_*\right|^2$ and $\frac{d}{dt}\left(e^{2\alpha t}\left|W(t) - W_*\right|^2\right) \le 0$. It yields $e^{2\alpha t}\left|W(t) - W_*\right|^2 \le \left|W(0) - W_*\right|^2$. □

Now we consider the discrete algorithm (5.2).

Lemma 5.1.3. *Assume (5.4-5.5) with* $\Delta t < \frac{2\alpha}{\beta^2}$. *Then there exists* $\varepsilon \in (0,1)$ *such that*

$$\left|W^n - W_*\right| \le (1-\varepsilon)^n\left|W^0 - W_*\right| .$$

Proof. The iteration (5.2) is rewritten as $W^{n+1} - W_* = W^n - W_* - \Delta t \nabla J(W^n)$. It implies

$$\left|W^{n+1} - W_*\right|^2 = \left|W^n - W_*\right|^2 - 2\Delta t\left\langle W^n - W_*, \nabla J(W^n)\right\rangle + \Delta t^2\left|\nabla J(W^n)\right|^2 .$$

A bound on $|\nabla J(W^n)|$ is easily obtained from the formula

$$\nabla J(W_1) - \nabla J(W_2) = \int_0^1 \frac{d}{ds}\nabla J(W_1 + s(W_2 - W_1))ds$$

$$= \int_0^1 \nabla^2 J(W_1 + s(W_2 - W_1))(W_2 - W_1)ds,$$

so one can write $|\nabla J(W^n)| = |\nabla J(W^n) - \nabla J(W_*)| \leq \beta |W^n - W_*|$. Using (5.8), one gets

$$\left|W^{n+1} - W_*\right|^2 \leq \left(1 - 2\Delta t\alpha + \Delta t^2\beta^2\right) \left|W^n - W_*\right|^2.$$

The hypothesis implies that $\varepsilon = 2\Delta t\alpha - \Delta t^2\beta^2 > 0$ so the claim is obtained. □

To implement the steepest gradient method, it is necessary to calculate the gradient $\nabla J(W)$ of the function (5.1). For this task, the softwares evoked in the introduction make a great use of exact methods by means of the chain rule formula. It yields the foundations of what is called **back propagation methods** [34]. In the following we reformulate the calculation of the gradient of the cost function with respect to various variables as the multiplication of Jacobian matrices in correct order. This reformulation is natural from a mathematical perspective [70]. The algorithmic foundations have been completely established in [38][formulas (3.5) and (3.8) page 39], even if this formalism does not seem to be employed any more in modern references [34][Chapter 6].

Notation 5.1.4. *We adopt the following conventions in this section.*

- *The gradient of $u \in C^1\left(\mathbb{R}^a : \mathbb{R}^b\right)$ is a matrix with b lines and a columns*

$$\nabla u = \begin{pmatrix} \partial_1 u_1 & \partial_2 u_1 & \dots & \partial_a u_1 \\ \partial_1 u_2 & \partial_2 u_2 & \dots & \partial_a u_2 \\ \dots & \dots & \dots & \dots \\ \partial_1 u_b & \partial_2 u_b & \dots & \partial_a u_b \end{pmatrix} \in \mathcal{M}_{b,a}(\mathbb{R}).$$

- *The gradient of $\psi \in C^1\left(\mathcal{M}_{a,b}(\mathbb{R}) : \mathbb{R}\right)$ is also a matrix with b lines and a columns*

$$\nabla \psi = \begin{pmatrix} \partial_{(11)}\psi & \partial_{(21)}\psi & \dots & \partial_{(a1)}\psi \\ \partial_{(12)}\psi & \partial_{(22)}\psi & \dots & \partial_{(a2)}\psi \\ \dots & \dots & \dots & \dots \\ \partial_{(1b)}\psi & \partial_{(2b)}\psi & \dots & \partial_{(ab)}\psi \end{pmatrix} \in \mathcal{M}_{b,a}(\mathbb{R}) = \mathcal{M}_{a,b}(\mathbb{R})^t.$$

Remark 5.1.5. *Take $b = 1$. Then both notations coincide and the gradient of a scalar function is vector which belongs to $\mathcal{M}_{1a}(\mathbb{R}) = (\mathbb{R}^a)^t$.*

Let $u \in C^1\left(\mathbb{R}^a : \mathbb{R}^b\right)$ and $v \in C^1\left(\mathbb{R}^b : \mathbb{R}^c\right)$. The gradient of $w = v \circ u \in C^1\left(\mathbb{R}^a : \mathbb{R}^c\right)$ is given by the chain rule which is the multiplication of matrices

$$\nabla w = \underbrace{\nabla v \circ u}_{\in \mathcal{M}_{c,b}(\mathbb{R})} \times \underbrace{\nabla u}_{\in \mathcal{M}_{b,a}(\mathbb{R})} \in \mathcal{M}_{c,a}(\mathbb{R}). \tag{5.9}$$

Take $f \in C^1(\mathbb{R}^m : \mathbb{R}^n)$ given by (1.25) with a smooth activation function S. The function can be noted

$$(x \mid W_p, \dots W_0 : b_p, \dots, b_0) \longmapsto f(x \mid W_p, \dots W_0 : b_p, \dots, b_0). \tag{5.10}$$

Take a real valued function $\varphi \in C^1(\mathbb{R}^n : \mathbb{R})$ such as the one used in (5.1) to assemble the cost function J: in this case $\varphi = \varphi_y$ has an additional dependance with respect to y. Since f in (5.10) displays a dependance with respect to the input x, the weights W_rs and to the biases b_rs, the function $\varphi \circ f$ has partial derivatives with respect to the input, the weights and the biases.

To ease the notations, the partial calculations of (1.25) are denoted as

$$f^r = f_r \circ S_r \circ f_{r-1} \circ \dots \circ f_1 \circ S_0 \circ f_0, \quad 0 \le r \le p,$$

where S_r, $0 \le r \le p$, denotes the sigmoid function applied component wise to a vector $\in \mathbb{R}^{a_{r+1}}$. The final step yields the function $f = f^p$. All functions f_r, $0 \le r \le p$, are affine functions whose parameters are controlled by the weight matrix $W_r \in \mathcal{M}_{a_{r+1},a_r}(\mathbb{R})$ and the bias vector $b_r \in \mathbb{R}^{a_{r+1}}$.

Lemma 5.1.6. *The gradient of $\varphi \circ f$ with respect to $x \in \mathbb{R}^m$ is $\nabla_x(\varphi \circ f) \in (\mathbb{R}^m)^t$ defined by*

$$\nabla_x(\varphi \circ f) = \nabla\varphi \circ f^p \times W_p \times \nabla S_p \circ f^{p-1} \times W_{p-1} \times \dots \times W_1 \times \nabla S_1 \circ f^0 \times W_0 \tag{5.11}$$

where $\nabla S_r \in \mathcal{M}_{a_{r+1}}(\mathbb{R})$, $0 \le r \le p$.

Proof. It comes from a repeated application of (5.9) combined with the fact that

$$\nabla_{x_r} f_r = \nabla_{x_r}(W_r x_r + b_r) = W_r, \qquad 0 \le r \le p.$$

The function S_r is a function from $\mathbb{R}^{a_{r+1}}$ into itself. This is why its gradient is a square matrix of size a_{r+1}. □

Lemma 5.1.7. *Let $0 \le r \le p$. The gradient of $\varphi \circ f$ with respect to $b_r \in \mathbb{R}^{a_{r+1}}$ is $\nabla_{b_r}(\varphi \circ f) \in (\mathbb{R}^{a_{r+1}})^t$ defined by*

$$\nabla_{b_r}(\varphi \circ f) = \nabla\varphi \circ f^p \times W_p \times \nabla S_p \circ f^{p-1} \times W_{p-1} \times \dots \times W_{r+1} \times \nabla S_{r+1} \circ f^r \tag{5.12}$$

with the same notations as in (5.11).

Proof. The algebra is similar as in the previous proof, but now

$$\nabla_{b_r}(f_r \circ S_{r-1} \circ f^{r-1}) = \nabla_{b_r}(W_r S_{r-1} \circ f^{r-1} + b_r) = I_{a_{r+1}}$$

because $b_r \in \mathbb{R}^{a_{r+1}}$. □

Lemma 5.1.8. *Let $0 \le r \le p$. The gradient of $\varphi \circ f$ with respect to $W^r \in \mathcal{M}_{a_{r+1},a_r}(\mathbb{R})$ is $\nabla_{W^r}(\varphi \circ f) \in \mathcal{M}_{a_r,a_{r+1}}(\mathbb{R})$ defined by*

$$\nabla_{W^r}(\varphi \circ f) = \underbrace{S_r \circ f^{r-1}}_{\in \mathcal{M}_{a_r,1}(\mathbb{R})} \times \underbrace{\nabla_{b_r}(\varphi \circ f)}_{\in \mathcal{M}_{1,a_{r+1}}(\mathbb{R}) = (\mathbb{R}^{a_{r+1}})^t} \tag{5.13}$$

with the same notations as in (5.11).

Proof. Denote $W_r(\alpha) = W_r + \alpha Z_r$ where $Z_r \in \mathcal{M}_{a_r,a_{r+1}}(\mathbb{R})$ is given. Consider $\mu(\alpha)$ which is the value of $\varphi \circ f$ when W_r is replaced by $W_r(\alpha)$. The chain rule yields $\mu'(0) = Z_r : \nabla_{W_r}(\varphi \circ f)^t$.
But one can also write $W_r(\alpha)S_r + b_r = W_r S_r + b_r(\alpha)$ where $b_r(\alpha) = b_r + \alpha Z_r S_r$, so $\mu(\alpha)$ is also the value of $\varphi \circ f$ is calculated when b_r is replaced by $b_r(\alpha)$. Therefore $\mu'(0) = \langle Z_r S_r, \nabla_{b_r}(\varphi \circ f)\rangle$.
It yields the equality $Z_r : \nabla_{W_r}(\varphi \circ f)^t = \langle Z_r S_r, \nabla_{b_r}(\varphi \circ f)\rangle$ for all $Z_r \in \mathcal{M}_{a_r,a_{r+1}}(\mathbb{R})$. With (1.5), one has

$$Z_r : \nabla_{W_r}(\varphi \circ f)^t = Z_r : \nabla_{b_r}(\varphi \circ f) \otimes S_r \text{ for all } Z_r \in \mathcal{M}_{a_r,a_{r+1}}(\mathbb{R}).$$

Since Z_r is arbitrary, it means that $\nabla_{W_r}(\varphi \circ f)^t = \nabla_{b_r}(\varphi \circ f) \otimes S_r$ which is the claim after transposition. □

These formulas allow to describe the essence of the back propagation method which is used to calculate all the gradients with respect to (W_r, b_r), $1 \le r \le p+1$, needed for the various descent methods. The issue is to determine an algorithmic procedure to calculate the derivatives with respect to all W_rs and all b_rs. Once the gradient with respect to b_r is obtained, one has the gradient with respect to W_r by formula (5.13), with minimal calculation cost. So the issue reduces to calculate the derivatives with respect to all b_rs. In view of (5.12), the best procedure is to proceed the calculation with the derivative with respect to b_p, then to calculate the derivative with respect to b_{p-1} and so on and so forth by descending iterations. The general formula writes

$$\nabla_{b_{r-1}}(\varphi \circ f) = \nabla_{b_r}(\varphi \circ f) \times W_r \times \nabla S_r \circ f^{r-1}. \tag{5.14}$$

Definition 5.1.9. *The* **back propagation method** *is the* **descending iterations** *(5.14) from $r = p$ to $r = 0$. It allows to compute the series of matrix-matrix multiplications required to calculate the gradient of J with respect to W.*
The situation is different for the calculation (5.12) of the derivatives with respect to x, for which the multiplication of matrices can be performed forward or backward.

Now one can describe the calculation of the gradient f the function J (5.1). It is sufficient to sum over all elements in the dataset. One obtains

$$\nabla J(W) = \sum_{(x,y)\in\mathcal{D}} \nabla_W \varphi_y(f(x,W)). \tag{5.15}$$

The notation ∇_W indicates a differentiation with respect to all the W_rs and b_rs which constitute the vector of all parameters W. This can be performed with formulas (5.12)-(5.13)-(5.14). The formula (5.11) will be used later.

Remark 5.1.10 (Automatic differentiation)**. Automatic differentiation***, also called symbolic differentiation, is the technology used in modern softwares to implement the calculation of the gradient of the cost function, in particular within a back propagation approach. This method can be understood as a computational exact differentiation of simple operations. The reader has to refer to [14, 34] for more explanations which are out of the scope of these notes. The end result of automatic differentiation is that the gradient of a given complex function is calculated exactly at the computer level,* **provided** *the function is assembled with simple operations whose differentiation is already implemented. In Neural Networks, functions assembled with linear operations and activation functions satisfy this last requirement.*

Automatic differentiation of the ReLU function is particularly simple, since the result is zero or one (except at the origin where the derivative is ambiguous). One can understand that it is a decisive argument in favor of NNs with ReLU, since the calculation of the gradient $\nabla J(W)$ is performed at minimal implementation cost. The fact that the the derivative at origin is ambiguous does not seem to have hampered the use of ReLU in modern NNs [14, 34], probably because the practical advantages outperform this theoretical drawback. Note that there is a theoretical possibility (explained in Section 5.6) to reformulate a minimization problem assembled with ReLU so as to avoid the differentiability problem at origin.

5.2 Momentum methods and ADAM

For many minimization problems, the steepest gradient method is far to be the most efficient one. This is due to many reasons. The first evident one is that it is known that the steepest gradient method has a slow rate of convergence. But the situation is more serious, indeed almost all functions J encountered in ML are not convex. This fact has many consequences, for example minimizers may be non unique. Therefore one needs methods which are robust with respect to such features. Momentum methods are quite popular in the ML community because they are well adapted to the kind of functions generated by NNs.

A striking feature of most modern ML softwares is that they do not propose to train with algorithms which are extremely popular in numerical analysis, such as the Newton-Raphson method or the conjugate gradient method and its generalizations [69]. Both methods have the ability to converge at a fast rate toward the local minimum of convex functions (quadratic strictly convex functions in the case of the conjugate gradient method). Even if no definitive explanation can be obtained from the available literature, it is clear the framework of globally convex functions is not adapted to the practical objectives of ML where the functions usually have many local minima and are **highly non convex** with low regularity. Moreover the calculation of the Hessian matrix of the cost function needed to implement the Newton-Raphson method is not really doable, for at least two reasons. The first reason seems that that the number of parameters (i.e. the size of W) can be very large, so the numerical cost of the calculation of the Hessian matrix would hamper the efficiency. The second reason seems that the ReLU function which helps to minimize the calculation with automatic differentiation makes impossible the calculation of second derivatives (because ReLU is not twice differentiable).

Most of momentum methods can be presented can be introduced from the following system of ODEs. One adds at one additional variable Z called the **moment** and considers the continuous in time method

$$t > 0: \qquad \begin{cases} W'(t) = Z(t), \\ Z'(t) = -\nabla J(W(t)). \end{cases} \tag{5.16}$$

A discrete version writes

$$\begin{cases} W^{n+1} = W^n + \Delta t Z^n, & \Delta t > 0, \\ Z^{n+1} = Z^n - \Delta t \ \nabla J(W^n) \end{cases} \tag{5.17}$$

We make the regularity assumption

$$J \in C^2(\Omega : \mathbb{R}). \tag{5.18}$$

An elementary property follows.

Lemma 5.2.1. *Solutions of (5.16) satisfy* $\frac{d}{dt}\left(J(W(t)) + \frac{1}{2}|Z(t)|^2\right) = 0.$

Proof. Indeed

$$\begin{aligned} \frac{d}{dt}\left(J(W(t)) + \frac{1}{2}|Z(t)|^2\right) &= \left\langle W', \nabla J(W)\right\rangle + \left\langle Z', Z\right\rangle \\ &= \langle Z, \nabla J(W)\rangle + \langle -\nabla J(W), Z\rangle = 0. \end{aligned}$$

□

Remark 5.2.2. *The system (5.16) has the same structure as the equations for a punctual mass in a potential field force. The Lemma expresses that the sum of the potential energy $J(W)$ and kinetic energy $\frac{1}{2}|Z(t)|^2$ is constant in time.*

Starting with $W(0) = W_0$ and $Z(0) = 0$, this method is able to visit a database for some W such that $J(W) \leq J(W_0)$. If $J(W) < J(W_0)$, then $Z \neq 0$. This approach is a priori not trapped in local minima, while the disadvantage is that it is non stationary even at a global minimum.

A popular enhancement is called the Nesterov method. It can be written as

$$t > 0 : \quad \begin{cases} W'(t) = Z(t), \\ Z'(t) = -\nabla J(W(t) + \nu Z(t)). \end{cases} \tag{5.19}$$

The parameter $\nu \geq 0$ is the Nesterov acceleration parameter.

Lemma 5.2.3. *Solutions of (5.19) satisfy*

$$\frac{d}{dt}\left(J(W(t) + \nu Z(t)) + \frac{1}{2}|Z(t)|^2\right) = -\nu \left|\nabla J(W(t) + \nu Z(t))\right|^2 \leq 0. \tag{5.20}$$

Proof. This identity shows that the parameter ν can also be interpreted as certain kind of mechanical friction. Note $E(t) = J(W(t) + \nu Z(t)) + \frac{1}{2}|Z(t)|^2$. One has

$$\begin{aligned} E'(t) &= \left\langle W' + \nu Z', \nabla J(W + \nu Z)\right\rangle + \left\langle Z', Z\right\rangle \\ &= \langle Z - \nu \nabla J(W + \nu Z), \nabla J(W + \nu Z)\rangle + \langle -\nabla J(W + \nu Z), Z\rangle \\ &= -\nu \left|\nabla J(W + \nu Z)\right|^2. \end{aligned}$$ □

Many variations are possible. Here we consider another type of friction

$$t > 0 : \quad \begin{cases} W'(t) = Z(t), \\ Z'(t) = -\nabla J(W(t)) - \nu \frac{Z(t)}{\sqrt{|Z(t)|^2 + \varepsilon}} \end{cases} \tag{5.21}$$

where the parameter $\varepsilon > 0$ insures that the system is non degenerate at $Z = 0$. It is immediate to show that $\frac{d}{dt}\left(J(W(t)) + \frac{1}{2}|Z(t)|^2\right) = -\nu \frac{|Z(t)|^2}{\sqrt{|Z(t)|^2 + \varepsilon}} \leq 0$. Another possibility is to use relaxation under the form

$$t > 0 : \quad \begin{cases} W'(t) = Z(t), \\ Z'(t) = -\frac{1}{\varepsilon}\left(\nabla J(W(t)) + Z(t)\right). \end{cases} \tag{5.22}$$

where $\varepsilon > 0$ is now the relaxation parameter.

Lemma 5.2.4. *Solutions of (5.22) satisfy $\frac{d}{dt}\left(J(W(t)) + \frac{\varepsilon}{2}|Z(t)|^2\right) = -|Z(t)|^2 \leq 0$.*

Proof. Evident. □

More complex systems are possible. We present a different method with an additional moment D which is here a diagonal matrix

$$t>0: \qquad \begin{cases} W'(t) = D(t)^{-1/2}Z(t), \\ Z'(t) = -\frac{1}{\varepsilon_1}\left(\nabla J(W(t)) + Z(t)\right), \\ D'(t) = \frac{1}{\varepsilon_2}\left(\operatorname{diag}\left(\nabla J(W(t)) \otimes \nabla J(W(t))\right) - D(t)\right) \end{cases} \tag{5.23}$$

It will be shown that a natural discretization of this system yields a numerical method which is close to the popular **ADAM algorithm** which was published in 2015 [48]. The analysis below is a simplified version of Barakat and Bianchi [7]. We notice 2 different parameters $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$. For a square matrix M, the notation diag(M) indicates the diagonal square matrix with the same diagonal coefficients as M. The square root of D is naturally the diagonal matrix with diagonal elements equal to the square root of the diagonal elements of D. This is correctly defined provided all diagonal elements of D are non negative, which is the case provided the initial data is chosen correctly.

Lemma 5.2.5. *Assume $D(0) > 0$. Then $D(t) > 0$ for all $t \geq 0$.*

Proof. One has the identity $\left(e^{t/\varepsilon_2}D(t)\right)' = e^{t/\varepsilon_2}\operatorname{diag}\left(\nabla J(W(t)) \otimes \nabla J(W(t))\right) > 0$. So $e^{t/\varepsilon_2}D(t) > D(0)$ which yields the claim. □

Lemma 5.2.6. *Assume $D(0) > 0$ and $\varepsilon_1 \leq 4\varepsilon_2$. Then solutions of (5.23) satisfy*

$$\frac{d}{dt}\left(J(W(t)) + \frac{\varepsilon_1}{2}\left\langle D(t)^{-1/2}Z(t), Z(t)\right\rangle\right) \leq 0 \quad \text{for all } t \geq 0. \tag{5.24}$$

Proof. We will make use the formula $\frac{d}{dt}M(t)^{-1} = M(t)^{-2}M'(t)$ where $M(t)$ is a square diagonal differentiable matrix. The variation of the total energy is

$$\begin{aligned} E'(t) &= \left\langle D(t)^{-1/2}Z, \nabla J(W)\right\rangle \\ &\quad - \left\langle \left(\nabla J(W(t)) + Z(t)\right), D(t)^{-1/2}Z\right\rangle \\ &\quad - \frac{\varepsilon_1}{4\varepsilon_2}\left\langle D(t)^{-3/2}\left(\operatorname{diag}\left(\nabla J(W(t)) \otimes \nabla J(W(t))\right) - D(t)\right)Z, Z\right\rangle \\ &= -\left(1 - \frac{\varepsilon_1}{4\varepsilon_2}\right)\left\langle Z(t), D(t)^{-1/2}Z\right\rangle \\ &\quad - \frac{\varepsilon_1}{4\varepsilon_2}\left\langle D(t)^{-3/2}\operatorname{diag}\left(\nabla J(W(t)) \otimes \nabla J(W(t))\right)Z, Z\right\rangle. \end{aligned}$$

Under the condition $\varepsilon_1 \leq 4\varepsilon_2$, then $E'(t) \leq 0$ since it is the sum of two non positive terms. □

Discretization with a natural procedure yields the discrete system

$$\begin{cases} \dfrac{W^{n+1}-W^n}{\Delta t} = \left(D^n\right)^{-\frac{1}{2}} Z^n, \\ \dfrac{Z^{n+1}-Z^n}{\Delta t} = -\dfrac{1}{\varepsilon_1}\left(\nabla J(W^n) + Z^n\right), \\ \dfrac{D^{n+1}-D^n}{\Delta t} = \dfrac{1}{\varepsilon_2}\left(\mathrm{diag}\left(\nabla J(W^n)\otimes \nabla J(W^n)\right) - D^n\right) \end{cases} \tag{5.25}$$

For comparison with the ADAM algorithm [48], let us denote $\theta^n = W^n$, $V^n = -Z^n$ and $\widehat{V}^n = D^n$. One rewrites (5.25) as

$$\begin{cases} V^{n+1} = \beta_1 V^n + (1-\beta_1)\nabla J(W^n), & \beta_1 = 1 - \frac{\Delta t}{\varepsilon_1}, \\ \widehat{V}^{n+1} = \beta_2 V^n + (1-\beta_2)\ \mathrm{diag}\left(\nabla J(W^n)\otimes \nabla J(W^n)\right), & \beta_2 = 1 - \frac{\Delta t}{\varepsilon_2}, \\ \theta^{n+1} = \theta^n - \alpha\left(\widehat{V}^n\right)^{-\frac{1}{2}} V^n, & \alpha = \Delta t. \end{cases} \tag{5.26}$$

The full ADAM algorithm uses additional rescaling steps. It writes

$$\begin{cases} V^{n+1} = \dfrac{1}{1-(\beta_1)^n}\left(\beta_1 V^n + (1-\beta_1)\nabla J(W^n)\right), & \beta_1 = 1 - \frac{\Delta t}{\varepsilon_1}, \\ \widehat{V}^{n+1} = \dfrac{1}{1-(\beta_2)^n}\left(\beta_2 V^n + (1-\beta_2)\ \mathrm{diag}\left(\nabla J(W^n)\otimes \nabla J(W^n)\right)\right), & \beta_2 = 1 - \frac{\Delta t}{\varepsilon_2}, \\ \theta^{n+1} = \theta^n - \alpha\left(\widehat{V}^n\right)^{-\frac{1}{2}} V^n, & \alpha = \Delta t. \end{cases} \tag{5.27}$$

Remark 5.2.7. *The condition $\varepsilon_1 \le 4\varepsilon_2$ becomes*

$$4(\beta_1 - 1) \le \beta_2 - 1.$$

It is instructing to observe that the default values of ADAM are $\beta_1 = 0.9$ and $\beta_2 = 0.999$, which are compatible with the inequality. The value [48] of the third parameter is $\alpha = 0.001$ which indicates that α behaves a priori like a small time step [7]. One obtains the numerical values of the relaxation times

$$\varepsilon_1 = \frac{\Delta t}{1-\beta_1} = 0.01 \text{ and } \varepsilon_2 = \frac{\Delta t}{1-\beta_2} = 1.$$

For large $n \gg 1$, then the rescaling stages are negligible $(\beta_{1,2})^n \approx 0$. Clearly $(\beta_1)^n$ tends to zero much faster than $(\beta_2)^n$.
Since the condition $\varepsilon_1 \le 4\varepsilon_2$ is verified for ADAM, we infer that the continuous ADAM method is endowed with the monotone behavior (5.24). It is an open problem to evaluate how the stability properties of the continuous ADAM in this range of parameters can be an help for the convergence study of the fully discrete ADAM. As quoted in the original publication [48], it is found empirically that ADAM consistently outperforms other methods for a variety of models and datasets. ADAM is nowadays becoming a de-facto standard.

There is a possibility to interpret ADAM as an asymptotic kind of **LASSO** procedure. The LASSO was proposed in [95]. It consists to introduce a penalization term in some given functional where the penalization term is the L^1 norm of the unknown. The idea is that the L^1 norm can be efficient to favor sparsity of the coefficients. Here we make the simple remark that the asymptotic value of the quantity (5.24) which is minimized by the system of ODEs (5.23) ressembles

$$E \approx J(W) + \frac{\varepsilon_1}{2} |\nabla J(W)|_1 \tag{5.28}$$

where $|U|_1 = |u_1| + \cdots + |u_q|$ is the L^1 norm of the vector $U = (u_1, \ldots, u_q) \in \mathbb{R}^q$. Indeed the asymptotic tendency of (5.23) is to enforce $Z \approx -\nabla J(W)$ and $D \approx \operatorname{diag}(\nabla J(W) \otimes \nabla J(W)) \approx \operatorname{diag}(Z \otimes Z)$. For $Z \in \mathbb{R}^q$, it is evident that $\left\langle \operatorname{diag}(Z \otimes Z)^{-\frac{1}{2}} Z, Z \right\rangle = \sum_{i=1}^q \frac{z_i^2}{\sqrt{z_i^2}} = |Z|_1$.
That is why an interpretation of the ADAM method is that it tends to minimize the LASSO-type functional (5.28) at the limit $t \to \infty$.

```
1: Python-Keras-Tensorflow Initialization
2: model.compile(loss='categorical_crossentropy',
3:           optimizer=Adam(beta_1=0.9,beta_2=0.6))
```

Algorithm 8: Compilation of a model with the ADAM optimizer. Here $\beta_1 = 0.9$ and $\beta_2 = 0.6$ which is not the default.

5.3 Batches

The issue is when $\#(\mathcal{D}) \gg 1$ is large, because the numerical cost of the calculation of ∇J may be too high because of the sum (5.1) or (5.15) over all elements in the dataset $\mathcal{D}$. The method of **batches**, which is a **stochastic** decomposition of the dataset, is the standard answer to to issue. This method can be justified with statistical or probabilistic ideas [74] out of the scope of this text. The presentation given below focuses on the interpretation of batches as a particular splitting algorithm.