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

$$E \approx J(W) + \frac{\varepsilon_1}{2} \left|\nabla J(W)\right|_1 \tag{5.28}$$ eq:llas

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 $t \to \infty$.

?⟨algoadam⟩?
```
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

⟨sec:bababa⟩ 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.

5.3.1 A stochastic steepest gradient method

A batch decomposition of the dataset $\mathcal{D}$ is defined as

$$\mathcal{D} = \bigcup_{r=1}^{p} \mathcal{B}_r \quad \text{with } \mathcal{B}_r \bigcap \mathcal{B}_s = \emptyset \text{ for } r \neq s,$$

where these batches are decided **at random** at all global iterations called **epoch**. The index of the global iteration is noted $k = 0, 1, \ldots$.
At each epoch k, one obtains a sequence of batches

$$\mathcal{D} = \bigcup_{r=1}^{p} \mathcal{B}_k^r \quad \text{with } \mathcal{B}_k^r \bigcap \mathcal{B}_k^s = \emptyset \quad \text{for } r \neq s.$$

One writes

$$J(W) = \sum_{r=1}^{p} J_k^r(W), \quad \text{where } J_k^r(W) = \sum_{(x,y)\in\mathcal{B}_k^r} \varphi_y(f(x, W)).$$

Let us decide arbitrarily of a small time step $\Delta t_k > 0$. One can solve (5.3) with a a splitting method which consists to introduce time steps $t_{k+1} = t_k + \Delta t_k$ and to write the stochastic steepest gradient method as

$$\begin{array}{rl} \text{firstly} : & \left\{ \begin{array}{l} Z_k^0(1) = W_k \\ (Z_k^1)'(t) = -\nabla J_k^1(Z_k^1(t)), \quad 0 < t \leq \Delta t_k, \end{array} \right. \\ r = 2, \ldots, p : & \left\{ \begin{array}{l} Z_k^r(0) = Z_k^{r-1}(\Delta t) \\ (Z_k^r)'(t) = -\nabla J_k^r(Z_k^r(t)), \quad 0 < t \leq \Delta t_k, \end{array} \right. \\ \text{finally} : & W_{k+1} = Z_k^p(\Delta t). \end{array} \tag{5.29}$$

eq:wnb

A fully discrete version with variable time steps writes

$$\begin{array}{rl} \text{firstly} : & Z_k^1(0) = W_k \\ r = 2, \ldots, p : & Z_k^r(\Delta t) = Z_k^{r-1}(0) - \Delta t_k \nabla J_k^r(Z_k^{r-1}(0)), \\ \text{finally} : & W_{k+1} = Z_k^p(\Delta t_k). \end{array}$$

The idea is easily generalized kind of gradient method, in particular for the stochastic extension of the ADAM algorithm (5.26).

5.3.2 Convergence

An interesting question is to determine condition such that a local minimum W_* of the function J is still correctly captured at the limit of the splitting method. We

add the very important requirement that the limit must still be the correct one whatever are the batches. Since the batches are decided at random at all t_n, the answer could as well be negative.

It appears that if one makes additional local assumptions which are reasonable in a vicinity of W_*, then the limit is independent of the batches. The local assumptions are that

$$\varphi_{y_i}(f(x_i, W)) = \alpha_i \left| Wx_i - y_i \right|^2, \qquad \alpha_i > 0, \quad i = 1, \ldots, \#(\mathcal{D}). \tag{5.30}$$ {?}

One may as well introduce the bias by considering in the cost function $\widehat{W}\widehat{x} = Wx + b$ instead of Wx. We use Wx just to simplify the notations. The fundamental ideas of the result below are quite similar to the ones by Turinici [98], and in particular the condition (5.31) on the time steps is the same. The result is a deterministic proof of convergence of the stochastic steepest gradient method (one can compare with [34, 25, 33] and references therein).

⟨theor:tii⟩ **Theorem 5.3.1.** *Assume the hypotheses of Lemma 1.2.1 are satisfied, in particular the one which ensures the uniqueness of the minimizer W_*. Assume*

$$\lim_{k \to \infty} \Delta t^k = 0 \text{ and } \sum_k \Delta t^k = \infty. \tag{5.31}$$ eq:timi

Then the splitting system admits a limit which is the correct one, that is $\lim_{k\to\infty} W_k = W_*$*, and this is independent of the batches.*

Remark 5.3.2. *In other words, the stochastic gradient method where the batches at decided at random behaves at the limit like a standard non stochastic steepest gradient method. This holds under the condition (5.31).*

Proof. In order to make a clear distinction between the notation for transposition of matrices M^t and the notation for the time variable of the continuous algorithm (5.29), the time variable is noted differently as $\tau \geq 0$.

• With the notations of the proof of Lemma 1.2.1 and formula (1.11), the global minimum W_* satisfies the equation $\nabla J(W_*) = 0$ that is

$$\sum_i \alpha_i A_i W_*^t = \sum_i x_i \otimes y_i \in \mathcal{M}_{mn}(\mathbb{R}), \qquad A_i = x_i \otimes x_i \in \mathcal{M}_m(\mathbb{R}),$$

written also as $AW_*^t = b$ with $A = \sum_i \alpha_i A_i$ and $b = \sum_i x_i \otimes y_i$.

• A preliminary remark is that the gradient descent (5.3) without splitting can be rewritten as $\frac{d}{dt} W^t(\tau) = -A(W^t(\tau) - W_*^t)$. The general solution is written with a matrix exponential

$$W^t(\tau) = \left(I_m - e^{-A\tau} \right) W_*^t + e^{-A\tau} W^t(0)$$

where I_m is the identity matrix in $\mathcal{M}_m(\mathbb{R})$.

• Next one analyzes the steps of the splitting method (5.29). For the batch $\mathcal{B}_k^r$ in the time interval $[t_k, t_{k+1}]$, one can determine a minimizer W_{kr} solution of

$$A_k^r (W_k^r)^t = b_{kr}, \quad \text{where } A_k^r = \sum_{(x_i,y_i)\in\mathcal{B}_k^r} \alpha_i A_i \text{ and } b_k^r = \sum_{(x_i,y_i)\in\mathcal{B}_k^r} x_i \otimes y_i.$$

Since W_k^r is bounded (consequence of the analysis on the least square method), then by varying along the finite number of possible batches, one gets a uniform bound

$$\sup_{kr} \|W_k^r\| \leq C. \tag{5.32}$$?eq:teci?

One can write $\frac{d}{dt}(Z_k^r)^t(\tau) = -A_k^r((Z_k^r)^t(\tau) - (W_k^r)^t)$. The solution is

$$(Z_k^r)^t(\Delta t_k) = e^{-A_k^r \Delta t_k} \left((Z_k^r)^t(0) - (W_k^r)^t \right) + (W_k^r)^t$$

$$= \left(I_m - e^{-A_k^r \Delta t_k} \right) (W_k^r)^t + e^{-A_k^r \Delta t_k} (Z_k^r)^t(0).$$

• So one has

$$W_{k+1}^t = \sum_{r=0}^{p} \left(\Pi_{s=r+1}^p e^{-A_k^r \Delta t_k} \right) \left(I_m - e^{-A_k^r \Delta t_k} \right) (W_k^r)^t + \left(\Pi_{r=0}^p e^{-A_k^r \Delta t_k} \right) W_k^t$$

where the order in the multiplication of matrices matters

$$\Pi_{s=r}^p e^{-A_k^s \Delta t_k} = e^{-A_k^p \Delta t_k} \dots e^{-A_k^r \Delta t_k}$$

because the matrices have no reason to commute a priori.

• As a preliminary remark, one notices that $\sum_{r=0}^p A_k^r = A$. Since all matrices are symmetric non negative, one gets a bound

$$\|A_k^r\| \leq C = \|A\|, \quad \forall k, r.$$

Also one has $\sum_{r=0}^p A_k^r (W_k^r)^t = AW^t$. One has

$$e^{-A_k^s \Delta t_k} = I_m - \Delta t_k A_k^s + O(\Delta t_k^2).$$

So one can write

$$W_{k+1}^t = \Delta t_k \sum_s A_k^s W_{ks}^t + \left(I_m - \Delta t_k \sum_s A_k^s \right) W_k^t + O(\Delta t_k^2),$$

that is

$$W_{k+1}^t - W_*^t = (I_m - \Delta t_k A) (W_k^t - W_*^t) + O(\Delta t_k^2),$$

where indices of the intermediate steps $0 \leq s \leq p$ do not show up.

• Denote $\lambda > 0$ the smallest eigenvalue of the positive matrix A. Then $a_k = \|W_{k+1}^t - W_*^t\|$ satisfies $a_{k+1} \leq (1 - \Delta t_k \lambda) a_k + C \Delta t_k^2$. So $a_k = \Pi_{q=0}^{k-1}(1 - \Delta t_q \lambda) a_0 + C \sum_{q=0}^{k-1} \Pi_{q=1}^{k-1}(1 - \Delta t_q \lambda) \Delta t_q^2$. Since $1 - \Delta t_k \lambda \leq e^{-\Delta t_k \lambda}$, one obtains

$$a_k \leq e^{-T_k \lambda} a_0 + C \sum_{q=0}^{k-1} e^{(T_q - T_k)\lambda} (T_{q+1} - T_q)^2$$

where $T_q = \sum_{p=0}^{q-1} \Delta t_p$ is the total time at step $q \geq 1$ (so $T_0 = 0$).

• The second part of hypothesis (5.31) implies that the first term tends to 0

$$\lim_{k \to \infty} e^{-T_k \lambda} a_0 = 0.$$

• To analyze the second term, let us define the bounded function

$$\begin{array}{rccl} s: & [0, \infty) & \to & (0, \infty) \\ & \tau & \mapsto & s(\tau) = \Delta t_{q+1} \text{ for } T_q \leq \tau < T_{q+1}, \end{array}$$

where the first part of hypothesis (5.31) implies that $s(t) \to 0$ as $t \to \infty$. Then one can write

$$\sum_{q=0}^{k-1} e^{(T_q - T_k)\lambda} (T_{q+1} - T_q)^2 \leq \int_0^{T_k} e^{\lambda(t - T_k)} s(\tau) d\tau = \int_0^{T_k} e^{-\lambda \tau} s\left(T_k - \tau\right) d\tau$$

The positive function under the integral is bounded by $Ce^{-\lambda t}$ which is integrable, and it tends to zero for all as $s\left(T_k - \tau\right) \to 0$ for $T_k \to \infty$. The Lebesgue dominated convergence Theorem yields that the integral tends to zero, so $\lim_{k \to \infty} \sum_{q=0}^{k-1} e^{(T_q - T_k)\lambda} (T_{q+1} - T_q)^2 = 0$. So the error a_k ends to zero, which is the claim.

• This result is independent of the batches because the constant $\lambda > 0$ is uniform with respect to random procedure used to determine the batches. □

5.4 Initialization

The initialization of $W(0)$ (and all other similar parameters) is of course necessary to use an iterative of gradient method. In classical algorithms in numerical analysis such as the conjugate gradient method [68], one usually initializes the conjugate gradient with the zero default value. In our case, it would correspond to take the initial value $W(0) = 0$ for the steepest gradient method(5.2). But on the contrary, random initialization is strongly suggested for ML algorithms [33]. This feature is