Statistical Learning : Project

Subject 8 : Adam : A method for Stochastic Optimization

Brieuc Antoine Dit Urban
Yves Leconte
Ivan Svatko
Benjamin Dahan-Monsonego
David Belo Nassauer
Anastasia Schenckery

Prof : Yating Liu

2022 — 2023

1 Introduction

1.1 Gradient descent algorithms

Gradient descent algorithms are widely used in numerical analysis to minimize differentiable convex functions. The underlying idea of these algorithms is to move iteratively towards the steepest direction of the slope, with an intensity proper to each algorithm. Here is a first example of a gradient descent algorithm.

Algorithm 1 Gradient Descent

Require: $F(\cdot)$: derivative of the function we want to minimize
Require: x_0 : Initial value
Require: α : Learning rate
Require: ϵ : Threshold
Require: $itemax$: Maximum number of steps performed
$x \leftarrow x_0$
$step \leftarrow 0$
while $\|F(x)\| > \epsilon$ and $step <$ itemax **do**
    $x \leftarrow x - \alpha \cdot F(x)$
    $step \leftarrow step + 1$
end while
return x

In our course of statistical learning, gradient descent algorithms are used to minimize cost functions. Let $\mathcal{D}_N = \{(X_i, Y_i), 1 \leq i \leq N\}$ be a training data set and $\widehat{f}_{w,b}(x) = \frac{1}{1+\exp(-(w^T x+b))}$ a prediction function in the context of logistic regression such that $\mathbb{P}(Y = 1|X = x) = \widehat{f}_{w,b}(x)$. We defined a cost function which is differentiable

$$\mathcal{R}^{D_N}\left(\widehat{f}_{w,b}\right) := -\sum_{1\leq i\leq N}\left(Y_i \log\left(\widehat{f}_{w,b}(X_i)\right) + (1-Y_i)\log\left(1-\widehat{f}_{w,b}(X_i)\right)\right).$$

As this cost function is strictly convex in (w, b), thus the minimum is unique. Here is an implementation of algorithms allowing to do a gradient descent that finds this minimum.

Algorithm 2 Sigmoid

Require: x : Input variable
return $1/(1 + \exp(-x))$

Algorithm 3 Prediction

Require: X : Input data
Require: w : Weight vector
Require: b : Bias term
return SIGMOID$(X \cdot w + b)$

Algorithm 4 Cost Log Likelihood

Require: X : Input data
Require: y : Target variable
Require: w : Weight vector
Require: b : Bias term
return $-y \cdot \log(\textsc{Prediction}(X, w, b)) - (1 - y) \cdot \log(1 - \textsc{Prediction}(X, w, b))$

Algorithm 5 Logistic Regression Gradient Descent

Require: X : input data in $\mathbb{R}^p$
Require: y : input labels
Require: α : Learning rate
Require: $threshold$: Threshold for the gradients
Require: $itemax$: Maximum number of steps
$w, b \leftarrow 0_{\mathbb{R}^p}, 0$
$step \leftarrow 0$
while $step < itemax$ **do**
    $a \leftarrow \textsc{Prediction}(X, w, b) - y$
    $grad_w \leftarrow X^T \cdot a$
    $grad_b \leftarrow \sum a$
    if $\|grad_w\| < threshold$ and $|grad_b| < threshold$ **then**
        break
    end if
    $w \leftarrow w - \alpha \cdot grad_w$
    $b \leftarrow b - \alpha \cdot grad_b$
    $step \leftarrow step + 1$
end while
return w, b

However, if the data set we consider is large, the computation of the gradients might take a while. One way to deal with that is to use random smaller samples of the data set for each step of the gradient descent instead of the whole data set.

Let $\mathcal{D}_N = \{(X_i, Y_i), 1 \leq i \leq N\}$ be a training data set. A mini-batch $\mathcal{B}_M \subset \mathcal{D}_N$ is a subset of M samples drawn randomly from the training data set. This approach offers the advantage of computational efficiency, as well as the ability to escape local minima due to the inherent noise in the estimation of the gradient. The size of the mini-batch (i.e., the number of samples M) is a hyperparameter of the algorithm that must be tuned. Here is an implementation the stochastic gradient descent as seen in course.

Algorithm 6 Stochastic Gradient Descent (SGD)

Require: X : input data in $\mathbb{R}^p$
Require: y : input labels
Require: α : Learning rate
Require: $threshold$: Threshold for the gradients
Require: $itemax$: Maximum number of steps
Require: M : Size of each batch
$w, b \leftarrow 0_{\mathbb{R}^p}, 0$
$costsSGD \leftarrow \text{CostLogLikelihood}(X, y, w, b)$
$step \leftarrow 0$
while $step < itemax$ **do**
    $ind \leftarrow (M\, random\, data\, points\, chosen\, among\, all\, available\, data)$
    $a \leftarrow \text{Prediction}(X[ind], w, b) - y[ind]$
    $grad_w \leftarrow X[ind]^T \cdot a$
    $grad_b \leftarrow \sum a$
    if $\|grad_w\| < threshold$ and $|grad_b| < threshold$ **then**
        break
    end if
    $w \leftarrow w - \alpha \cdot grad_w$
    $b \leftarrow b - \alpha \cdot grad_b$
    $costsSGD \leftarrow costsSGD + \text{CostLogLikelihood}(X, y, w, b)$
    $step \leftarrow step + 1$
end while
return w, b

The goal of our project is to explain the work of *Diederick P. Kingma* and *Jimmy Lei Ba*, who have introduced a noteworthy improvement to Stochastic Gradient Descent the *Adam* method. Before delving into their contribution, lets provide a small insight on the research conducted in this field.

1.2 Context and research in the sector

The idea behind stochastic approximation can be found for the first time in the Robbins-Monro algorithm, from the 1950s [Robbins and Monro, 1951]. Stochastic gradient descent has become important with the rise of machine learning. Ever since, many improvements to the original algorithm have been proposed. In particular, for machine learning applications, setting an appropriate learning rate as a hyperparameter of the model is challenging. If this parameter is not properly chosen, the algorithm can diverge (if the parameter is too high) or be too slow to converge (if the parameter is too low). Various methods have been developed to counter this. Notably, *AdaGrad* (Adaptive gradient algorithm) was developed by Duchi et al. [2011]. It adjusts the learning rate based on the sparsity of the parameter. This allows the algorithm to work very well for sparse parameters. Another well-known improvement to the SGD algorithm is *RMSProp* which also adapts the learning rate for each of the parameters [Hinton and Tieleman, 2012]. The parameters are updated with momentum on the rescaled gradient.

Adam is an extension of the traditional stochastic gradient descent and offers a new method based on adaptive estimates of first and second order moments. This paper focuses on first-order optimization as higher-order optimization methods are not adapted to the optimization of stochastic objectives with high-dimensional parameter spaces. *Adam* is one of the key algorithms that use stochastic gradient descent to optimize an objective function. This algorithm had a significant influence in the field and inspired multiple other optimization schemes including, for example, Nesterov-enhanced gradients [Dozat, 2016] or schemes based on different interpretations of second-order information [Hu et al., 2019]. Different variants of *Adam* have been developed, notably *Adamax* in the paper by Kingma and Ba

[2014].

2 Some Definitions

Before diving into the explanation of the ADAM method, lets introduce some definitions about mathematical tools used in the paper and required to understand the method.

— *Sparsity* : a sparse gradient is a gradient with many zeros or values without a significant impact.

— *Exponential Moving Average* : a Moving Average is a calculation used to analyze data points by creating a series of averages of different subsets of the full data set. In essence, a moving average smoothens the data, making it easier to spot trends by reducing the 'noise' of daily fluctuations.
An Exponential Moving Average is a type of moving average that gives more weight to the most recent data points, aiming to reduce a lag often found in simple moving averages. It is calculated by applying a weighting factor, which is larger when the data point is more recent.

— *Exponential Decay Rates* : The Exponential Decay Rate is the weighting factor used to calculate the Exponential Moving Average. It can be thought of as the rate at which a data point loses its importance when Exponential Moving Average iis updated with new data subsets.

— *Regret* : Regret is the sum of all the previous difference between the online prediction (prediction determined through the online learning framework which we explain later) and the best fixed point parameter for all of the previous steps ($\theta^* = argmin_{\theta \in \mathbb{X}} \sum_{t=1}^{T} f_t(\theta)$). Mathematically, Regret is defined as $R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^\star)]$

3 ADAM method

The main objective of the *Adam* method is to minimize the expected value of a stochastic function $f(\theta)$ given at its input section. To be more precise, it returns its estimate of the value θ^* that minimizes the expected value of $f(\theta)$. Thus the utility of this algorithm goes beyond our course framework as the "stochasticity" of the function f can be inherent and not necessarily depending only on a hyper-parameter as is the case in our course with the mini-batch method.

For the sake of clarity, and to link this project to our course, we will do the rest of the explanation considering that f is a deterministic, convex and $\mathcal{C}^1$ function and that the "stochasticity" the stochasticity here comes from the use of the mini-batch method when evaluating f at each step. The main difference between the Adam method and the usual stochastic gradient descent (SGD) is that the direction of the gradient descent is not given by one evaluation of the gradient of f but by a function of exponential moving averages of all the gradients considered since the initialisation of the algorithm.

Let us start with a linear comment of the algorithm.

3.1 Linear explanation of the algorithm

First we can see that the inputs of the algorithm are nearly the same that for the usual SGD. The only difference is that *Adam* requires an exponential decay rates β_1 and β_2 that will be used for the computation of the exponential moving average of estimations of the gradient m_t and the square gradient v_t.

Then Lines 1 to 4 are the initialisation of the algorithm. The iteration starts at line 5. Line 6 and 7 are the actualisation of the time t and the current gradient g_t. Line 8 and 9 are dedicated to

Algorithm 7 Adam

Require: $\nabla_\theta f(\cdot)$: gradient of f
Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: θ_0 : Initial parameter vector of $\mathbb{R}^d$
1: $\epsilon = 10^{-8}$
2: $m_0 \leftarrow 0$
3: $v_0 \leftarrow 0$
4: $t \leftarrow 0$
5: **while** θ_t not converge **do**
6: $t \leftarrow t + 1$
7: $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
8: $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
9: $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
10: $\hat{m}_t \leftarrow \frac{m_t}{(1-\beta_1^t)}$
11: $\hat{v}_t \leftarrow \frac{v_t}{(1-\beta_2^t)}$
12: $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{(\sqrt{\hat{v}_t}+\epsilon)}$
13: **end while**
14: **return** θ_t

the computation of the exponential moving average of estimations of the gradient m_t and the square gradient v_t with the new gradient. A good way to visualise that these are indeed exponential moving averages is to write them explicitly :

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ m_t &= \beta_1^2 m_{t-2} + (1-\beta_1)\beta_1 g_{t-1} + (1-\beta_1) g_t \\ m_t &= \cdots \\ m_t &= (1-\beta_1) \sum_{i=0}^{t-1} \beta_1^i g_{t-i} \end{aligned} \qquad \begin{aligned} v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ v_t &= \beta_2^2 v_{t-2} + (1-\beta_2)\beta_2 g_{t-1}^2 + (1-\beta_2) g_t^2 \\ v_t &= \cdots \\ v_t &= (1-\beta_2) \sum_{i=0}^{t-1} \beta_2^i g_{t-i}^2 \quad . \end{aligned}$$

These expressions reveal a problem with m_t and v_t. As they are both initialised at 0, it leads them to be biased towards zero, especially during the first steps, when t is small. Indeed, as we usually take β_1 and β_2 close to 1, we can easily see that the factors of both sums $(1 - \beta_1)$ and $(1 - \beta_2)$ compress the sums to zero when t is small. Lines 10 and 11 resolve this problem adding a correction to this bias. Once again, it is helpful to visualise it explicitly :

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{(1-\beta_1^t)} \\ \hat{m}_t &= \frac{(1-\beta_1)}{(1-\beta_1^t)} \sum_{i=0}^{t-1} \beta_1^i g_{t-i} \end{aligned} \qquad \begin{aligned} \hat{v}_t &= \frac{v_t}{(1-\beta_2^t)} \\ \hat{v}_t &= \frac{(1-\beta_2)}{(1-\beta_2^t)} \sum_{i=0}^{t-1} \beta_2^i g_{t-i}^2 \quad . \end{aligned}$$

We observe the smaller t is, the closer $(1 - \beta_1^t)$ and $(1 - \beta_2^t)$ are respectively close to $(1 - \beta_1)$ and $(1 - \beta_1)$ and thus compensate the "compression toward zero" effect. However, when t becomes large, both quantities approach 1, and the correction disappears.

Finally, line 12 updates the parameter θ by the same way usual SGD method does using the step-size α, but instead of using the current gradient for the direction, it uses the ratio of the corrected estimations of the first and second moment $\hat{m}_t$ and $\hat{v}_t$ obtained before. In the paper, the ratio $\frac{\hat{m}_t}{\hat{v}_t}$ is called the signal-to-noise ratio (SNR). Here we can see that an ϵ is added at the denominator. It is in

order to avoid division by 0 in case of sparsity of the gradient. As in SGD, the algorithm stops when $\theta_t \simeq \theta_{t-1}$ and returns the last version of the parameter.

3.2 Convergence Analysis

To evaluate the performance of various algorithms, their convergence is key. That the algorithm converges towards the true value it is trying to estimate is important for the precision of its results. As gradient descent is used for machine learning with "big data" (large dimension of each observation, a large number of observations, and/or a large dimension of output), optimizing how much computational power is needed is also important. This depends on the convergence speed.

Kingma and Ba [2014] first examine the performance of their algorithm by comparing *Adam* with various other methods including *Adagrad*, SGD (with and without momentum, with Nesterov momentum), RMSProp. They evaluate *Adam* on L2-regularized multi-class logistic regression, multi-layer neural networks, and convolutional neural networks. We have coded and reproduced their graphs, as can be found in the python notebook accompanying our report.

To analyze the convergence of Adam from a theoretical perspective, the authors use the online learning framework developed by Zinkevich [2003]. This framework provides a way to model problems where decisions have to be made before the cost of value of the decision is known. The online learning framework is based on a convex set F and a sequence of unknown cost functions (f_t): $F \longrightarrow \mathbb{R}$. The idea behind the framework is that at each time step t, the online convex programming algorithm (Adam in our case) selects x^t a vector belonging to F the convex set. Once the vector is selected, it receives a cost function c^t through which it is evaluated. In a sense, the online learning framework amounts to choosing a point before knowing which criteria or standard will be used to evaluate it.

As the cost function is unknown and can be any convex function, instead of minimizing it, the framework compares the online prediction to an optimal feasible fixed point. The difference between the best fixed point parameter and the online prediction is called regret, defined as :

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^\star)]$$

To evaluate Adam, the authors of the paper compare Adam's online prediction with $\theta^\star$, defined as the best fixed point parameter from all of the previous steps, namely, $\theta^\star = argmin_{\theta \in \mathbb{X}} \sum_{t=1}^{T} f_t(\theta)$.

In their main theorem, under certain conditions detailed below, Kingma and Ba [2014] show that Adam has a $O(\sqrt{T})$ regret bound and that the average regret of Adam converges, namely that $\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$.

This convergence is shown under the following conditions :

— The cost function f_t has bounded gradients, $||\nabla f_t(\theta)||_2 \leq G$ and $||\nabla f_t(\theta)||_\infty \leq G_\infty$

— The distance between any θ_t generated by Adam is bounded, $||\theta_n - \theta_m||_2 \leq D_2$ and $||\theta_n - \theta_m||_\infty \leq D_\infty$

— β_1 and β_2 satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$

The proof of this theorem, which is given in the appendix of the paper, relies on two Lemmas. The first one states that if we set $g_t = \nabla f_t(\theta_t)$ with $g_{t,i}$ its i^{th} element and $g_{1:t,i}$ as a vector that contains the i^{th} of the gradient over all iteration until t, and if $||g_t||_2 \leq G$, $||g_t||_\infty \leq G_\infty$, then,

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty ||g_{1:T,i}||_2 \quad .$$

The proof of this lemma relies on a basic induction where the main trick is to use the inequality

$$||g_{1:T,i}|| - g_{T,i}^2 + \frac{g_{T,i}^4}{4||g_{1:T,i}||_2^2} \geq ||g_{1:T,i}||_2^2 - g_{T,i}^2 \quad .$$

The second Lemma states that if $\beta_1, \beta_2 \in [0, 1)$ are such that $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$, if $||g_t||_2 \leq G$, $||g_t||_\infty \leq G_\infty$ and setting $\gamma = \frac{\beta_1^2}{\sqrt{\beta_2}}$ then,

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} \leq \frac{2}{1-\gamma} \frac{1}{\sqrt{1-\beta_2}} ||g_{1:T,i}||_2 \quad .$$

Here the proof relies on an application of the first lemma over a upper bound of this sum given by the limit of an arithmetic-geometric series and a development of a separation of the sum and its last term. This lemma and basic properties about convexity finally allow Kingma and Ba to easily prove the theorem on the convergence of the method.

However, the convergence of Adam has been the topic of much debate in academic literature. Later papers show errors in the proof of Adam's convergence. In particular, Reddi et al. [2018] gives an example of a function for which Adam fails to converge and identify a theoretical error in the proof. In practice, this error does not have a widespread of an impact of Adam's performance. Adam remains one of the foundational algorithms based on the stochastic gradient.

4 Illustrations

Here are two plots we obtained in the Jupiter Notebook showing the effiency of the convergence the of *Adam*.

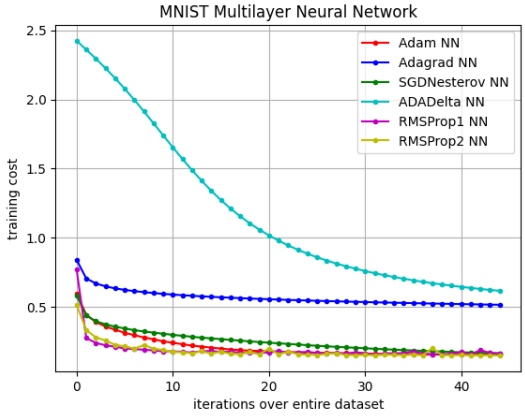

Figure 1 : Logistic regression training negative log likelihood on MNIST images

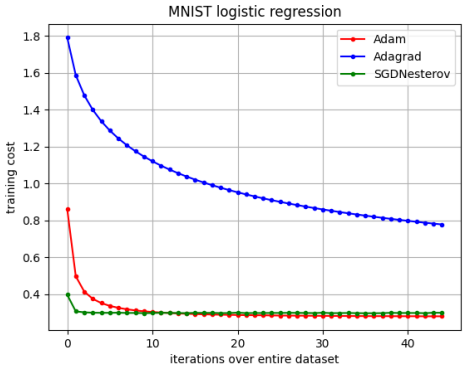

Figure 2 : Training of multilayer neural networks on MNIST images

5 AdaMax

At end the of their paper, Kingma and Ba [2014] give a variation of their algorithm in a method titled *AdaMax*. Fundamentally, the algorithm is the almost the same, the only difference is in the computation of the $\hat{v}$ at lines 9 and 11 of the *Adam* algorithm. While the update rule in *Adam* is based on the L^2 norm (individual weights are updated based on the scaled L^2 norm of the gradients), in *Adamax*, the update rule is generalized and based on an L^∞ norm. Instead of computing a moving average for v, they simply take $max(\beta_2 u_{t-1}, |g_t|)$ where u replaces v and is also initialized to 0. According to the authors, it keeps the stability of the algorithm and can improves its results in several cases.

6 Conclusion

All in all, *Adam* is an optimization algorithm that extends the classic stochastic gradient descent. It has been widely used for deep learning, in particularly for computer vision and natural language processing. By combining the advantages of two pre-exisitng extensions of stochastic gradient descent, namely *Adagrad* and *RMSProp*, this algorithm is part of a larger research topic that examines and tries to optimize stochastic gradient descent. *Adam* has had a significant influence in the field and has inspired other improved algorithms.

Références

T. Dozat. Incorporating nesterov momentum into adam. 2016.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61) :2121–2159, 2011.

G. Hinton and T. Tieleman. Lecture 6.5 - rmsprop, coursera : Neural networks for machine learning. technical report, 2012.

Y. Hu, L. Lin, and S. Tang. Second-order information in first-order optimization methods, 2019.

D. P. Kingma and J. Ba. Adam : A method for stochastic optimization, 2014.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3) :400 – 407, 1951.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. 2003.