Adaptive Stochastic Optimization: AdaGrad and ADAM

Parolo Davide Ruggeri Nicoló

Universitá degli Studi di Padova
MSc Data Science

Introduction and Motivation

In many applications, either coming from Machine Learning or other optimization tasks, we are presented with the issue of solving a maximization/minimization problem in very high dimensional spaces.
While methods like SGD and its reduced variance modifications have been developed, it is argued that in sparse problems these methods are suboptimal, due to the lack of discrimination of the incoming data.
Here we present two of the most used methods at the present time for stochastic optimization, AdaGrad [1] and ADAM [2]. Their main advantage is their adaptive learning rate, that takes into account the geometry of the incoming data, which leads to an increase in performances over very sparse problems, where taking different stepsizes accordingly to the information brought by the data is essential. What the authors report in their original paper about AdaGrad is that "informally, our procedures associate frequently occurring features with low learning rates and infrequent features with high learning rates". This is desiderable to rapidly converge to optimal points, by accelerating over sparse directions, while staying accurate and not overshooting over frequently occurring ones.

The Methods

Both the methods start from the same idea, being ADAM an incorporation of AdaGrad with another stochastic descent algorithm called RMSProp: they gather first order information during the iterations to make an approximate second order normalization of the learning rate, different for every coordinate.

Notation

Let's first introduce some notation: we want to minimize the expected value of the sum of some loss functions f_t, given at time t, by finding its minimum θ^*. Formally we define the regret for the sequence of selected points during the

iterations as

$$R(T) = \sum_{t=1}^{T} f_t(\theta_t) - f(\theta^*)$$

This definition has been introduced as a framework for convergence analysis in [4] and is used in all the methods' original papers.
We also write the *gradients* of the function at every step as

$$\nabla_\theta f_t(\theta_t) := g_t$$

Finally, we denote with $g_{1:t}$ the matrix having as columns the gradients from time 1 to t. With $g_{1:t,i}$ we indicate its i-th row, i.e. the row vector of the i-th entry of every gradient.

Description of the Methods

As outlined, the two algorithms perform a normalization using approximated second order informations. They do so in slightly different ways:

- AdaGrad: gathers second order informations by using an approximation of the correlation matrix $G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^T$ and then updating the current point

  $$\theta_{t+1} = \theta_t - \alpha\, G_t^{-1/2} g_t$$

  To lighten calculation in high dimensional problems the methods actually exploits only the diagonal matrix $diag(G_t)^{-1/2}$. Moreover, to assure invertibility, for the implementation we add a diagonal correction $\delta\mathbb{1}$ (with $\delta > 0$ and $\mathbb{1}$ identity matrix). This leads to the update:

  $$\theta_{t+1} = \theta_t - \alpha\, (\delta\mathbb{1} + diag(G_t))^{-1/2} g_t$$

  Since in the pseudocode we will only work with the diagonal, we call s_t the vector of the diagonal entries of $diag(G_t)$.

- ADAM: similarly to the above, ADAM only updates the entries using the diagonal of the cumulative biased second order matrix, but it uses exponentially decreasing cumulative moments to perform the update. Namely it uses the first and second order moments approximations

  $$m_t := \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

  $$v_t := \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

  where g_t^2 is the elementwise square and $\beta_1, \beta_2 \in [0, 1)$ are two constants determining the speed of the exponential decay of moments away in time. Then the two moments are corrected to obtain unbiasdness and used to perform the update:

  $$\hat{m}_t = m_t / (1 - \beta_1^t)$$

  $$\hat{v}_t = v_t / (1 - \beta_2^t)$$

$$\theta_{t+1} = \theta_t - \alpha \, \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

A difference with respect to AdaGrad is the usefulness of this method also in non stationary setting: the exponential decay of moments far in time assures a good adaptation of the estimation, while in AdaGrad the accumulation of equally weighted descent directions may lead to unclear results.

Why correcting the bias? This a very important feature of ADAM: we want to obtain unbiased estimators for the first and second moment. For the second moment estimate v_t (without considering ϵ; calculations for m_t are similar) we obtain:

$$\begin{aligned}
\mathbb{E}[v_t] &= \mathbb{E}\left[(1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i} g_i^2\right] \\
&= \mathbb{E}[g_t^2](1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i} + \zeta \\
&= \mathbb{E}[g_t^2](1-\beta_2^t) + \zeta
\end{aligned}$$

where $\zeta = 0$ if the true second moment $\mathbb{E}[g_i^2]$ is stationary. Otherwise the authors recommend to choose β_1 so as to keep ζ small.

Note that in both the algorithms α is a fixed initial learning rate, and the ϵ term in the update of ADAM is just a small correction to avoid division by 0.

Here are the two pseudocodes for the methods:

Algorithm 1: AdaGrad. Computes the cumulative correlation matrix and normalizes the learning rate using the diagonal

---

Input: α initial learning rate, $\delta > 0$. Initialize $x_1 = 0$

1 Receive $g_t = \nabla f_t(\theta_t)$, gradient of objective f_t in current point θ_t

2 Update $g_{1:t} = [g_{1:t}\ g_t]$, $s_{t,i} = ||g_{1:t,i}||_2$

3 Call $H_t = \delta \mathbb{1} + diag(s_t)$ ($diag(s_t)$ is the matrix with diagonal equal to s_t and other entries 0)

4 Update current point

$$\theta_{t+1} = \theta_t - \alpha\ H_t^{-1/2} g_t$$

Note that every coordinate i has its own collection of previous gradient values, and has normalization equal to the norm $||g_{1:t,i}||_2$. This leads to a personalized learning rate for every coordinate, and resembles the intuitive meaning we gave in the introduction of "capturing" the frequency and information carried for every coordinate.
ADAM behaves similarly, assigning to every single coordinate two dedicated moment estimates:

Algorithm 2: ADAM. Computes exponentially decaying moments estimaes, the correct their bias and performs the update. Authors suggest the following defualt values for the parameters: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. All operations on vectors are elementwise.

Input: α initial learning rate, $\beta_1, \beta_2, \epsilon$ and θ_0 starting point.
Initialize $m_0 = v_0 = t = 0$

1 Receive $g_t = \nabla f_t(\theta_t)$, gradient of objective f_t in current point θ_t

2 Update moment estimates:

$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
$\hat{m}_t = m_t / (1 - \beta_1^t)$
$\hat{v}_t = v_t / (1 - \beta_2^t)$

3 Update current point

$$\theta_{t+1} = \theta_t - \alpha\ \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

On Convergence of ADAM

Even though a result about the convergence of ADAM has been reported [2] in the original paper, some issues in the proof have been found. Despite the effort put into adjusting it, a recent publication [5] theoretically proved that the method fails in converging even with univariate convex problems.
In the mentioned paper the authors argue that the non convergence comes from the form of exponential decrease presented in ADAM, and present experimental setting to back up their statement. They then proceed proposing a new method, called **AMSGrad**, that preserves the exponential decrease and practical benefits of ADAM and RMSProp, while asuring convergence. The from of the moments update is much similar to ADAM, but assures a slower exponential decrease by normalization (see [5] for more details)

Algorithm 3: AMSGrad. Modification of ADAM and RMSProp, that showed empirical improvements and has theoretically justified convergence

---

Input: $\{\alpha_t\}_{t=1}^{T}, \{\beta_{1t}\}_{t=1}^{T}, \beta_2$.
Initialize $x_0, m_0 = v_0 = \hat{v}_0 = 0$

1 Receive $g_t = \nabla f_t(\theta_t)$, gradient of objective f_t in current point θ_t
2 Update moment estimates:
$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
$\hat{v}_t = \max \hat{v}_{t-1}, v_t$
3 Update current point

$$\theta_{t+1} = \theta_t - \alpha_t \, \hat{m}_t / \sqrt{\hat{v}_t}$$

Experiments

In the experiments section we present a classic logisitic regression setting with:

- cross-entropy loss: given a set of predictions $t_i \in (0,1)$ and a set of labels $y_i \in \{0,1\}$ for $i = 1, \ldots, m$ we define the loss as

$$l(\mathbf{y}) := \frac{1}{m} \sum_{i=1}^{m} y_i \log(t_i) + (1-y_i) \log(1-t_i)$$

- sigmoid link function i.e.

$$t_i = \sigma((w \cdot x_i))$$

where w is the vector of weights to be stochastically optimized, and x_i the i-th sample vector

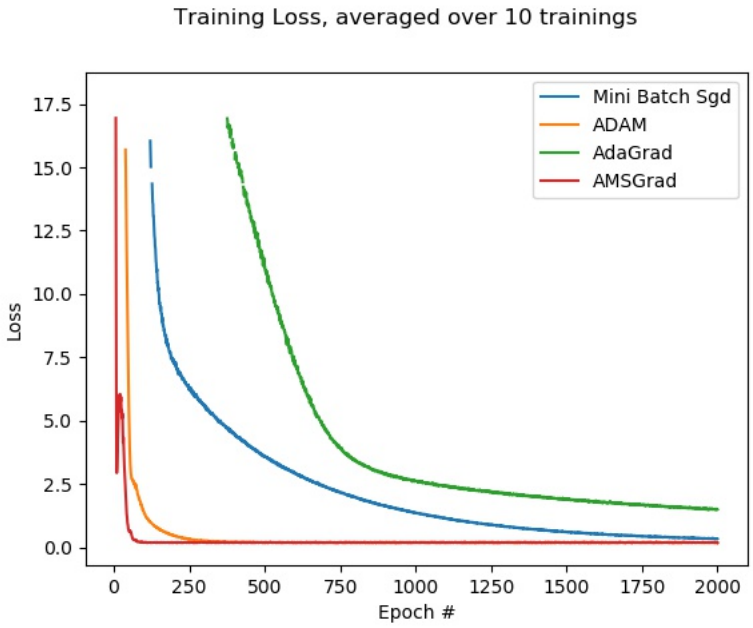

Figure 1: Objective function values of the three methods during training. The three curves are the average of 10 training curves

- l_2 regularization with parameter λ:

$$\frac{\lambda}{2}||w||_2^2$$

Denote with X the matrix containing as rows the vectors x_i, and with $\mathbf{y}, \mathbf{t}$ respectively the vectors of labels and predictions. In this case the (batch) gradient to perform the updates with takes form:

$$g_t = \frac{1}{m} X^T(\mathbf{t} - \mathbf{y}) + \lambda w$$

We mimcked the experiment presented in paper [2], using the MNIST data set to compare the performances of AdaGrad, ADAM and a classic SGD in mini-batches. It is worth highlighting that, despite the unsure theoretical convergence of ADAM, the algorithm has been one of the most used in the latest years, yielding great performances in many cases, including our experiment. We include also AMSGrad for completeness. As suggested in [5], we kept β_1 and $\beta 2$ fixed for simplicity, getting better results than ADAM anyway.

References

[1] Duchi J, Hazan E, Singer Y *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, Journal of Machine Learning Research 12/2011, 2121-2159

[2] Kingma D.P, Lei Ba J *ADAM: a Method for Stochastic Optimization*, Conference Paper at ICLR 2015

[3] Ruder S, *An Overview of Gradient Descent Optimization Algorithms*

[4] Zinkevich M, *Online convex programming and generalized infinitesimal gradient ascent*, 2003

[5] Sashank J.R, Satyen K, Sanjiv K, *On the Convergence of ADAM and Beyond*, Conference Paper at ICLR 2018