Adam (Adaptive Moments) I

- The update rule (Kingma and Ba, 2015)

\[
\begin{align*}
g & \leftarrow \frac{\theta}{C} + \frac{1}{|S|} \nabla_{\theta} \sum_{i:i \in S} \xi(\theta; y^i, Z^{1,i}) \\
s & \leftarrow \rho_1 s + (1 - \rho_1) g \\
r & \leftarrow \rho_2 r + (1 - \rho_2) g \odot g \\
\hat{s} & \leftarrow \frac{s}{1 - \rho_1^t} \\
\hat{r} & \leftarrow \frac{r}{1 - \rho_2^t} \\
\theta & \leftarrow \theta - \frac{\epsilon}{\sqrt{\hat{r}} + \delta} \odot \hat{s}
\end{align*}
\]
Adam (Adaptive Moments) II

- $t$ is the current iteration index
- Roughly speaking, Adam is the combination of
  - Momentum
  - RMSprop
- From Goodfellow et al. (2016),
  \[
  \frac{\epsilon}{\sqrt{\hat{r}} + \delta} \odot \hat{s}
  \]
  (i.e., the use of momentum combined with rescaling) "does not have a clear theoretical motivation"
How about Adam’s practical performance?

From Goodfellow et al. (2016), “generally regarded as being fairly robust to the choice of hyperparameters, though the learning rate may need to be changed from the default”

However, from the web page we referred to for deriving the bias correction, “The original paper ... showing huge performance gains in terms of speed of training. However, after a while people started noticing, that in some cases Adam actually finds worse solution than stochastic gradient”
One example of showing the above is Wilson et al. (2017)
Bias Correction in Adam I

The two steps in Adam

\[ \hat{s} \leftarrow \frac{s}{1 - \rho_1^t} \]
\[ \hat{r} \leftarrow \frac{r}{1 - \rho_2^t} \]

are called “bias correction”

Why do we need this “bias correction”?

Note that \( s \) is the direction used to update \( \theta \).
We hope that its expectation is similar to the expected gradient

$$E[s_t] = E[g_t]$$

and

$$E[r_t] = E[g_t \odot g_t],$$

where $t$ is the iteration index.

The problem is that due to the moving average, the vector is biased toward the initial value.

Note that our initial $s$ is 0.
For $s_t$, we have

\[ s_t = \rho_1 s_{t-1} + (1 - \rho_1) g_t \]

\[ = \rho_1 (\rho_1 s_{t-2} + (1 - \rho_1) g_{t-1}) + (1 - \rho_1) g_t \]

\[ = (1 - \rho_1) \sum_{i=1}^{t} \rho_1^{t-i} g_i \]
Bias Correction in Adam IV

Then

\[
E[s_t] = E[(1 - \rho^1) \sum_{i=1}^{t} \rho_1^{t-i} g_i]
\]

\[
= E[g_t](1 - \rho^1) \sum_{i=1}^{t} \rho_1^{t-i}
\]

Note that we assume

\[
E[g_i], \forall i \geq 1
\]

are the same.
Bias Correction in Adam V

Next,

\[(1 - \rho_1) \sum_{i=1}^{t} \rho_1^{t-i} = (1 - \rho_1)(1 + \cdots + \rho_1^{t-1}) = 1 - \rho_1 \]

Thus

\[E[s_t] = E[g_t](1 - \rho_1^t)\]

and they do

\[\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}\]

The situation for $E[g_t \odot g_t]$ is similar.
Recall in our earlier description, the simple stochastic gradient update is

\[ \theta \leftarrow \theta - \eta \left( \frac{\theta}{C} + \frac{1}{|S|} \nabla_{\theta} \sum_{i:i \in S} \xi(\theta; y^i, Z^{1,i}) \right) \]

In this calculation \( \frac{\theta}{C} \) comes from the regularization term in \( f(\theta) \)

The use of regularization follows from standard machine learning settings
However, in the area of neural networks, this term may come from a setting called weight decay (Hanson and Pratt, 1988)

\[
\theta \leftarrow (1 - \lambda)\theta - \eta \left( \frac{1}{|S|} \nabla \theta \sum_{i:i \in S} \xi(\theta; y^i, Z^{1,i}) \right)
\]

where \( \lambda \) is the rate of weight decay

In fact, Hanson and Pratt (1988) did not give good reasons for decaying the weight of \( \theta \)
Clearly, if

\[ \lambda = \frac{1}{C} \]

then weight decay is the same as regularization

However, as pointed out in Loshchilov and Hutter (2019), the equivalence does not hold if adaptive learning rate is used
For example, in AdaGrad, the update rule is

$$\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\mathbf{r}} + \delta} \odot \left( \frac{1}{|S|} \nabla_{\theta} \sum_{i:i \in S} \xi(\theta; y^i, Z^{1,i}) \right)$$

$$- \frac{\epsilon}{\sqrt{\mathbf{r}} + \delta} \odot \frac{\theta}{\mathbf{C}}$$

so the regularization term is scaled in a component-wise way.

Loshchilov and Hutter (2019) advocate to decouple the weight decay step.
For example, for the momentum algorithm

\[
v \leftarrow \alpha v - \eta \left( \frac{\theta}{C} + \frac{1}{|S|} \nabla_\theta \sum_{i:i \in S} \xi(\theta; y^i, Z^{1,i}) \right)
\]

\[
\theta \leftarrow \theta + v
\]

they prefer the following equivalent form

\[
v \leftarrow \alpha v - \eta \left( \frac{1}{|S|} \nabla_\theta \sum_{i:i \in S} \xi(\theta; y^i, Z^{1,i}) \right)
\]

\[
\theta \leftarrow \theta + v - \eta \frac{\theta}{C}
\]
Based on this, Loshchilov and Hutter (2019) proposed AdamW
\[ g \leftarrow \frac{1}{|S|} \nabla_{\theta} \sum_{i:i \in S} \xi(\theta; y^i, Z^{1, i}) \]

\[ s \leftarrow \rho_1 s + (1 - \rho_1) g \]

\[ r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g \]

\[ \hat{s} \leftarrow \frac{s}{1 - \rho_1^t} \]

\[ \hat{r} \leftarrow \frac{r}{1 - \rho_2^t} \]

\[ \theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\hat{r}} + \delta} \odot \hat{s} - \epsilon \frac{\theta}{C} \]
This is not equivalent to Adam because in Adam, $\theta/C$ has been used in calculating $g$ and then scaled after.

Why is the decoupled setting better? Some discussions are in Section 3 of Loshchilov and Hutter (2019)
Choosing Stochastic Gradient Algorithms

- From Goodfellow et al. (2016), “there is currently no consensus”
- Further, “the choice ... seemed to depend on the user’s familiarity with the algorithm”
In machine learning fast final convergence may not be important

- An optimal solution $\theta^*$ may not lead to the best model
- Further, we don’t need a point close to $\theta^*$. In prediction we find

$$\arg \max_k z_k^{L+1}(\theta)$$

A not-so-accurate $\theta$ may be good enough

An illustration
Why Stochastic Gradient Widely Used? II

Slow final convergence

Fast final convergence
The special property of data classification is essential

\[
E(\nabla_\theta \xi(z^{L+1}; x, y)) = \frac{1}{l} \nabla_\theta \sum_{i=1}^{l} \xi(z^{L+1,i}(\theta); x^i, y^i)
\]

We can cheaply get a good approximation of the gradient

Indeed stochastic gradient is less used outside machine learning
Easy implementation. It’s simpler than methods using, for example, second derivative. Now for complicated networks, (subsampled) gradient is calculated by automatic differentiation.

We will explain more about this.

Non-convexity plays a role.

For convex, other methods may possess advantages to more efficiently find the global minimum.

But for non-convex, efficiency to reach a stationary point is less useful.
A global minimum usually gives a good model (as loss is minimized), but for a stationary point we are less sure.

Some variants of SG have been proposed to improve the robustness or the convergence.

All these explain why SG is popular for deep learning.


