# Adam Performance

Siddhant Chaudhary

CMI, December 2021

# Recall

```
Input: α and β₁, β₂ ∈ [0, 1).
Required: f(θ) (objective) and θ₀ (initial parameter)
m₀ ← 0
v₀ ← 0
t ← 0
while (some convergence criterion on θ_t) do
    t ← t + 1
    g_t ← ∇_θ f_t(θ_{t−1})
    m_t ← β₁ m_{t−1} + (1 − β₁) g_t
    v_t ← β₂ v_{t−1} + (1 − β₂) g_t²
    m̂_t ← m_t / (1 − β₁^t)
    v̂_t ← v_t / (1 − β₂^t)
    θ_t ← θ_{t−1} − α m̂_t / (√v̂_t + ε)
end while
```

**Input**: $\alpha$ and $\beta_1, \beta_2 \in [0, 1)$.

**Required**: $f(\theta)$ (objective) and $\theta_0$ (initial parameter)

$m_0 \leftarrow 0$

$v_0 \leftarrow 0$

$t \leftarrow 0$

**while** (some convergence criterion on $\theta_t$) **do**

  $t \leftarrow t + 1$

  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

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

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

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

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

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

**end while**

# Deriving bias corrections

- Suppose $g_1, ..., g_T$ are the gradients obtained at the time steps; for simplicity assume that each $g_t$ is obtained from the same distribution, i.e

$$\mathbb{E}[g_i] = \mathbb{E}[g_j]$$

for all $1 \leq i, j \leq T$.

# Deriving bias corrections

- Suppose $g_1, \ldots, g_T$ are the gradients obtained at the time steps; for simplicity assume that each $g_t$ is obtained from the same distribution, i.e

$$\mathbb{E}\left[g_i\right] = \mathbb{E}\left[g_j\right]$$

for all $1 \leq i, j \leq T$.

- Expanding the recurrence $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ with the condition $v_0 = 0$, we have the following.

$$v_t = (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} g_i^2$$

# Deriving bias corrections

- Suppose $g_1, ..., g_T$ are the gradients obtained at the time steps; for simplicity assume that each $g_t$ is obtained from the same distribution, i.e

$$\mathbb{E}\left[g_i\right] = \mathbb{E}\left[g_j\right]$$

for all $1 \leq i, j \leq T$.

- Expanding the recurrence $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ with the condition $v_0 = 0$, we have the following.

$$v_t = (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} g_i^2$$

- We want $\mathbb{E}\left[v_t\right]$ to be equal to $\mathbb{E}\left[g_t^2\right]$ (the true second moment).

# (contd.)

- Taking expected values of both sides, we get the following.

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

# (contd.)

- Taking expected values of both sides, we get the following.

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

- So, dividing out by $(1-\beta_2^t)$ does the job.

# (contd.)

- Taking expected values of both sides, we get the following.

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

- So, dividing out by $(1-\beta_2^t)$ does the job.
- Even if the $g_t$s are not sampled from the same distribution, $\beta_2$ is chosen such that the weights assigned to gradients too far in the past are small.

# Convergence Guarantees

- Adam provides guarantees on *regret*, which is defined in the *online convex optimization* framework. Over a time $T$, the regret is defined as follows.

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

# Convergence Guarantees

- Adam provides guarantees on *regret*, which is defined in the *online convex optimization* framework. Over a time $T$, the regret is defined as follows.

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

- Adam has a sublinear regret bound under the following conditions.
  - Bounded gradients: $||\nabla f_t(\theta)||_2 \leq G$, $||\nabla f_t(\theta)||_\infty \leq G_\infty$ for all $\theta$.

# Convergence Guarantees

- Adam provides guarantees on *regret*, which is defined in the *online convex optimization* framework. Over a time $T$, the regret is defined as follows.

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

- Adam has a sublinear regret bound under the following conditions.
  - Bounded gradients: $||\nabla f_t(\theta)||_2 \leq G$, $||\nabla f_t(\theta)||_\infty \leq G_\infty$ for all $\theta$.
  - $||\theta_i - \theta_j||_2 \leq D$, $||\theta_i - \theta_j||_\infty \leq D_\infty$ for all $i, j \in [T]$.

# (contd.)

- (list contd.)
  - $\beta_1, \beta_2 \in [0, 1]$ satisfy $\beta_1^2 < \sqrt{\beta_2}$.

# (contd.)

- (list contd.)
  - $\beta_1, \beta_2 \in [0, 1]$ satisfy $\beta_1^2 < \sqrt{\beta_2}$.
  - Step sizes: $\alpha_t = \frac{\alpha}{\sqrt{t}}$.

# (contd.)

- (list contd.)
  - $\beta_1, \beta_2 \in [0,1]$ satisfy $\beta_1^2 < \sqrt{\beta_2}$.
  - Step sizes: $\alpha_t = \frac{\alpha}{\sqrt{t}}$.
  - $\beta_{1,t} = \beta_1 \lambda^{t-1}$ for some $\lambda \in (0,1)$, i.e the first moment averaging coefficient decays exponentially.

# (contd.)

- (list contd.)
  - $\beta_1, \beta_2 \in [0, 1]$ satisfy $\beta_1^2 < \sqrt{\beta_2}$.
  - Step sizes: $\alpha_t = \frac{\alpha}{\sqrt{t}}$.
  - $\beta_{1,t} = \beta_1 \lambda^{t-1}$ for some $\lambda \in (0, 1)$, i.e the first moment averaging coefficient decays exponentially.
- Under the above conditions, Adam has $O(dG_\infty \sqrt{T})$ regret bound, where $d =$ dimension of the data space.

# Experiment: Fashion MNIST and MNIST

- We compare the performance of Adam and AdaGrad in terms of the convergence in training accuracies on the MNIST and Fashion MNIST datasets.

# Experiment: Fashion MNIST and MNIST

- We compare the performance of Adam and AdaGrad in terms of the convergence in training accuracies on the MNIST and Fashion MNIST datasets.
- For this, we will use *softmax classification*; our output will be a probability distribution $a = (a_1, ..., a_{10})$, each coordinate indicating the likelihood of the sample belonging to a class.

# Experiment: Fashion MNIST and MNIST

- We compare the performance of Adam and AdaGrad in terms of the convergence in training accuracies on the MNIST and Fashion MNIST datasets.
- For this, we will use *softmax classification*; our output will be a probability distribution $a = (a_1, ..., a_{10})$, each coordinate indicating the likelihood of the sample belonging to a class.
- Loss function: *cross entropy loss*. Given a data point $(x, y)$ where $y = (y_1, ..., y_{10})$ is a one-hot encoding of the label,

$$\mathsf{Loss}_{(x,y)}(a) = -\sum_{k=1}^{10} y_i \log(a_i)$$

# 4 Layer NN for MNIST

- For the MNIST dataset, our neural network is 4 layered: the first three layers have $128$ nodes each (with no activation) and the last layer has $10$ nodes with softmax activation. Both Adam and AdaGrad were trained with $10$ iterations and batch size $128$.

# 4 Layer NN for MNIST

- For the MNIST dataset, our neural network is 4 layered: the first three layers have $128$ nodes each (with no activation) and the last layer has $10$ nodes with softmax activation. Both Adam and AdaGrad were trained with $10$ iterations and batch size $128$.
- Adam was trained with $\alpha = 0.001$ (learning rate), $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. These values seem to be the sweet spot (as claimed by the authors).

# 4 Layer NN for MNIST

- For the MNIST dataset, our neural network is 4 layered: the first three layers have $128$ nodes each (with no activation) and the last layer has $10$ nodes with softmax activation. Both Adam and AdaGrad were trained with $10$ iterations and batch size $128$.
- Adam was trained with $\alpha = 0.001$ (learning rate), $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. These values seem to be the sweet spot (as claimed by the authors).
- AdaGrad was trained with learning rate $= 0.001$ and $\epsilon = 10^{-7}$.

# Accuracies

| T | Adam | Adagrad |
|---|---|---|
| 1 | 0.8899 | 0.6364 |
| 2 | 0.9143 | 0.8318 |
| 3 | 0.9160 | 0.8602 |
| 4 | 0.9199 | 0.8751 |
| 5 | 0.9195 | 0.8836 |
| 6 | 0.9196 | 0.8889 |
| 7 | 0.9207 | 0.8935 |
| 8 | 0.9221 | 0.8963 |
| 9 | 0.9223 | 0.8982 |
| 10 | 0.9228 | 0.9000 |

# 2 Layer NN for Fashion MNIST

1. We did the same thing for the Fashion MNIST dataset: in this case we have a $2$ layered neural network, in which the first layer had $128$ nodes with ReLu activation, and the second layer had $10$ nodes with softmax activation. The hyperparameters were the same.

# 2 Layer NN for Fashion MNIST

1. We did the same thing for the Fashion MNIST dataset: in this case we have a $2$ layered neural network, in which the first layer had $128$ nodes with ReLu activation, and the second layer had $10$ nodes with softmax activation. The hyperparameters were the same.
2. We trained the networks for $T = 50$ timesteps.

# 2 Layer NN for Fashion MNIST

1. We did the same thing for the Fashion MNIST dataset: in this case we have a $2$ layered neural network, in which the first layer had $128$ nodes with ReLu activation, and the second layer had $10$ nodes with softmax activation. The hyperparameters were the same.
2. We trained the networks for $T = 50$ timesteps.
3. Again, Adam was much better than AdaGrad: after $50$ iterations, Adam ended up with an accuracy of $0.9606$, while AdaGrad ended up with an accuracy of just $0.8434$!