Adam: A Method for Stochastic Optimization

Daniil Sherki
Egor Cherepanov

Optimization methods, 2023

Introduction

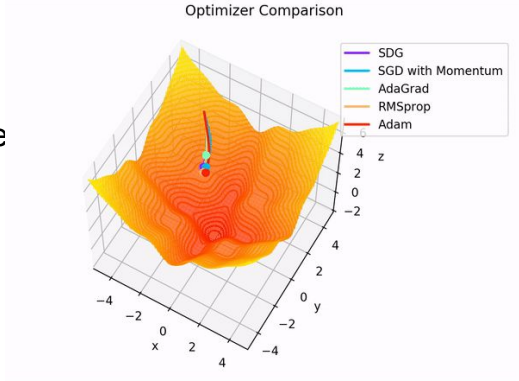
Adam can be looked at as a combination of **RMSprop** and **Stochastic Gradient Descent** with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. Let's take a closer look at how it works.

Adagrad + **RMSProp**

works well with sparse gradients

works well in on-line and non-stationary settings

Adam



Adam's advantages:

- the magnitudes of parameter updates are invariant to rescaling of the gradient;
- stepsizes are approximately bounded by the stepsize hyperparameter;
- does not require a stationary objective;
- works with sparse gradients;
- naturally performs a form of step size annealing.

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
$v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Adam’s update rule. Part 1

Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation.

N-th **moment** of a random variable is defined as the expected value of that variable to the power of n.

$$m_n = \mathbb{E}[X^n]$$

The first moment is mean, and the second moment is uncentered variance.

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

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

m and v are moving averages, g is gradient on current mini-batch, and betas — new introduced hyper-parameters of the algorithm.

Moving averages of gradient and squared gradient.

Adam’s update rule. Part 2

Since m and v are estimates of first and second moments, we want to have the following property:

$$\mathbb{E}[m_t] = \mathbb{E}[g_t]$$
$$\mathbb{E}[v_t] = \mathbb{E}[g_t^2]$$

If these properties held true, that would mean, that we have **unbiased estimators**.

Do not hold true for the our moving averages =(

We can rewrite the formula for our moving average:

$$m_t = (1-\beta_1)\sum_{i=0}^{t}\beta_1^{t-1}g_i$$

Correct the estimator:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Bias correction for the first momentum estimator

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

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

Final expression =)

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Convergence analysis

Theorem 4.1. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$, and β_1, $\beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}$, $\lambda \in (0, 1)$. Adam achieves the following guarantee, for all $T \geq 1$.*

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\widehat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Corollary 4.2. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$. Adam achieves the following guarantee, for all $T \geq 1$.*

$$\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$$

This result can be obtained by using Theorem 4.1 and $\sum_{i=1}^{d} \|g_{1:T,i}\|_2 \leq dG_\infty\sqrt{T}$. Thus, $\lim_{T\to\infty} \frac{R(T)}{T} = 0$.

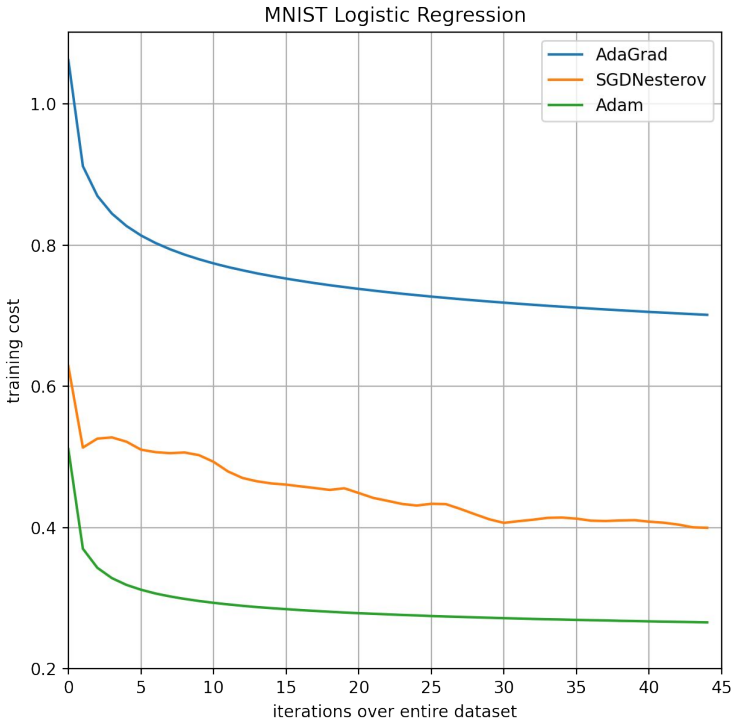
Adaptive

$\mathcal{O}(\log d\sqrt{T})$

Non-adaptive

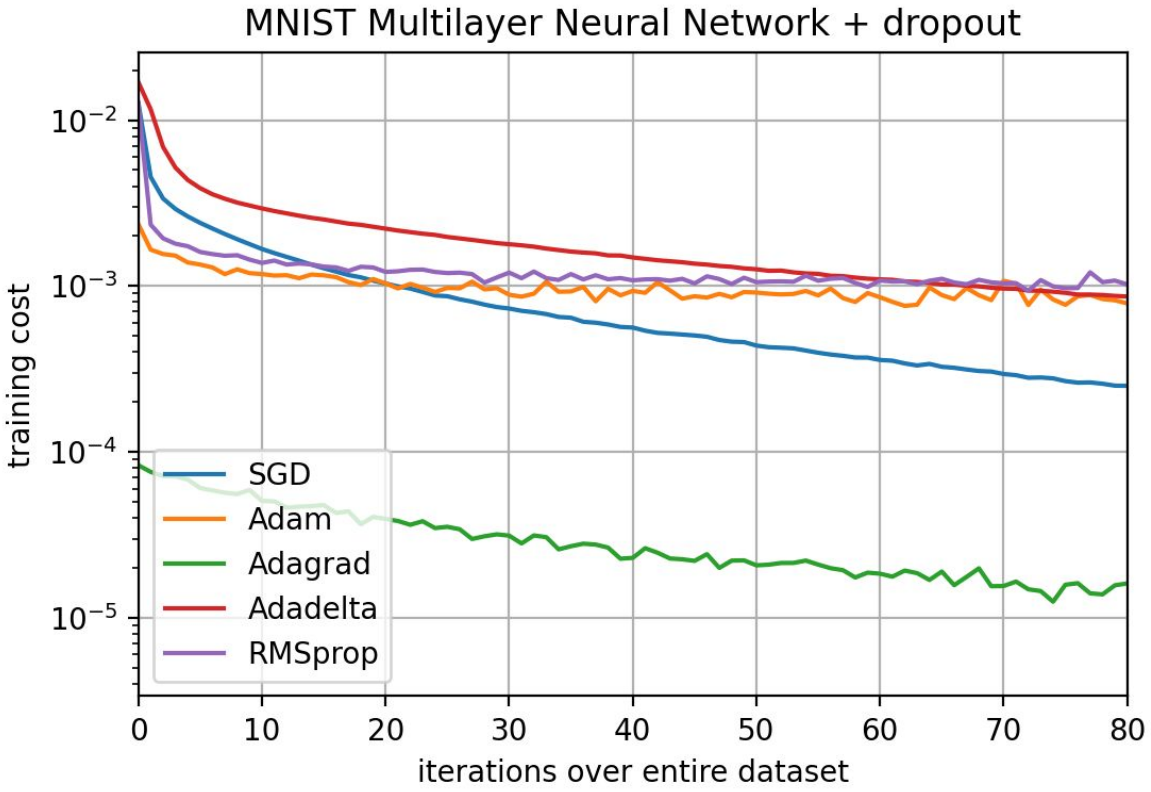
$\mathcal{O}(\sqrt{dT})$

Experiments: Logistic Regression



1. Logistic regression has a well-studied convex objective, making it suitable for comparison of different optimizers without worrying about local minimum issues
2. The logistic regression classifies the class label directly on the 784 dimension image vectors
3. We found that the Adam yields similar convergence as SGD with momentum and both converge faster than Adagrad.

Experiments: Multi-layer Neural Networks



MNN architecture:

- two fully connected hidden layers
- 1000 hidden units each layer
- ReLU activation
- mini-batch size 128

Results:

1. Empirically found that Adam often outperforms other methods in non-convex problems
2. There is no L2-regularization in loss function -> Adagrad with overfitting

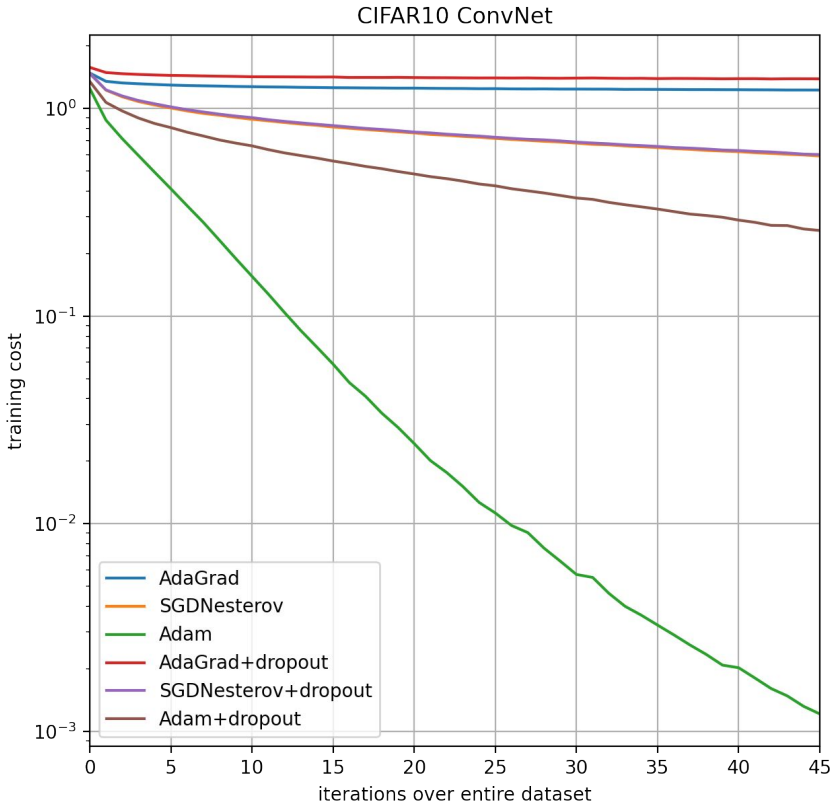
Experiments: Convolutional Neural Networks

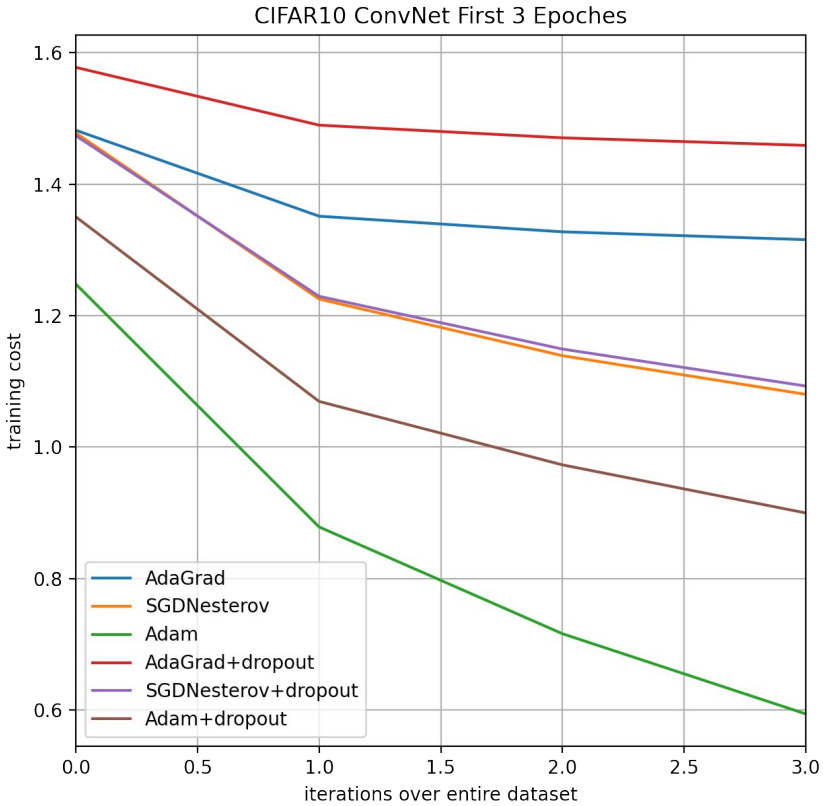
1. Unlike most fully connected neural nets, weight sharing in CNNs results in vastly different gradients in different layers
2. Our CNN architecture has three alternating stages with:
    a. 5x5 convolution filters
    b. 3x3 max pooling with stride of 2

that are followed by a fully connected layer of 1000 rectified linear hidden units (ReLU's).

3. The minibatch size is also set to 128 similar to previous experiments.
4. We show the effectiveness of Adam in deep CNNs.

Experiments: Convolutional Neural Networks





Algorithm 2: *AdaMax*, a variant of Adam based on the infinity norm. See section 7.1 for details. Good default settings for the tested machine learning problems are $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. With β_1^t we denote β_1 to the power t. Here, $(\alpha/(1-\beta_1^t))$ is the learning rate with the bias-correction term for the first moment. All operations on vectors are element-wise.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
$u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$u_t \leftarrow \max(\beta_2 \cdot u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
$\theta_t \leftarrow \theta_{t-1} - (\alpha/(1-\beta_1^t)) \cdot m_t/u_t$ (Update parameters)
end while
return θ_t (Resulting parameters)

Conclusion

1. We have introduced a simple and computationally efficient algorithm for gradient-based optimization of stochastic objective functions.

2. Our method is aimed towards machine learning problems with large datasets and/or high-dimensional parameter spaces.

3. The method combines the advantages of two recently popular optimization methods: the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objectives.

4. The method is straightforward to implement and requires little memory.

5. Overall, we found Adam to be robust and well-suited to a wide range of non-convex optimization problems in the field machine learning.

Thank you for your attention!

GitHub repository