An Investigation of ADAM: A Stochastic Optimization Method

Michael Pérez [1]

Abstract

I propose to review and implement "ADAM: A Method for Stochastic Optimization" (Kingma & Ba, 2014). This paper investigates *Adam*, an algorithm for first-order optimization of stochastic objective functions.

1. Introduction

Stochastic gradient optimization is important in many science and engineering fields. In these fields, problems are commonly framed as minimization or maximization of an objective function with respect to its scalar parameters. If the objective function is differentiable with respect to its parameters, traditional gradient descent is an efficient optimization method. The objective function is often stochastic. Stochastic objective functions can take the form of a sum of subfunctions evaluated on different subsamples of data (minibatches). With a stochastic objective, the efficiency of optimization can be improved by taking gradient steps with respect to individual subfunctions, or stochastic gradient descent (SGD). SGD has been instrumental to recent advances in machine learning (Krizhevsky et al., 2012). Other sources can contribute noise to the objective, such as dropout (Hinton et al., 2012). The optimization of stochastic objective functions with high-dimensional parameter spaces is challenging.

Adam is an efficient first-order optimization algorithm. It computes individual adaptive learning rates from estimates of the first and second moments of the gradients, avoiding expensive higher order computations. The name Adam is derived from adaptive moment estimation. Adam is interesting because it is commonly used today as an alternative to SGD for training deep neural networks. It is useful because it handles sparse gradients, is memory efficient, and is well-suited for non-convex optimization problems.

[*]Equal contribution [1]Computer and Information Science and Engineering Department, University of Florida, Gainesville, USA. Correspondence to: Michael Pérez <michaelperez012@ufl.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Algorithm 1 Adam. All vector operations are element-wise.

Input: α: Stepsize
Input: ϵ: Machine precision threshold
Input: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Input: $f(\theta)$: Stochastic objective function with parameters θ
Input: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd 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)
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$ (Update parameters)
end while
return θ_t (Resulting parameters)

2. Background

Adam is a development that builds upon RMSProp (Tieleman et al., 2012; Graves, 2013) and AdaGrad (Duchi et al., 2011). Adam proceeds as shown in Algorithm 1. Exponentially decaying averages of the gradient (m_t) and squared gradient (v_t) are updated with β_1 and β_2 as hyperparameters that control the decay rate. These averages are estimates of the first (mean) and second moments (uncentered variance) of the gradient. The moving averages are initialized to $\mathbf{0}$'s, which can lead to moment averages that are biased towards zero. This initialization bias is countered by calculating the bias-corrected estimates $\hat{m}_t$ and $\hat{v}_t$ The efficiency of Algorithm 1 can be improved by replacing the last three lines with: $\alpha_t = \frac{\alpha \sqrt{1 - \beta_2^t}}{1 - \beta_1^t}$ and $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t m_t}{\sqrt{v_t} + \hat{\epsilon}}$.

Adam is closely related to RMSProp (Tieleman et al., 2012)

Algorithm 2 AdaMax. All vector operations are element-wise.

Input: α: Stepsize
Input: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Input: $f(\theta)$: Stochastic objective function with parameters θ
Input: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st 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)
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
  $u_t \leftarrow \max(\beta_2 u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
  $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha}{(1-\beta_1^t)} \frac{m_t}{u_t}$ (Update parameters)
end while
return θ_t (Resulting parameters)

and RMSProp with momentum (Graves, 2013), optimization algorithms designed for non-stationary objectives. In RMSProp with momentum the parameters are updated using momentum on the rescaled gradient. In Adam, the updates are directly computed using a running average of first and second moments of the gradient. RMSProp also does not have a bias correction term, which can lead to divergence in some cases.

Adam is also related to AdaGrad (Duchi et al., 2011), an optimization algorithm designed for dealing with sparse gradients. AdaGrad updates the parameters using $\theta_{t+1} = \theta_t - \frac{\alpha g_t}{\sum_{i=1}^t \sqrt{g_t^2}}$ (element-wise division). AdaGrad is a version of Adam with $\beta_1 = 0$, infinitesimally small β_2, and a different step size, $\alpha_t = \frac{\alpha}{\sqrt{t}}$. AdaGrad uses the Mahalanobis norm $\|\theta\|_{G^{(t)1/2}} = \theta^T G^{(t)1/2} \theta$ as the distance metric instead of $\|\theta - \theta^{(t)}\|$. Computing $(G^{(t)})^{-1/2}$ is expensive so a diagonal approximation is used.

AdaMax is an extension to Adam based on the infinity norm. In Adam individual weights are updated by scaling their gradients inversely proportional to a scaled L^2 norm of their current and past gradients. If the update rule based on the L^2 norm is generalized to a rule based on the L^∞ norm, a simple algorithm is produced, AdaMax. AdaMax does not need bias-correction. See Algorithm 2 for pseudocode.

3. Experiments

The algorithms explained will be implemented on different machine learning models to evaluate their efficacy in Python. For each model, I will determine the best hyper-parameter values like the step size and momentum coefficients by using a grid search.

3.1. Adam

3.1.1. Quadratically Regularized Multi-Class Logistic Regression

Algorithm 1 will be implemented on L^2-regularized multi-class logistic regression on the MNIST dataset (Deng, 2012). I will use a decaying step size $\alpha_t = \frac{\alpha}{\sqrt{t}}$. The implementation will solve the following optimization problem:

$$\underset{\Theta}{\text{minimize}} \frac{1}{n} \sum_{i=1}^n (\log(\sum_{c=1}^k \exp x_i^T \theta_c) - x_i^T \theta_{y_i}) + \lambda \|\Theta\|^2$$

3.1.2. Multi-layer Neural Network

Algorithm 1 will be implemented on a multi-layer neural network on the MNIST dataset. The neural network architecture will have two fully connected hidden layers with 1000 hidden units each. The ReLU activation function will be used along with a minibatch size of 128. I will investigate the use of dropout in this experiment because dropout adds stochastic regularization in order to reduce overfitting.

3.1.3. Convolutional Neural Network

Algorithm 1 will be implemented on a convolutional neural network (CNN) on the MNIST dataset. The CNN will have three alternating stages of a 5×5 convolutional layer, a 3×3 max pooling layer with a stride of 2, and a fully connected layer with 1000 ReLU's. I will use a minibatch size of 128. I will compare Adam's performance across the three models described when classifying the MNIST dataset.

3.2. AdaMax

3.2.1. Quadratically Regularized Multi-Class Logistic Regression

Algorithm 2 will be implemented on L^2-regularized multi-class logistic regression on the MNIST dataset. I will compare the classification performance of Adam and AdaMax when used to optimize the same model in this experiment.

References

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgra-

dient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 (61):2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Graves, A. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors, 2012. URL https://arxiv.org/abs/1207.0580.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Tieleman, T., Hinton, G., et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.