Exploration of the Adaptive Moment Estimation (Adam) Optimizer

By
Emmanuel Nyandu Kagarabi &
Hermann Michael Titcho

AIMS | African Institute for Mathematical Sciences SOUTH AFRICA

May 10, 2024

Supervised by Claire David

I. Introduction

Many optimization algorithms are used in Machine Learning to minimize loss & improve model performance:

1. `Gradient Descent (GD)`: Fundamental algorithm, moves in the opposite direction of the gradient for a smooth convex function on the whole dataset. Inefficient in a higher dimensional space, get stuck in a saddle point.
2. `Stochastic Gradient Descent (SGD)` : Randomness, update params via mini-batches sampled from the dataset $->$ Fast but noisier, can escape the minimum.
3. `Adaptive Gradient (Adagrad)`: Adapts the *lr* for each parameter using historical gradient. As *lr* decays over time, it can fail..
4. `Root Mean Square Propagation(RMSProp)`: Improve Adagrad by using moving(exponentially decaying) average, it can fail as Adagrad.
5. `Adaptive Moment Estimation(Adam)` : Combines Adagrad and RMSProp by incorporating moment and adaptive learning rate. It is widely used in DL, it is efficient, in general.

II.1. ADAM Algorithm

Algorithm 1: Adaptive Moment Estimation (ADAM) Optimizer

Input: Learning rate α, β_1, β_2, ϵ, Initial parameters θ
Output: Updated parameters θ

Initialize $m_0 = 0$ (Initial first moment estimate);
Initialize $v_0 = 0$ (Initial second moment estimate);
Initialize $t = 0$ (Initialize timestep);
while *stopping criterion not met* **do**
  $t = t + 1$;
  Compute gradient: $g_t = \nabla_\theta J(\theta_{t-1})$;
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate);
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second moment estimate);
  $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ (Compute bias-corrected first moment estimate);
  $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ (Compute bias-corrected second moment estimate);
  $\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$ (Update parameters);
end
return *Updated parameters* θ;

MACHINE LEARNING BY HANDS : ADAM Project/AIMS SOUTH AFRICA/2024

Figure: $\alpha := 0.001$, $\beta_1 := 0.9$ & $\beta_2 := 0.999$ (decay rates), $\epsilon := 10^{-8}$.

II.2. Experimentation

Data availability

We cpmpared the performance of SGD with Adam by using the MINST dataset ; large collection of written digits (0 - 9), widely used when developing and testing some ML algorithms. The dataset consists of $60,000$ training images and $10,000$ testing images, each one is a (28×28 pixels) gray-image, with pixel values ranging from 0 to 255. All images were converted to the float32 data type with size-normalized values in the range from 0 to 1.

CNN Architecture

We built a CNN with a convolutional layer having 32 filters, each with a 3×3 kernel, followed by a rectified linear unit (ReLU) activation function. Then, a max-pooling layer with a 2×2 pool size is applied to reduce the spatial dimensions. To prevent overfitting, a dropout layer with a rate of 0.25 is added. The output is then flattened into a 1D array and passed through a dense layer with 256 neurons and a ReLU activation function. Another dropout layer with a rate of 0.5 is included to further regularize the network. Finally, a dense layer with 10 neurons and a softmax activation function is used to output the classification probabilities. The model is compiled with categorical cross-entropy loss, the specified optimizer, and accuracy metrics. It is trained for 10 epochs with a batch size of 64, and $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-8$.

Evolution of Loss Functions (SGD vs ADAM)

Evolution of Loss Functions (SGD vs ADAM)

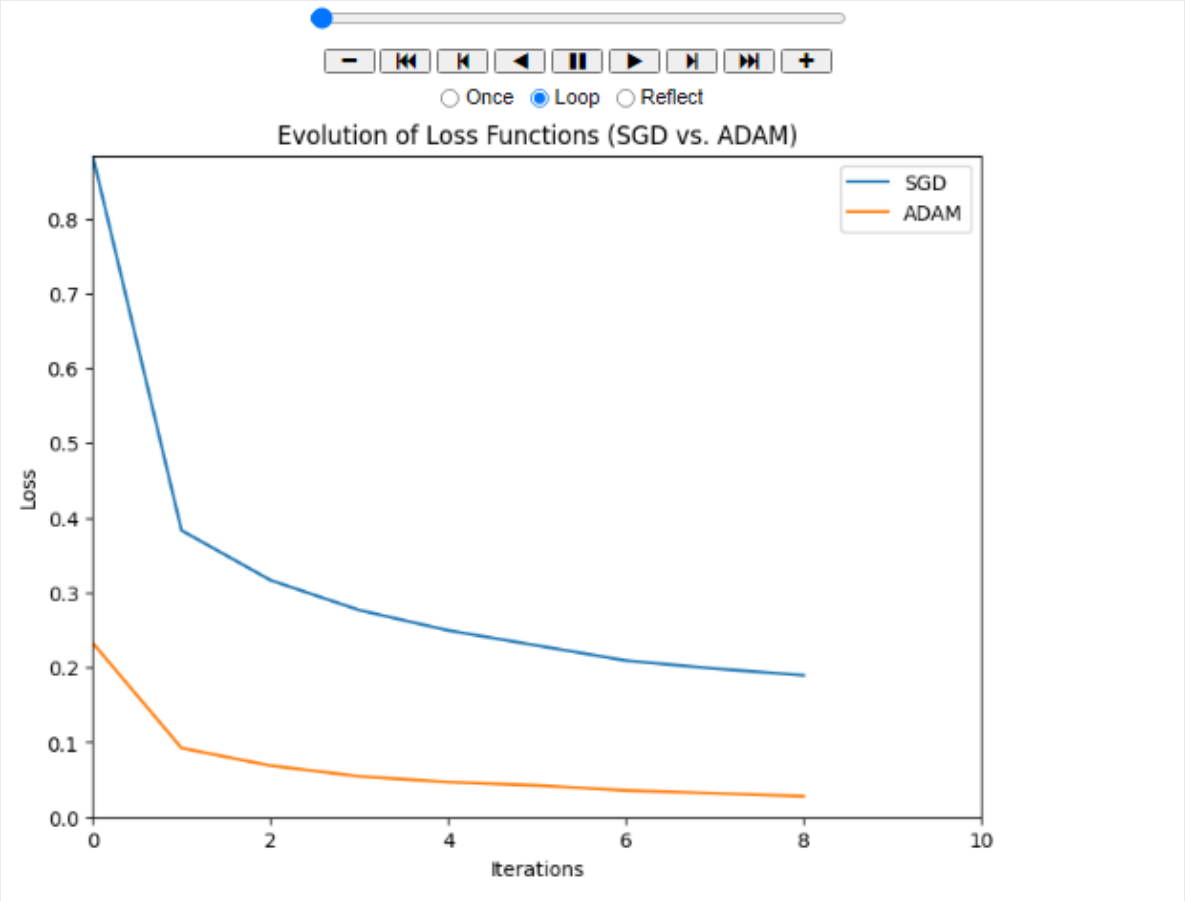

Figure: Evolution of Loss Functions (SGD vs ADAM)

Evolution of Loss Functions (SGD vs ADAM)

Evolution of Accuracy (SGD vs. ADAM)

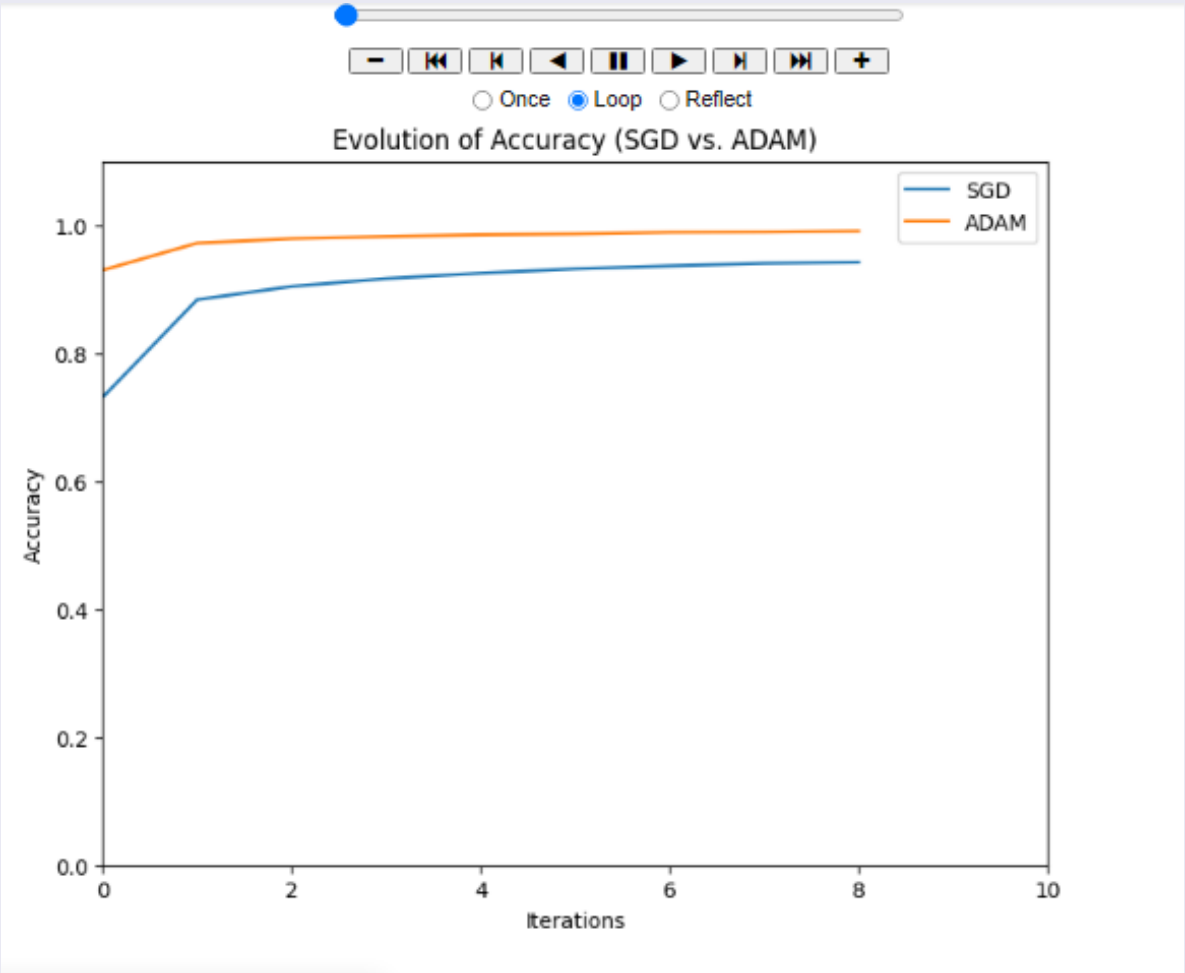

Figure: Evolution of Accuracy (SGD vs. ADAM)

Notebook

ADAM_PROJECT_ORAL_PRESENTATION.ipynb

III. Conclusion

The comparison between the **Adam** and **SGD** optimizers on the MNIST dataset reveals significant differences in their performance. Adam shows an interesting convergence, achieving an impressive accuracy of **99.16%** on the training set and **98.97%** on the validation set within **443.36** seconds. Its ability to adapt learning rates for each parameter dynamically contributes to its efficiency in navigating the complex optimization landscape. On the other hand, SGD, while still effective, demonstrates a given convergence rate, reaching an accuracy of **94.56%** on the training set and **96.88%** on the validation set in **442.87** seconds. Overall, Adam clearly outperforms SGD.

References

- **Diederik P. Kingma** and **Jimmy Lei Ba**. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, 2017.
- **Artem Oppermann**. Optimization in Deep Learning: AdaGrad, RMSProp, ADAM.
- **Mohamed Reyad, and all.** A modified Adam algorithm for deep neural network optimization, 2023.
- **Shiksha**. Adam Optimizer for Stochastic Gradient Descent, 2023.
- **Jason Brownlee**. Gradient Descent With Momentum from Scratch, 2021.
- **Kevin P Murphy**. Probabilistic machine learning : An introduction. MIT press, 2022.

Acknowledgement

☺Thank you for your attention and participation!☺