Project2: Deep learning Optimizers

Group Members:
Paul sanyang
Khadija Iddrisu
Albert Agisha Ntwali
Idriss Nguepi Nguefack

African Institute for Mathematical Sciences, AIMS-Senegal

Supervised by: Tutors

April 29, 2022

Presentation outline

1 Introduction
- Motivation
- Problem
- Objective

2 Methodology

3 Results and discussion

4 Conclusion

Motivation I

- Deep Learning is a sub-field of machine learning that is motivated by how the human brain works.

Motivation II

How Neural Networks work?

Neurons:

Problem I

- We evaluate the performance of neural networks by calculating the loss, i.e the difference between predicted output and actual output
- How do we minimize this loss?

Problem II

OPTIMIZATION!!!

Objective

- Compare methods of optimization (Gradient descent variants and Adaptive Optimizers)
- Pros and cons of each methods
- Compare Results of various optimizers

Gradient Descent Family

Gradient Descent Algorithm iteratively calculates the next point using gradient at the current position, then scales it (by a learning rate "η") i.e

Gradient Descent

$$p_{n+1} = p_n - \eta \nabla f(p_n) \tag{1.1}$$

There are three types of gradient descent

- **Batch gradient descent**
- **Stochastic gradient descent**
- **Mini batch gradient descent**

Drawbacks of base optimizer:(GD, SGD, mini-batch GD)

- **Gradient descent** is computationally expensive
- **SGD** takes a longer time to converge.
- **Mini-batch GD** doesn't use entire records to update parameter and hence, the path to reach global minima is not as smooth as Gradient Descent.

General Challenges of the Gradient Descent Family

1. **Local Minima and Saddle Point**
   They are very efficient for convex problems but performs poorly on non-convex problems
2. **Vanishing and Exploding Gradient**
   The former is encountered when the gradient is smaller, otherwise the latter

Adaptive Learning Methods

- Adaptive learning methods are optimization techniques that accelerates training of deep learning models by adjusting the learning rate based on the model parameters
- The general idea is the use of different learning rates for each parameter after every iteration

WWs of Adaptive Learning Rates?

- Adaptive learning rates can accelerate training and alleviate some of the pressure of choosing a learning rate and learning rate schedule.
- When to use Adaptive learning methods: For faster model training and for more stable models

Adaptive Methods Continued...

- Momentum
- Adagrad
- Adadelta
- RMSProp
- Adam.

Momentum

- The Momentum method is a method to accelerate performance of SGD by smoothing out noise in gradients:

Momentum

- Momentum introduces the use of an **exponentially moving average** (velocity) of the negative gradients

Update rule:

$$w \leftarrow w + v$$

$$v \leftarrow \alpha v - \varepsilon \cdot \nabla_w L(w), \alpha \in [0, 1) \tag{2.1}$$

Adagrad

- The idea behind Adagrad is to use different learning rates
- It is much simple compared to SGD with momentum

Update rule

$$w_t = w_{t-1} - \eta_t^{'} \nabla_w L(w) \quad (2.2)$$

$$b_t = b_{t-1} - \eta_t^{'} \nabla_b L(b) \quad (2.3)$$

Where

$$\eta_t^{'} = \frac{\eta}{\sqrt{\alpha_t + \epsilon}} \; and \; \alpha_t = \sum_{i=1}^{t} (\nabla_w L(w))^2 \quad (2.4)$$

Adagrad

- The idea behind Adagrad is to use different learning rates
- It is much simple compared to SGD with momentum

Update rule

$$w_t = w_{t-1} - \eta_t^{'} \nabla_w L(w) \tag{2.2}$$

$$b_t = b_{t-1} - \eta_t^{'} \nabla_b L(b) \tag{2.3}$$

Where

$$\eta_t^{'} = \frac{\eta}{\sqrt{\alpha_t + \epsilon}} \; and \; \alpha_t = \sum_{i=1}^{t} \left(\nabla_w L(w)\right)^2 \tag{2.4}$$

Adadelta I

The **Adadelta** optimizer is used to solve the two main drawbacks of the **Adagrad** method:

- Continual decay of learning rates throughout training.
- Manual selection of global learning rate.

Adadelta II

- update rule

$$\Delta w_t = -\frac{RMS[\Delta w]_{t-1}}{RMS[g]_t} \cdot g_t \tag{2.5}$$

$$w_{t+1} = w_t + \Delta w_t \tag{2.6}$$

where $RMS[g]_t = \sqrt{E[g^2] + \epsilon}$,

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) \cdot g_t^2 \tag{2.7}$$

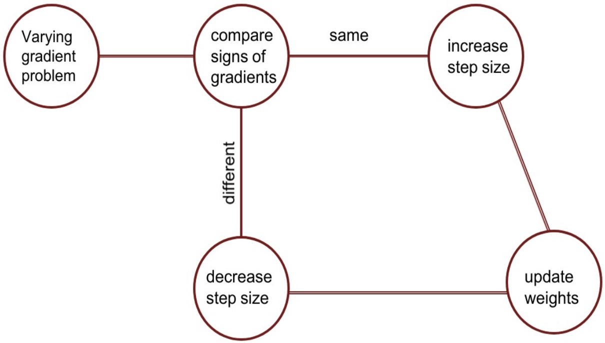
Also $RMS[\Delta w]_t = \sqrt{E[\Delta w^2] + \epsilon}$ and

$$E[\Delta w^2]_t = \rho E[\Delta w^2]_{t-1} + (1-\rho) \cdot \Delta w_t^2 \tag{2.8}$$

with $g_t = \nabla_w L(w)$

RMSProp I

- The Root Mean Squared Prop is an extension of the RPPROP algorithm which is used to solve the problem of varying gradients



RMSProp II

- RMSPROP is used as an alternative to introduce robustness and efficiency of minibatches while accelerating the optimization process
- Here EMA is given by:

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)\big(\nabla_w L(w)\big)^2$$

RMSProp III

Update Rule

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla_w L(w)$$

$$b := b - \frac{\eta}{\sqrt{v(b,t)}} \nabla_b L(b)$$

We can prevent division by zero with:

normalization factor

$$w := w - \frac{\eta}{\sqrt{v(w,t) + \epsilon}} \nabla_w L(w)$$

$$b := b - \frac{\eta}{\sqrt{v(b,t) + \epsilon}} \nabla_b L(b)$$

RMSProp IV

Pros:

- It converges quickly and requires lesser tuning as compared to gradient descent algorithms and their variants.

Cons:

- The problem with RMS Prop is that the learning rate has to be defined manually and the suggested value does not work for every application.

Adam(Adaptive moment estimation) I

- Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

Adam(Adaptive moment estimation) II

- $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ and $\varepsilon = 10^{-8}$, with β_1 & $\beta_2 =$ decay rates of average of gradients in the above two methods.
- $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$, with $g_t = \nabla_w L(w)$
- $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
  m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively
- $\hat{m}_t = \frac{m_{t-1}}{1-\beta_1^t}$
- $\hat{v}_t = \frac{v_{t-1}}{1-\beta_2^t}$

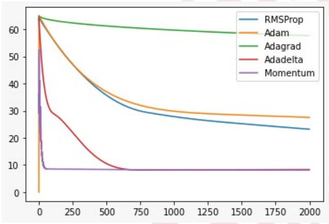
Adam(Adaptive moment estimation) III

- **Updates rule**

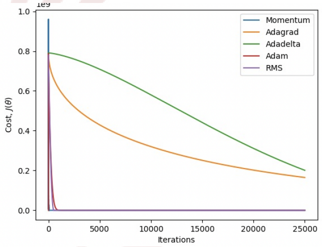
$$w := w - \alpha \frac{\hat{m}}{\sqrt{\hat{v}} + \varepsilon} \tag{2.9}$$

- Adam has some downsides. It tends to focus on faster computation time, whereas algorithms like stochastic gradient descent focus on data points. That's why algorithms like SGD generalize the data in a better manner at the cost of low computation speed.

Results and discussion



(a) Iris data



(b) data

Conclusion

- Adaptive learning optimizers generally accelerate model training in a fast and efficient mannet
- The convergence of the different adaptive optimizers highly depends on the nature of the data and the model
- Adam is mostly used for optimization in wide range of problem in classification, when training deep neural networks.

References

Diederik P. Kingma and Jimmy Lei Ba, ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, ICLR 2015.

Geoffrey Hinton, Neural Networks for Machine Learning,Overview of mini-batch gradient descent

Matthew D. Zeiler, ADADELTA: AN ADAPTIVE LEARNING RATE METHOD,Google Inc., USA 2New York University, USA

THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
BIYAN
SHUKRIA
TASHAKKUR ATU
YAQHANYELAY
SUKSAMA
EKHMET
GRAZIE
MEHRBANI
GOZAIMASHITA
EFCHARISTO
JUSPAXAR
BOLZIN
MERCI
TINGKI
PALDIES