DEEP LEARNING OPTIMIZERS

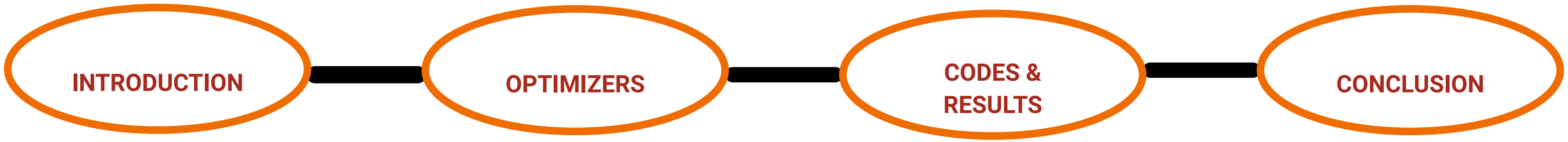
ADAPTIVE LEARNING ALGORITHM

GROUP 2 TEAM

- Aman Kassahun Wassie
- Yaseen Eltahir
- Catherine Monoue
- Ifeoma Veronica Nwabufo

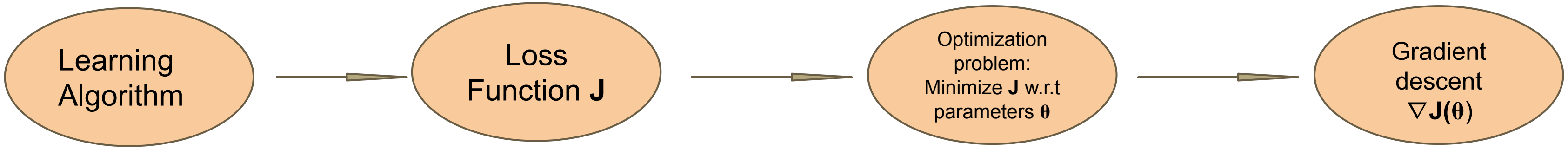
Supervised by: Benjamin Benteke

OUTLINE



INTRODUCTION

Machine Learning formulation:

- Data $\{X, Y\}, \quad X \in \mathbb{R}^{n \times m}, \ Y \in \mathbb{R}^{n}$
- 

- Hypothesis
- Prediction

Gradient Descent Algorithm

Repeat until convergence:

- Compute the gradient of the loss function
- Update the parameters

INTRODUCTION

First generation

1. Batch (Vanilla) gradient descent
2. Stochastic gradient descent
3. Mini batch gradient descent
4. Momentum
5. Nesterov accelerated gradient

Second generation: Adaptive learning

1. Adagrad
2. Adadelta
3. RMSprop
4. Adam

OPTIMIZERS: First Generation

Batch gradient descent (BGD)

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

- Gradient updates after calculating loss of entire training example.
- High resource demands - lots of space in memory.
- Long training time.

- Takes fewer steps to converge.
- Perfect gradient.

Stochastic gradient descent (SGD)

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- Gradient updates after loss for one training example $(x_i, y_i) \in \{X, Y\}$
- Faster training time than BGD.
- Less need for memory.
- Gives quick info about model performance.
- Suitable for online learning.

- Noisy gradient.
- Many steps to converge.

Mini-batch gradient descent (MBGD)

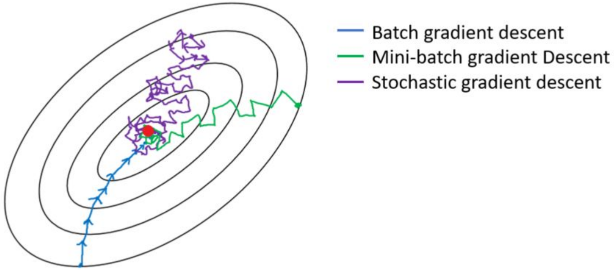
$$\theta = \theta - \eta \cdot \nabla_\theta (\theta; x^{(i:i+k)}; y^{(i:i+k)})$$

- Gradient updates per batch.
- Less noisy than SGD.
- Fewer steps to converge than SGD.

- Optimal batch size may be difficult to get.

OPTIMIZERS: First Generation

Convergence steps



Convergence diagram for BGD, SGD, MBGD

Image Source

Challenges

- ❖ Local minima

- ❖ Plateau, Saddle point
- ❖ How to adjust the learning rate
- ❖ Choice of a proper learning rate
- ❖ Dealing with sparse data

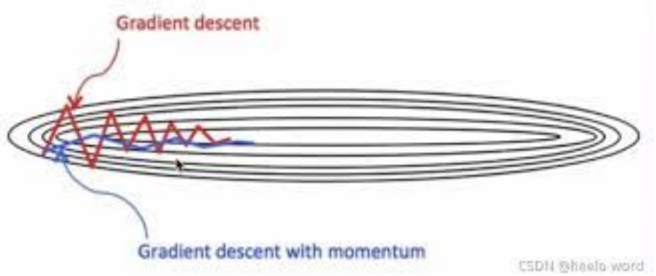
MOMENTUM

- **Our goals**:
  - ❖ We do not want the high oscillations.
  - ❖ We want to move towards the minimum faster.
- Smoothens the noise.
- Gives weight based on the previous steps.

$$v_t = \beta v_{t-1} + (1-\beta)\nabla_\theta J(\theta)$$

$$\theta_t = \theta_{t-1} - \eta v_t$$

where t is the time, β is the momentum term and v_{t-1} is the mean of past gradients. β is usually taken as 0.9.



Gradient Descent with Momentum

Image Source: GD with momentum

OPTIMIZERS: Second Generation

With the momentum, we try to increase the convergence speed and avoid local minima

What about sparsity in data

Keep updating the parameters as there is no difference between their associated features.

Why?

1. **The average gradient for sparse feature is small, so slower rate of training.**
2. **Can end up with saddle point.**

OPTIMIZERS: Second Generation

- Adaptive learning algorithms avoid saddle points.

Definition

Adaptive learning algorithm is an algorithm which tries to adjust the learning rate to the specificities (***frequency)*** of features associated to a parameter.

OPTIMIZERS: Second Generation

Adagrad

Choose the learning rate η

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \quad \textbf{where} \quad G_{t,ii} = \sum_{j \le t} g_{j,i}^2$$

Individualise learning rate

Mesures how much we have moved

Advantage:

- **Deals with sparse features.**

Limits:

- Slow convergence because the learning rate is drastically reduced.
- End up with infinitesimally small learning rate which leads to no learning.

OPTIMIZERS: Second Generation

RMSprop

Choose the learning rate η

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

$$\mathrm{E}[g_i^2]_t = 0.9\mathrm{E}[g_i^2]_{t-1} + 0.1g_{t,i}^2$$

Exponential moving average

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

where $G_{t,ii} = \mathrm{E}[g_i^2]_t$

Advantage:

- **Faster than Adagrad.**

OPTIMIZERS: Second Generation

Illustration of the improvement of Adagrad with RMSprop.

OPTIMIZERS: Second Generation

Adadelta tries to:

- improve Adagrad as RMSprop
- solve some problems with unit

Adadelta

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \quad \text{and} \quad RMS[g]_t = \sqrt{E[g^2]_t + \epsilon}$$

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

Replace the previous learning rate to solve unit issue

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2 \quad \text{and} \quad RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon}$$

Advantages:

- **No need to choose a global learning rate**
- **robust to large sudden gradient**

OPTIMIZERS: Second Generation

Adam wants to:

- **Adapt learning to each feature.**
- **Reduces the noise in the gradient (momentum).**

Adam

Choose the learning rate η

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

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

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

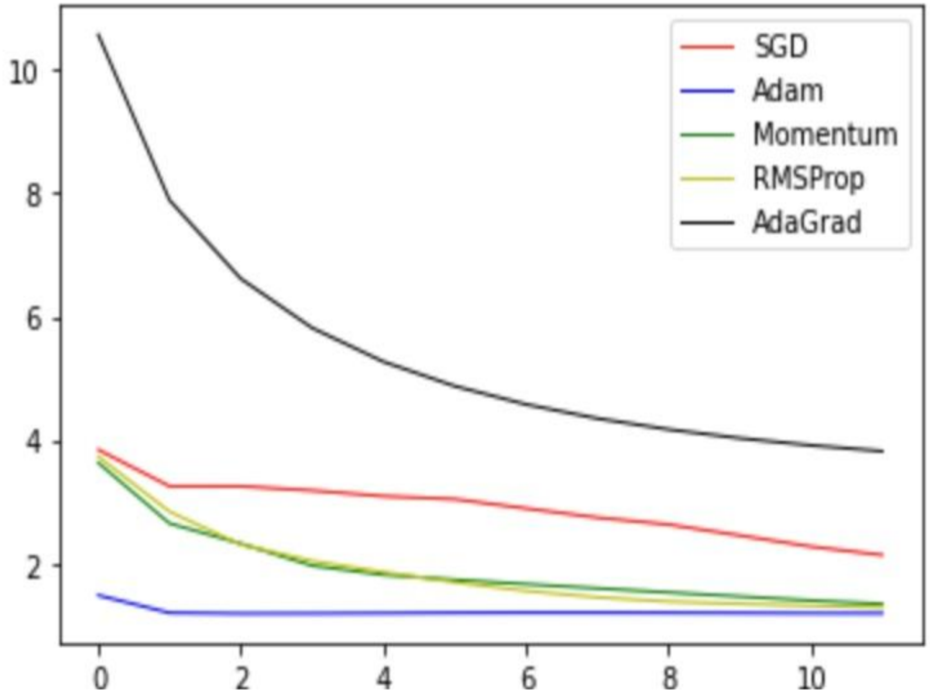
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Advantages:

1. **Succeeds in avoiding local minima.**
2. **Can escape plateau region.**

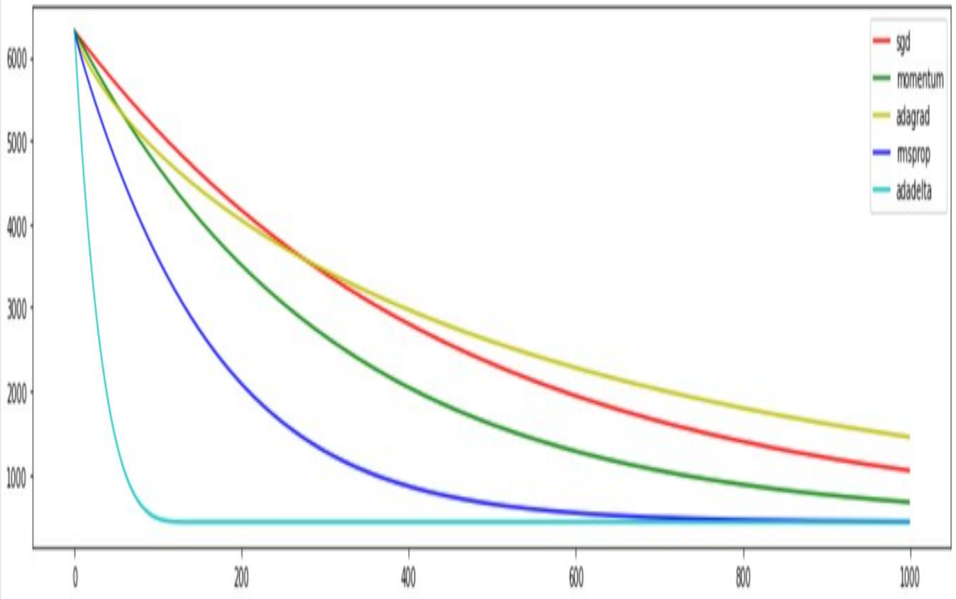
CODES: Results

Dataset
We implement a neural network on MNIST dataset.



CODES: Results

Dataset
We implement a linear regression on random generate data using Sklearn dataset.



CONCLUSION

The optimization step is an important part of Machine Learning program. An importance that we can see through all the proposition make by research.

We can divide the optimizers based on gradient descent in two generations, where the first one is compose with the are using the same learning rate for all the parameters and the second are trying to individualise it.

We saw that the optimizer try to improve speed in convergence, solve the issue with the appropriate learning rate and it scheduling, escape region for local minima, plateau region, and saddle point.

Based on the literature, only the Adam optimizer is able to give the solve all those problems.

BIBLIOGRAPHY

- Diederik P. Kingma, Jimmy Ba: “Adam: A Method for Stochastic Optimization”. ICLR (Poster) 2015
- Matthew D. Zeiler. “Adadelta: An Adaptive Learning Rate”, 2012
- Ashok Cutkosky, Harsh Mehta. “Momentum Improves Normalized SGD”, 2020.
- https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c
- https://www.youtube.com/watch?v=TudQZtgpoHk

Thank you
for your kind attention