# ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

# MID PRESENTATION

Team no - 43

## TABLE OF CONTENTS

## 01
## Introduction and motivation

## Introduction

Optimizers are very important in the training of neural networks. After the advent of backpropagation, a lot of scientists have worked on making better and more robust optimizers, to make the training of neural networks faster and better. Various famous optimizers have been developed so far. Some being -

- SGD
- Momentum
- Nesterov Accelerated GD
- ADAGRAD
- ADADELTA
- RMSprop
- ADAM

We will be talking about ADAM. It was proposed in 2015, and combines the advantages of RMSprop and Adagrad.

## Previous work

Before jumping into ADAM, we will give a brief about other optimizers.

- Momentum - In this optimiser, a term called momentum would is added that accounts for previous steps in the current step. It is such that more recent steps are given an exponentially higher weight than less recent steps.

$$v = \gamma . v + \eta . \nabla_\theta J(\theta)$$
$$\theta = \theta - \alpha\, v$$

- Nesterov Accelerated Gradient - In this one, an acceleration term is added along with momentum. This was done because sometimes outliers in a data set would be completely ignored by the momentum optimiser, simply because the sum of all the steps before it would overpower it. This was solved by adding an acceleration term.

$$v = \gamma . v + \eta . \nabla_\theta J(\theta - \gamma . v)$$
$$\theta = \theta - \alpha\, v$$

## Previous work

Before jumping into ADAM, we will give a brief about other optimizers.

- Adagrad - Previous optimisers had fixed learning rates for each parameter.This optimiser was made to allow for adaptive learning rates for each parameter. That is, with this optimiser, you are creating more degrees of freedom for your learner, as it can have different learning rates for different parameters.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \nabla_{\theta_{t,i}} J(\theta_{t,i})$$

$$G_{t,ii} = G_{t-1,ii} + \nabla^2_{\theta_{t,i}} J(\theta_{t,i})$$

- AdaDelta - The problem with adaGrad is that, is that the G term is monotonically increasing, so eventually, the model learns slower every step it takes, and eventually just stops moving. Ada Delta solves this by added a weight to the G term, that reduces the influence of previous steps in G exponentially. This weight is gamma in the below equations.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E\left[G_{t,ii}\right] + \epsilon}} \nabla_{\theta_{t,i}} J(\theta_{t,i})$$

$$G_{t,ii} = \gamma G_{t-1,ii} + (1 - \gamma) \nabla^2_{\theta_{t,i}} J(\theta_{t,i})$$

## 02

## ADAM

Adaptive Moment Estimation Optimizer

## ADAM

**Adam adds onto AddaDelta by accounting for momentum, but using the expected value of past gradients.**

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[G_{t,ii}] + \epsilon}} \times \mathrm{E}[\mathrm{g}_{t,i}]$$

$$G_{t,ii} = \gamma G_{t-1,ii} + (1-\gamma)\nabla^2_{\theta_{t,i}} J(\theta_{t,i})$$

$$\mathrm{E}[\mathrm{g}_{t,i}] = \beta E[g_{t-1,i}] + (1-\beta)\nabla_{\theta_{t,i}} J(\theta_{t,i})$$

**In essence, Adam uses exponentially moving averages on the gradient and the square of the gradient, which is evaluated on mini-batches. It also adds bias correction to the exponentially moving averages.**

$$\hat{m}_t \leftarrow m_t/(1-\beta_1^t)$$
$$\hat{v}_t \leftarrow v_t/(1-\beta_2^t) \; ($$

03

## DELIVERABLES

What do we plan on doing?

## DELIVERABLES

### ADAM
Creating a class ADAM inheriting `torch.optim.optimizer.Optimizer` from PyTorch

### DATASETS
Creating data-loaders and pre-processors for different datasets IMDB, MNIST, CIFAR10.

### LOGISTIC REGRESSION
Comparing ADAM with previous optimizers on 2 datasets: MNIST and IMDB

### MLP
Comparing ADAM's performance  on MNIST dataset.

### CNN
Running ADAM on the CIFAR10 dataset using a Convolutional Neural Network model

### EXPERIMENTS
Analysing the results and improvements by using different experiments on all models and compare them.

04

## WORK DONE SO FAR

## Paper Review

Initially, during the first 4-5 days, time was spent on understanding the paper and how ADAM optimizer works

## ADAM Implementation

The ADAM optimizer was implemented from scratch by referring to the published algorithm.

## Models for Comparison

The process of developing further models was started. For first, we coded a Logistic Regression model on the MNIST dataset, and compared various gradient optimizers with each other and with ADAM

## 05

## TIMELINE

## Future Timeline

- Test the performance of gradient optimizers on other datasets and models as well
- What we have in mind?
  - CNN
    - CIFAR10
  - Logistic Regression
    - MNIST (done)
    - IMDB
  - MLP
    - MNIST
- Build a visualizer to look at how these gradient optimizers compare across each other.