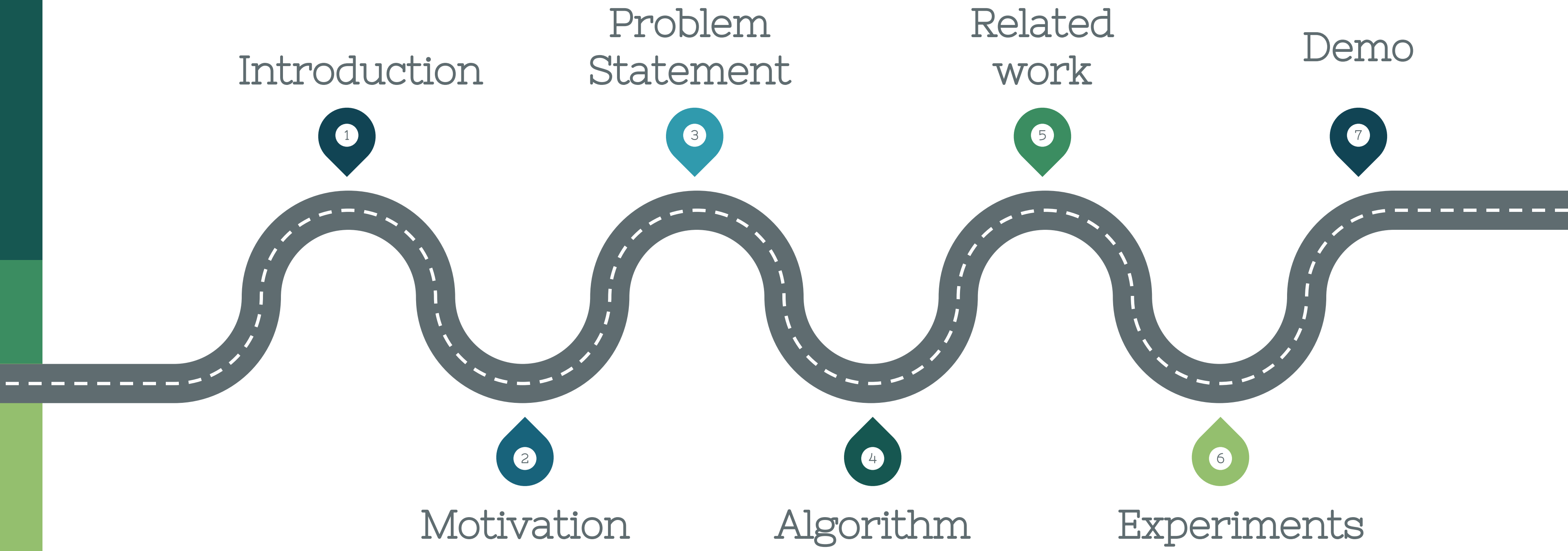
ADAM: A Method For Stochastic Optimization

Presentation By
KAJOL
UB ID: 50478691

Roadmap


1
Introduction
2
Motivation
3
Problem Statement
4
Algorithm
5
Related work
6
Experiments
7
Demo

Introduction

- The background of "ADAM: A Method for Stochastic Optimization" is rooted in the training of artificial neural networks, which is a crucial task in deep learning.
- At the time of the paper's publication, the most widely used optimization algorithms for training deep learning models were gradient descent and its variants, such as momentum and Nesterov acceleration.
- However, these algorithms had limitations, such as the need for careful tuning of the learning rate and difficulty in handling noisy gradients.
- The background of the paper is to address these limitations and provide a new optimization algorithm that can improve the training of deep learning models.

Motivation

- The motivation behind this paper is to address the challenges faced by traditional optimization algorithms in training deep learning models.
- These challenges include slow convergence, sensitivity to the choice of the learning rate, and difficulty in handling noisy gradients.
- The authors aimed to develop a new optimization algorithm that can overcome these challenges and lead to faster and more reliable convergence.
- The ADAM algorithm was introduced as a solution to these challenges by incorporating ideas from adaptive learning rate methods and second-order gradient information.

Problem Statement

- The problem statement is to propose a new optimization algorithm for training deep neural networks.

- ADAM combines the advantages of two popular optimization methods:
  – Root Mean Square Propagation (RMSprop)
  – Adaptive Gradient Algorithm (AdaGrad)
  and handles the challenges of adapting the learning rates for different parameters in a computationally efficient manner.

- The authors aimed to demonstrate that ADAM can effectively optimize complex neural network models, achieve faster convergence and achieve better results compared to other existing optimization methods.

ALGORITHM

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: 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)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

ALGORITHM

- A new optimization algorithm called "Adam" (Adaptive Moment Estimation) for minimizing the expected value of a noisy, differentiable objective function ($f(\theta)$).

- The algorithm estimates the gradient (first moment) and the squared gradient (second raw moment) of the objective function at each timestep using exponential moving averages.

- The hyper-parameters β_1 and β_2 control the exponential decay rates of these moving averages, which are initially set to 0s and can result in biased moment estimates.

- However, the paper outlines how to counteract this initialization bias and obtain bias-corrected estimates.
- The algorithm updates the parameters using the bias-corrected moment estimates, with the learning rate α_t being a function of the moving average parameters.
- The authors note that the efficiency of the algorithm can be improved by changing the order of computation.

ADAM'S UPDATE RULE

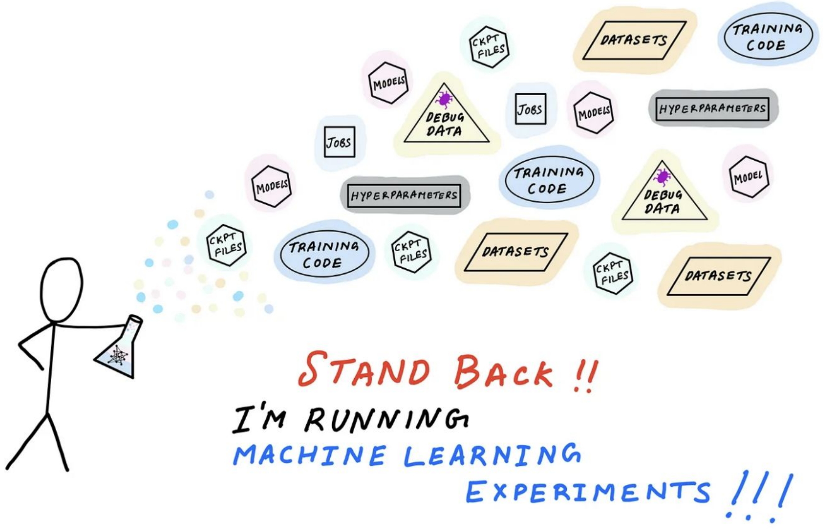
- Adam's update rule carefully chooses stepsizes for optimization
- In less sparse cases, effective step size will be smaller
- Ratio $\widehat{m}_t/\sqrt{\widehat{v}_t}$ is referred to as the signal-to-noise ratio (SNR)
- With a smaller SNR, effective step size is closer to zero
- Smaller SNR means greater uncertainty about gradient direction
- SNR typically becomes smaller towards optimum, leading to smaller effective steps
- Effective step size is invariant to gradient scale

INITIALIZATION BIAS CORRECTION

- Initialization bias correction is used to correct the discrepancy between
  – True second moment of gradient
  – Estimated second moment.
- The exponential moving average of the squared gradient is used to estimate the second raw moment, with a decay rate of β_2.
- If the true second moment is not stationary, the exponential moving average can be kept small by choosing a small value of β_2.
- However, if the gradients are sparse, a reliable estimate of the second moment requires a small value of β_2, which without correction would result in larger initial steps.
- To correct this, the algorithm divides the estimate by $(1 - \beta_t^{2})$, which corrects the initialization bias.

CONVERGENCE ANALYSIS

- In the paper "Convergence Analysis of Adam", the convergence of Adam optimization algorithm is analyzed using the online learning framework proposed by Zinkevich, 2003.
- The goal is to predict the parameter θ at each time t and evaluate it on a previously unknown cost function f_t.
- The evaluation of the algorithm is done using regret, which is the sum of all previous differences between the online prediction $f_t(\theta_t)$ and the best fixed point parameter $f_t(\theta^*)$.
- When the data features are sparse and bounded gradients, the summation term can be much smaller than its upper bound. Finally, the average regret of Adam converges to $O(1/\sqrt{T})$.



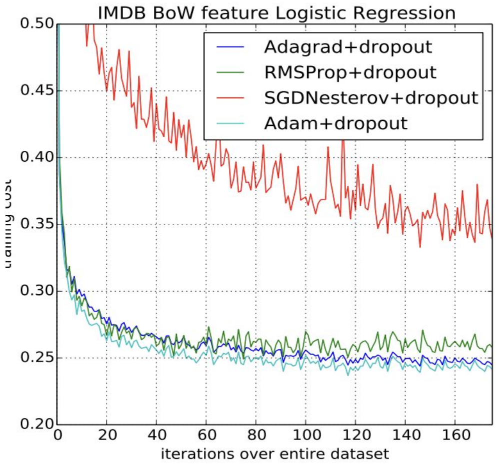
Experiments

Let's start experimenting

EXPERIMENT: Logistic Regression

- Evaluation of the proposed Adam algorithm on L_2-regularized multi-class logistic regression using the MNIST dataset.
- Stepsizes alpha adjusted with $1/\sqrt{t}$ decay in the experiments.
- Comparison of Adam to accelerated SGD with Nesterov momentum and Adagrad, with a minibatch size of 128.
- Adam yields similar convergence as SGD with momentum and both converge faster than Adagrad.

- Examination of the sparse feature problem using the IMDB movie review dataset.
- Adagrad outperforms SGD with Nesterov momentum both with and without dropout noise.
- Adam converges as fast as Adagrad and can take advantage of sparse features.
- Empirical performance of Adam is consistent with the theoretical findings in the paper.



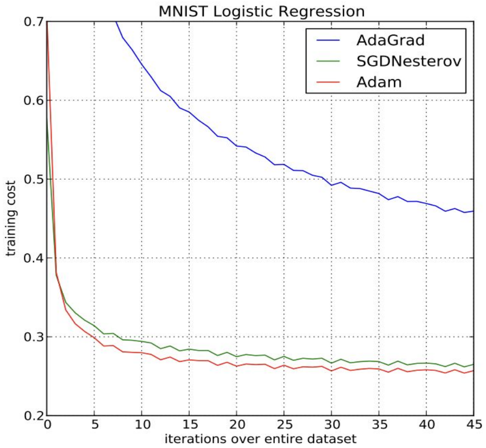


Figure : Logistic regression training negative log likelihood on MNIST images and IMDB movie reviews with 10,000 bag-of-words (BoW) feature vectors.

Experiment: Multi-layer Neural Networks

- Models are powerful but have non-convex objective functions
- Experiment used neural network with 2 fully connected hidden layers with 1000 hidden units each and ReLU activation with minibatch size of 128
- Study of different optimizers with standard deterministic cross-entropy objective and L_2 weight decay to prevent overfitting
- Comparison between Adam and sum-of-functions (SFO) method
- Results show that Adam is faster in terms of iterations and wall-clock time and SFO is slower with linear memory requirement

- Stochastic regularization methods like dropout used to prevent over-fitting
- SFO fails to converge with cost functions with stochastic regularization
- Comparison between Adam and other stochastic first order methods on multi-layer neural networks trained with dropout noise
- Results show that Adam is better in terms of convergence than other methods

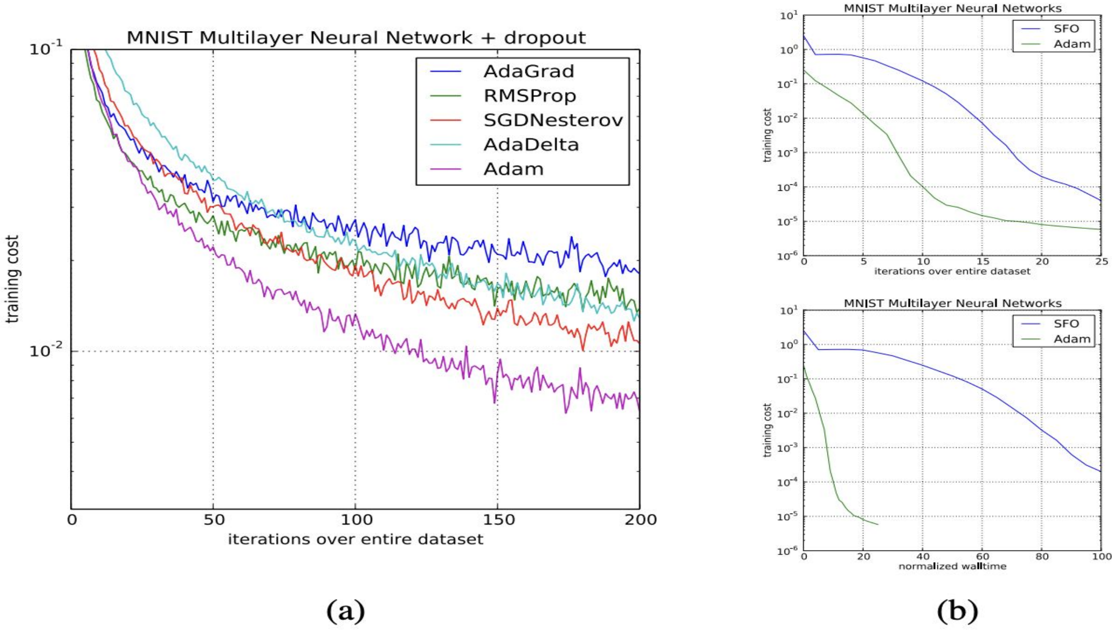


Figure 2: Training of multilayer neural networks on MNIST images. (a) Neural networks using dropout stochastic regularization. (b) Neural networks with deterministic cost function. We compare with the sum-of-functions (SFO) optimizer (Sohl-Dickstein et al., 2014

EXPERIMENT: Convolutional Neural Networks

- Convolutional Neural Networks (CNNs) have demonstrated great success in computer vision tasks.
- Weight sharing in CNNs leads to vastly different gradients compared to fully connected neural networks.
- A smaller learning rate is often used for convolution layers in CNNs.
- The CNN architecture used in the experiment consists of alternating convolution and pooling layers, followed by a fully connected layer.

- Input images are pre-processed by whitening and dropout noise is applied to the input and fully connected layers.
- Although Adam and Adagrad make rapid progress in the initial stage of training, Adam and SGD eventually converge faster.
- The second moment estimate in Adagrad is a poor approximation for the cost function in CNNs.
- Reducing minibatch variance through the first moment is more important in CNNs and contributes to the speed-up.
- Adagrad converges much slower compared to other methods.
- Adam shows marginal improvement over SGD with momentum and adapts the learning rate scale for different layers.

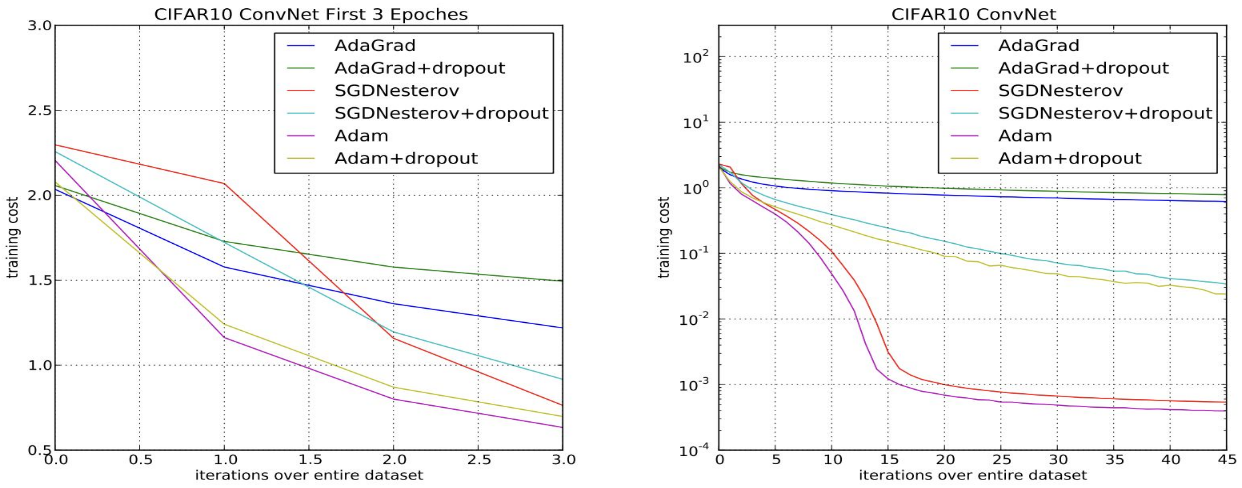


Figure: Convolutional neural networks training cost. (left) Training cost for the first three epochs. (right) Training cost over 45 epochs. CIFAR-10 with c64-c64-c128-1000 architecture.

EXPERIMENT:
Bias-Correction Term

- The experiment evaluates the effect of the bias correction term on training a variational autoencoder.
- The results show that without the bias correction term, training becomes unstable when values of β_2 are close to 1, especially in the early epochs of training.
- The best results were achieved with small values of $(1-\beta_2)$ and with the bias correction term present.
- Adam performed equal or better than RMSProp, regardless of the hyper-parameter setting.

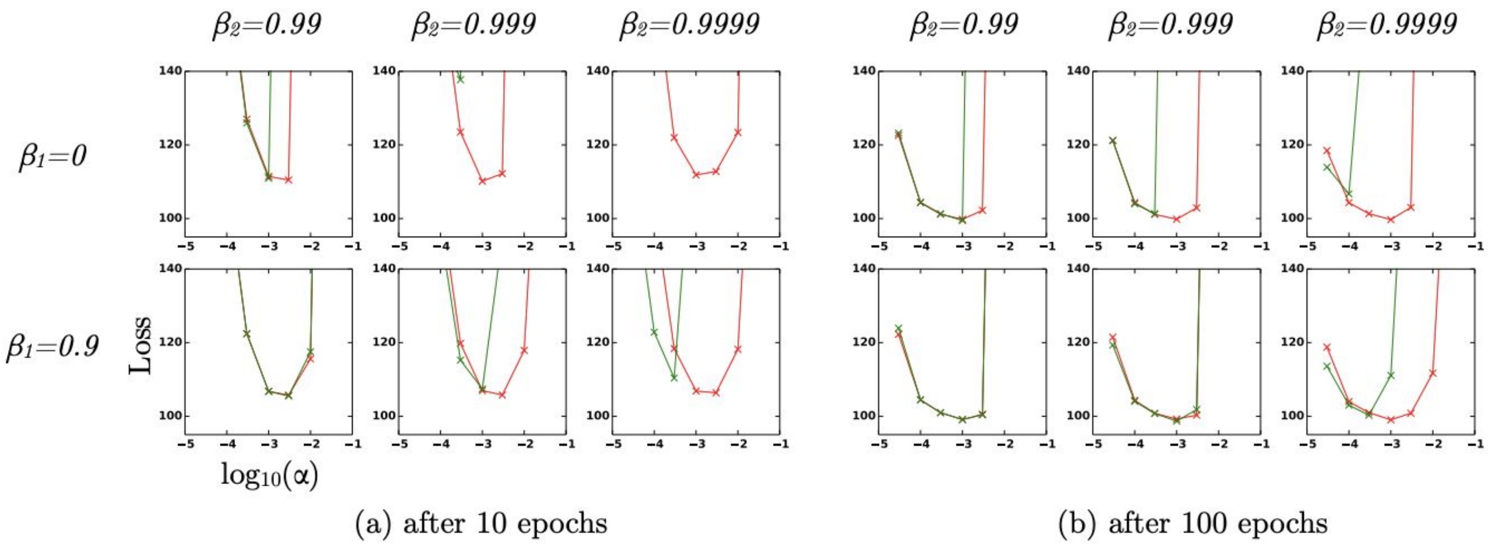


Figure : Effect of bias-correction terms (red line) versus no bias correction terms (green line) after 10 epochs (left) and 100 epochs (right) on the loss (y-axes) when learning a Variational AutoEncoder (VAE) (Kingma & Welling, 2013), for different settings of stepsize α (x-axes) and hyperparameters β_1 and β_2.

ADAMAX

- Adamax is a variant of Adam optimization algorithm that uses L-infinity norm based updates.
- In Adamax, the update rule for individual weights involves scaling their gradients inversely proportional to the L-infinity norm of their current and past gradients.
- The exponential weighted infinity norm is updated using a simple recursive formula:

$$u_t = \max(\beta_2 \cdot u_{t-1}, |g_t|)$$

- Unlike standard Adam, there is no need to correct for initialization bias in Adamax.
- This helps to prevent the Adam optimizer from over-fitting and improving generalization.
- It is a popular choice for optimization due to its fast convergence and robustness to noisy gradients.
- Adamax is well suited for sparse data and high dimensional parameters.

TEMPORAL AVERAGING

- Averaging the last iterate can improve the generalization performance in stochastic approximation as it is noisy.
- Polyak–Ruppert averaging and exponential moving average can be used for averaging the parameters.
- The exponential moving average can be easily implemented by adding this line to the inner loop of algorithms.

$$\bar{\theta}_t \leftarrow \beta_2 \cdot \bar{\theta}_{t-1} + (1 - \beta_2)\theta_t$$

- Initialization bias can be corrected by the estimator

$$\widehat{\theta}_t = \bar{\theta}_t / (1 - \beta_2^t)$$

DEMO

Conclusion -ADAM

1

Introduction of a simple and efficient algorithm for gradient-based optimization.

2

Aimed towards machine learning problems with large datasets and/or high-dimensional parameter spaces.

3

Combination of advantages of AdaGrad and RMSProp.

4

Easy implementation and low memory requirements.

5

Confirmation of rate of convergence in convex problems.

6

Robustness and suitability for non-convex optimization problems in machine learning.

Thank you!

You have been a great audience!

Let’s rock this semester together!!