ADADELTA: AN ADAPTIVE LEARNING RATE METHOD

Zeiler et al., 2012

Manas Jain

Idea

The idea presented in this paper was derived from ADAGRAD in order to improve upon the two main drawbacks of the method:

1) the continual decay of learning rates throughout training, and

2) the need for a manually selected global learning rate.

In the ADAGRAD method the denominator accumulates the squared gradients from each iteration starting at the beginning of training. Since each term is positive, this accumulated sum continues to grow throughout training, effectively shrinking the learning rate on each dimension. After many iterations, this learning rate will become infinitesimally small.

Adadelta - Introduction

Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w.

Instead of inefficiently storing w previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients.

The authors note that the units in this update (as well as in SGD, Momentum, or Adagrad) do not match, i.e. the update should have the same hypothetical units as the parameter. To realize this, they first define another exponentially decaying average, this time not of squared gradients but of squared parameter updates

Adadelta

- Adadelta [Zeiler, 2012] restricts the window of accumulated past gradients to a **fixed size**. SGD update:

$$\begin{aligned} \Delta\theta_t &= -\eta \cdot g_t \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \tag{4}$$

- Defines **running average** of squared gradients $E[g^2]_t$ at time t:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{5}$$

  - γ: fraction similarly to momentum term, around 0.9
- Adagrad update:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{6}$$

- Preliminary Adadelta update:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{7}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{8}$$

- Denominator is just root mean squared (RMS) error of gradient:

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} g_t \tag{9}$$

- Note: **Hypothetical units do not match**.
- Define **running average of squared parameter updates** and RMS:

$$\begin{aligned} E[\Delta\theta^2]_t &= \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2 \\ RMS[\Delta\theta]_t &= \sqrt{E[\Delta\theta^2]_t + \epsilon} \end{aligned} \tag{10}$$

- Approximate with $RMS[\Delta\theta]_{t-1}$, replace η for **final Adadelta update**:

$$\begin{aligned} \Delta\theta_t &= -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \tag{11}$$

Algorithm

Algorithm 1 Computing ADADELTA update at time t

Require: Decay rate ρ, Constant ϵ

Require: Initial parameter x_1

1: Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta x^2]_0 = 0$

2: **for** $t = 1 : T$ **do** %% Loop over # of updates

3: Compute Gradient: g_t

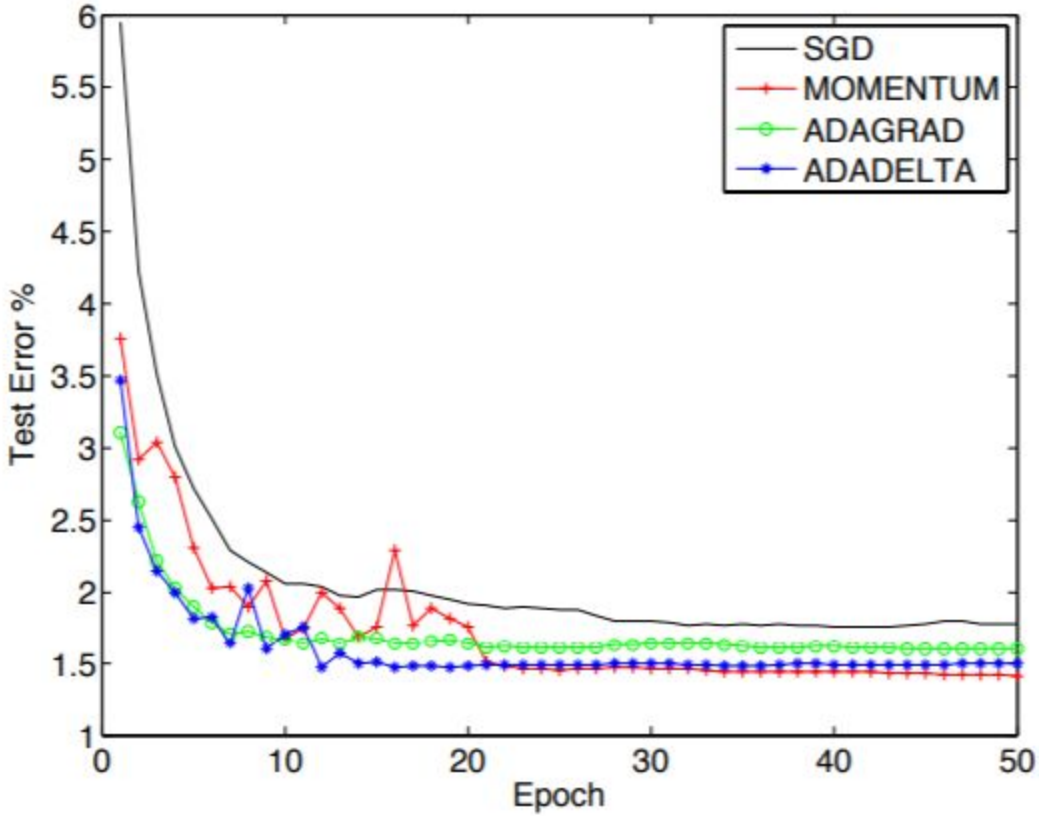
4: Accumulate Gradient: $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$

5: Compute Update: $\Delta x_t = -\frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t$

6: Accumulate Updates: $E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)\Delta x_t^2$

7: Apply Update: $x_{t+1} = x_t + \Delta x_t$

8: **end for**


SGD
MOMENTUM
ADAGRAD
ADADELTA
Test Error %
Epoch
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
10
20
30
40
50

	SGD	MOMENTUM	ADAGRAD
$\epsilon = 1e^{0}$	**2.26%**	89.68%	43.76%
$\epsilon = 1e^{-1}$	2.51%	**2.03%**	2.82%
$\epsilon = 1e^{-2}$	7.02%	2.68%	**1.79%**
$\epsilon = 1e^{-3}$	17.01%	6.98%	5.21%
$\epsilon = 1e^{-4}$	58.10%	16.98%	12.59%

Table 1. MNIST test error rates after 6 epochs of training for various hyperparameter settings using SGD, MOMENTUM, and ADAGRAD.

	$\rho = 0.9$	$\rho = 0.95$	$\rho = 0.99$
$\epsilon = 1e^{-2}$	2.59%	2.58%	2.32%
$\epsilon = 1e^{-4}$	2.05%	1.99%	2.28%
$\epsilon = 1e^{-6}$	1.90%	**1.83%**	2.05%
$\epsilon = 1e^{-8}$	2.29%	2.13%	2.00%

Table 2. MNIST test error rate after 6 epochs for various hyperparameter settings using ADADELTA.

ADADELTA - advantages

The benefits of ADADELTA are as follows:

- no manual setting of a learning rate.
- insensitive to hyperparameters.
- separate dynamic learning rate per-dimension.
- minimal computation over gradient descent.
- robust to large gradients, noise and architecture choice.
- applicable in both local or distributed environments.

Critical points

- *While AdaDelta is assumed to be parameter-free, there are some hyperparameters that need to be carefully tuned in order to exploit the most out of the algorithm. These parameters include ρ in running averages and the ε constant. However, mispecification of such hyperparameters is less sensitive to erroneous performance, compared to mispecificition of the learning rates.*
- *Having no explicit annealing schedule imposed on the learning rate could be why momentum with the proper hyperparameters outperforms ADADELTA later in training as seen in above figure. With momentum, oscillations that can occur near a minima are smoothed out, whereas with ADADELTA these can accumulate in the numerator. An annealing schedule could possibly be added to the ADADELTA method to counteract this in future work.*

Conclusion

With Adadelta, we do not even need to set a default learning rate, as it has been eliminated from the update rule. ADADELTA is a new learning rate method based on only first order information which shows promising result on MNIST and a large scale Speech recognition dataset.