Adadelta: An Adaptive Learning Rate Method

Mathew D. Zeiler
Google Inc., USA
New York University, USA

presented by Yao-Chi Hsu
2016/05/24
Outline

• Introduction
• Related Work
  – Learning Rate Annealing
  – Per-dimension First Order Methods
  – Methods Using Second Order Information
• Adadelta Method
  – Idea1: Accumulate Over Window
  – Idea2: Correct Units with Hessian Approximation
• Experiments
• Conclusion
• The aim of many machine learning methods is to update a set of parameters $x$ in order to optimize an objective function $f(x)$.

$$x_{t+1} = x_t + \Delta x_t$$

$$\Delta x_t = -\eta g_t$$

• Where $g_t$ is the gradient of the parameters at the $t$-th iteration $\frac{\partial f(x_t)}{\partial x_t}$ and $\eta$ is a learning rate which controls how large of a step to take in the direction of the negative gradient.

• stochastic gradient descent (SGD) +

• Choosing **higher than this rate** can cause the system to **diverge** in terms of the objective function, and choosing this **rate too low** results in **slow learning**.

• Determining a good learning rate becomes more of an art than science for many problems.

• This work attempts to alleviate the task of choosing a learning rate by introducing a new **dynamic learning rate** that is computed on a per-dimension basis using **only first order information**.
Introduction (3/3)

The benefits of this approach are as follows:

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

分散式環境
Related Work

- There are many modifications to the gradient descent algorithm.

- The most powerful such modification is **Newton’s method** which requires second order derivatives of the cost function:

\[
\Delta x_t = H_t^{-1} g_t
\]

- This determines the optimal step size to take for quadratic problems, but unfortunately is prohibitive to compute in practice for large models.

- Therefore, many additional approaches have been proposed to either improve the use of first order information or to approximate the second order information.
There have been several attempts to use heuristics for estimating a good learning rate at each iteration of gradient descent.

When gradient descent nears a minima in the cost surface, the parameter values can oscillate back and forth around the minima.

One method to prevent this is to slow down the parameter updates by decreasing the learning rate.

- validation accuracy
- learning rate decays
• The heuristic annealing procedure discussed above modifies a single global learning rate that applies to all dimensions of the parameters.

• Since each dimension of the parameter vector can relate to the overall cost in completely different ways, a per-dimension learning rate that can compensate for these differences is often advantageous.
A. *Momentum*

- This is perhaps the simplest extension to SGD that has been successfully used for decades.
- This is done by keeping track of past parameter updates with an exponential decay:

\[ \Delta x_t = \rho \Delta x_{t-1} - \eta g_t \]

- The gradients along the valley, despite being much smaller than the gradients across the valley, are typically in the same direction and thus the momentum term accumulates to speed up progress.
B. **ADAGRAD**

- This method relies on only first order information but has some properties of second order methods and annealing.

\[
\Delta x_t = - \frac{\eta}{\sqrt{\sum_j^t g_j^2}} g_t
\]

- Here the denominator computes the $\ell_2$ norm of all previous gradients on a per-dimension basis and $\eta$ is a global learning rate shared by all dimensions.

- Since this dynamic rate grows with the inverse of the gradient magnitudes, large gradients have smaller learning rates and small gradients have large learning rates.
B. **ADAGRAD**

\[
\Delta x_t = -\frac{\eta}{\sqrt{\sum_{i=1}^{t} g_i^2}} g_t
\]

- This is very beneficial for training deep neural networks since the scale of the gradients in each layer is often different by several orders of magnitude, so the optimal learning rate should take that into account.

- DNN訓練時error propagation隨著層數遞減(因為sigmoid)，使得層數較接近輸出層的更新量都較大，但使用ADAGRAD來更新的話則會使得過去更新量較大的參數得到較少的學習率。

- This method can be sensitive to initial conditions of the parameters and the corresponding gradients. (If the initial gradients are large, the learning rates will be low for the remainder of training.)
• While this provides additional curvature information useful for optimization, computing accurate second order information is often expensive.

• Becker and LeCun \(^+\) proposed a diagonal approximation to the Hessian.

\[
\Delta x_t = -\frac{1}{|\text{diag}(H_t)| + \mu} g_t
\]

• where the absolute value of this diagonal Hessian is used to ensure the negative gradient direction is always followed.

A recent method by Schaul et al. incorporating the diagonal Hessian with ADAGRAD-like terms has been introduced to alleviate the need for hand specified learning rates.

\[
\Delta x_t = -\frac{1}{|\text{diag}(H_t)|} \frac{E[g_{t-w:t}]^2}{E[g_{t-w:t}^2]} g_t
\]

Where \(E[g_{t-w:t}]\) is the expected value of the previous \(w\) gradients and \(E[g_{t-w:t}^2]\) is the expected value of squared gradients over the same window \(w\).
Adadelta Method (1/5)

- Idea 1: Accumulate Over Window

- Our methods implement this accumulation as an exponentially decaying average of the squared gradients.
- Assume at time $t$ this running average is then we compute:

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

- where $\rho$ is a decay constant similar to that used in the momentum method.

$$\text{RMS}[g]_t = \sqrt{E[g^2]_t + \epsilon} \quad \Delta x_t = -\frac{\eta}{\text{RMS}[g]_t} g_t$$

- where a constant $\epsilon$ is added to better condition the denominator.
Adadelta Method (2/5)

- Idea 2: Correct Units with Hessian Approximation

- When considering the parameter updates, $\Delta x$, being applied to $x$, the units should match.

- If the parameter had some hypothetical units, the changes to the parameter should be changes in those units as well.

- The units in SGD and Momentum relate to the gradient, not the parameter:

$$
\text{unit of } \Delta x \propto \text{unit of } g \propto \frac{\partial f}{\partial x} \propto \frac{1}{\text{unit of } x}
$$
Adadelta Method (3/5)

➢ Idea 2: Correct Units with Hessian Approximation

• In contrast, second order methods such as Newton’s method that use Hessian information or an approximation to the Hessian do have the correct units for the parameter updates:

\[ \Delta x \propto H^{-1} g \propto \frac{\partial f}{\partial x} \frac{\partial^2 f}{\partial x^2} \propto \text{unit of } x \]

• We rearrange Newton’s method (assuming a diagonal Hessian) for the inverse of the second derivative to determine the quantities involved:

\[ \Delta x = \frac{1}{\frac{\partial^2 f}{\partial x^2}} g = \frac{\partial f}{\partial x} \frac{1}{\frac{\partial^2 f}{\partial x^2}} \]

\[ \frac{1}{\frac{\partial^2 f}{\partial x^2}} \frac{\partial^2 f}{\partial x^2} \frac{\partial f}{\partial x} = \frac{\partial f}{\partial x} \]

\[ \Delta x \]

= \frac{\partial f}{\partial x} \]
Adadelta Method (4/5)

- Idea 2: Correct Units with Hessian Approximation

\[
\Delta x = \frac{1}{\frac{\partial^2 f}{\partial x^2}} g = \frac{\Delta x}{\frac{\partial f}{\partial x} g} \quad \Delta x_t = \frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t
\]

**Algorithm 1** Computing ADADELTA update at time \( t \)

**Require:** Decay rate \( \rho \), Constant \( \epsilon \)

**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
Adadelta Method (5/5)

- Idea 2: Correct Units with Hessian Approximation
  
  \[
  \Delta x_t = \frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t
  \]

1) 在分子的部分有類似Momentum的效果

2) 分母則保留Adagrad的特性(每個維度給予不同的學習率)

3) 且只利用first order information就達到Hessian matrix的效果(近似)
Experiments (1/4)

- Handwritten Digit Classification

Fig. 1. Comparison of learning rate methods on MNIST digit classification for 50 epochs.
Experiments (2/4)

- Sensitivity to Hyperparameters

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>SGD</th>
<th>MOMENTUM</th>
<th>ADAGRAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^0$</td>
<td>2.26%</td>
<td>89.68%</td>
<td>43.76%</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>2.51%</td>
<td>2.03%</td>
<td>2.82%</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>7.02%</td>
<td>2.68%</td>
<td>1.79%</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>17.01%</td>
<td>6.98%</td>
<td>5.21%</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>58.10%</td>
<td>16.98%</td>
<td>12.59%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$\rho = 0.9$</th>
<th>$\rho = 0.95$</th>
<th>$\rho = 0.99$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>2.59%</td>
<td>2.58%</td>
<td>2.32%</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>2.05%</td>
<td>1.99%</td>
<td>2.28%</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>1.90%</td>
<td>1.83%</td>
<td>2.05%</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>2.29%</td>
<td>2.13%</td>
<td>2.00%</td>
</tr>
</tbody>
</table>

Table 2. MNIST test error rate after 6 epochs for various hyperparameter settings using ADADELTA.
Experiments (3/4)

- Effective Learning Rates

![Graphs showing effective learning rates and vanishing gradients](image-url)
Experiments (4/4)

- Speech Data

**Fig. 3.** Comparison of ADAGRAD and ADADELTA on the Speech Dataset with 100 replicas using logistic nonlinearities.

**Fig. 4.** Comparison of ADAGRAD, Momentum, and ADADELTA on the Speech Dataset with 200 replicas using rectified linear nonlinearities.
Conclusion

• In this tech report we introduced a new learning rate method based on only first order information which shows promising result on MNIST and a large scale Speech recognition dataset.

• This method has trivial computational overhead compared to SGD while providing a per-dimension learning rate.

• Despite the wide variation of input data types, number of hidden units, nonlinearities and number of distributed replicas, the hyperparameters did not need to be tuned, showing that ADADELTA is a robust learning rate method that can be applied in a variety of situations.
THANK YOU