CS7150.37338.202130 Day 8 Paper Questions

Aishwarya Vantipuli, Harshita Ved, Aveek Choudhury

TOTAL POINTS

5 / 5

QUESTION 1

1 Question 1 **1 / 1**

✓ **- 0 pts** Correct

QUESTION 2

2 Question 2 **1 / 1**

✓ **- 0 pts** Correct

QUESTION 3

3 Question 3 **1 / 1**

✓ **- 0 pts** Correct

QUESTION 4

4 Question 5 **1 / 1**

✓ **- 0 pts** Correct

QUESTION 5

5 Question 6 **1 / 1**

✓ **- 0 pts** Correct

CS 7150: Deep Learning — Spring 2021 — Paul Hand

Day 8 — Preparation Questions For Class
Due: Wednesday 2/17/2021 at 2:30pm via Gradescope

Names: [Aishwarya Vantipuli, Aveek Choudhury, Harshita Ved]

You may consult any and all resources in answering the questions. Your goal is to have answers that are ready to be shared with the class (or on a hypothetical job interview) as written. Your answers should be as concise as possible. When asked to explain a figure, your response should have the following structure: provide context (state what experiment was being run / state what problem is being solved), state what has been plotted, remark on what we observe from the plots, and interpret the results.

Submit one document for your group and tag all group members. We recommend you use Overleaf for joint editing of this TeX document.

Directions: Read 'Deep Learning Book - Chapter 8' by Goodfellow et al.

- Read Section 8.3.2, 8.5.1, 8.5.2, 8.5.3, 8.5.4.

Question 1. *What is the idea behind momentum in first order optimization?*

Response:
Momentum is a popular technique used with Stochastic Gradient Descent (SGD) to accelerate learning and reduce oscillation. The problem with standard SGD is that it tries to reach mnima because of high oscillations and thus gradients are very noisy. SGD with momentum overcomes this by denoising gradients using exponential weighting average.

Instead of using only the gradient of the current step to guide the search, momentum also accumulates the gradient of the past steps to determine the direction to go. Momentum is computed by introducing velocity v using all previous updates giving more weightage to recent updates compared to the previous update. This lead to speed up the convergence.

$$w_t = w_{t-1} - \eta V_{dw_t}$$

where

$$V_{dw_t} = \beta V_{dw_{t-1}} + (1-\beta)\frac{\partial L}{\partial w_{t-1}}$$

1 Question 1 **1 / 1**

✓ **- 0 pts** Correct

Question 2. *What is the idea behind the AdaGrad algorithm?*

Response:

We need different learning rates because the learning rate for sparse features parameters needs to be higher compare to the dense features parameter because the frequency of occurrence of sparse features is lower. The idea behind Adaptive Gradient Algorithm (Adagrad) is to use different learning rates for each parameter based on iterations. It individually adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of the gradient.

$$w_t = w_{t-1} - \eta_t' \frac{\partial L}{\partial w_{t-1}}$$

where

$$\eta_t' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$$

$$\alpha_t \sum_{i=1}^{t} (\frac{\partial L}{\partial w_{t-1}})^2$$

One huge disadvantage is due to monotonically decreasing learning rates, at some point in time step, the model will stop learning as the learning rate is almost close to 0.

Question 3. *What is the idea behind the RMSProp algorithm?*

Response:

RMSprop, or Root Mean Square Propogation was developed in order to overcome the short comings of the AdaGrad algorithm. That is, RMSProp does not decay the learning rate too quickly preventing convergence. It uses an exponentially decaying average to discard history from the extreme past so that it can converge rapidly and will decrease the size of the gradient steps towards minima when the steps are too large.The central idea of RMSprop is keep the moving average of the squared gradients for each weight and divide the gradient by square root the mean square (which is why the name relates to RMS).

$$w_t = w_{t-1} = -\frac{\eta}{E[g^2]_t} \frac{\partial L}{\partial w}$$

where

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\frac{\partial L}{\partial w})^2$$

E[g] — moving average of squared gradients. dL/dw — gradient of the cost function with respect to the weight. n — learning rate. Beta — moving average parameter

2 Question 2 **1 / 1**

✓ **- 0 pts** Correct

Question 2. *What is the idea behind the AdaGrad algorithm?*

Response:

We need different learning rates because the learning rate for sparse features parameters needs to be higher compare to the dense features parameter because the frequency of occurrence of sparse features is lower. The idea behind Adaptive Gradient Algorithm (Adagrad) is to use different learning rates for each parameter based on iterations. It individually adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of the gradient.

$$w_t = w_{t-1} - \eta_t' \frac{\partial L}{\partial w_{t-1}}$$

where

$$\eta_t' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$$

$$\alpha_t \sum_{i=1}^{t} (\frac{\partial L}{\partial w_{t-1}})^2$$

One huge disadvantage is due to monotonically decreasing learning rates, at some point in time step, the model will stop learning as the learning rate is almost close to 0.

Question 3. *What is the idea behind the RMSProp algorithm?*

Response:

RMSprop, or Root Mean Square Propogation was developed in order to overcome the short comings of the AdaGrad algorithm. That is, RMSProp does not decay the learning rate too quickly preventing convergence. It uses an exponentially decaying average to discard history from the extreme past so that it can converge rapidly and will decrease the size of the gradient steps towards minima when the steps are too large.The central idea of RMSprop is keep the moving average of the squared gradients for each weight and divide the gradient by square root the mean square (which is why the name relates to RMS).

$$w_t = w_{t-1} = -\frac{\eta}{E[g^2]_t} \frac{\partial L}{\partial w}$$

where

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\frac{\partial L}{\partial w})^2$$

E[g] — moving average of squared gradients. dL/dw — gradient of the cost function with respect to the weight. n — learning rate. Beta — moving average parameter

3 Question 3 **1 / 1**

✓ **- 0 pts** Correct

Directions: Read 'Adam: A Method for Stochastic Optimization' by Kingma and Ba.

- Read Section 1, 2, 5, 6.3

Question 5. *The authors say that the benefits of Adam are: that the magnitudes of parameter updates are invariant to rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing. Briefly interpret/explain each of these points.*

Response:

Stepsizes are approximately bounded by stepsize hyperparameter:

In Adam's update rule, assuming $\epsilon = 0$, the effective step taken in parameter space at timestep t is $\Delta_t = \alpha.\hat{m}_t/\hat{v}_t$. In more common cases,i.e. less sparse, $\hat{m}_t/\sqrt{\hat{v}_t} \approx \pm 1$ since $| E[g]/\sqrt{E[g^2]} | \leq 1$. When $(1-\beta_1) = \sqrt{1-\beta_2}$ (the exponential decay rates for the moment estimates), $| \hat{m}_t/\sqrt{\hat{v}_t} | < 1$, therefore - $| \Delta_t | < \alpha$. For the case when $(1-\beta_1) > \sqrt{1-\beta_2}$, i.e. severe case of sparsity, $| \Delta_t | \leq \alpha.(1-\beta_1)/\sqrt{1-\beta_2}$. Thus, stepsizes are approximately bounded by stepsize hyperparameter.

Does not require stationary objective:

Similar to RMSProp, Adam maintains per-parameter learning rates that are adapted based on exponential average of recent magnitudes of the gradients for the weights, i.e. tracking how quickly they change. This means that the algorithm converges even if the objective changes with time and thus, Adam works with non-stationary (noisy) objective.

It works with sparse gradients:

For Adam optimization method, the step sizes are controlled even in cases of extreme sparsity and hence, are not vanishing. It is important to note that in order to work with sparse cases, there is a need to remember many past values. This can be done by setting default values of β_2 to high, thereby having a larger window of averaging.

For sparse gradients, $(1-\beta_1) > \sqrt{1-\beta_2}$, the effective step size is upper bounded by - $| \Delta_t | \leq \alpha.(1-\beta_1)/\sqrt{1-\beta_2}$.

Naturally performs a form of step size annealing:

The authors loosely define signal-to-noise ratio (SNR) as the ratio $\hat{m}_t/\sqrt{\hat{v}_t}$. The property of effective stepsize Δ_t being closer to 0 with a smaller SNR is desirable since a smaller SNR means there is greater uncertainty whether the direction of $\hat{m}_t$ corresponds to the direction of the true gradient. This is substantiated by the example that SNR value typically becomes closer to 0 towards an optimum, leading to a smaller effective steps in the parameter space - naturally performing a form of step size annealing.

Question 6. *Explain Figure 3.*

Response:

Context:

A convolution neural network architecture was trained on the CIFAR-10 dataset for image classification. The architecture has 3 alternating stages of 5x5 convolution filters and 3x3 max

4 Question 5 **1 / 1**

✓ **- 0 pts** Correct

Directions: Read 'Adam: A Method for Stochastic Optimization' by Kingma and Ba.

- Read Section 1, 2, 5, 6.3

Question 5. *The authors say that the benefits of Adam are: that the magnitudes of parameter updates are invariant to rescaling of the gradient, its stepsizes are approximately bounded by the stepsize hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing. Briefly interpret/explain each of these points.*

Response:

Stepsizes are approximately bounded by stepsize hyperparameter:

In Adam's update rule, assuming $\epsilon = 0$, the effective step taken in parameter space at timestep t is $\Delta_t = \alpha . \hat{m}_t / \hat{v}_t$. In more common cases,i.e. less sparse, $\hat{m}_t/\sqrt{\hat{v}_t} \approx \pm 1$ since $\mid E[g]/\sqrt{E[g^2]} \mid \leq 1$. When $(1-\beta_1) = \sqrt{1-\beta_2}$ (the exponential decay rates for the moment estimates), $\mid \hat{m}_t/\sqrt{\hat{v}_t} \mid < 1$, therefore - $\mid \Delta_t \mid < \alpha$. For the case when $(1-\beta_1) > \sqrt{1-\beta_2}$, i.e. severe case of sparsity, $\mid \Delta_t \mid \leq \alpha . (1-\beta_1)/\sqrt{1-\beta_2}$. Thus, stepsizes are approximately bounded by stepsize hyperparameter.

Does not require stationary objective:

Similar to RMSProp, Adam maintains per-parameter learning rates that are adapted based on exponential average of recent magnitudes of the gradients for the weights, i.e. tracking how quickly they change. This means that the algorithm converges even if the objective changes with time and thus, Adam works with non-stationary (noisy) objective.

It works with sparse gradients:

For Adam optimization method, the step sizes are controlled even in cases of extreme sparsity and hence, are not vanishing. It is important to note that in order to work with sparse cases, there is a need to remember many past values. This can be done by setting default values of β_2 to high, thereby having a larger window of averaging.

For sparse gradients, $(1-\beta_1) > \sqrt{1-\beta_2}$, the effective step size is upper bounded by - $\mid \Delta_t \mid \leq \alpha . (1-\beta_1)/\sqrt{1-\beta_2}$.

Naturally performs a form of step size annealing:

The authors loosely define signal-to-noise ratio (SNR) as the ratio $\hat{m}_t/\sqrt{\hat{v}_t}$. The property of effective stepsize Δ_t being closer to 0 with a smaller SNR is desirable since a smaller SNR means there is greater uncertainty whether the direction of $\hat{m}_t$ corresponds to the direction of the true gradient. This is substantiated by the example that SNR value typically becomes closer to 0 towards an optimum, leading to a smaller effective steps in the parameter space - naturally performing a form of step size annealing.

Question 6. *Explain Figure 3.*

Response:

Context:

A convolution neural network architecture was trained on the CIFAR-10 dataset for image classification. The architecture has 3 alternating stages of 5x5 convolution filters and 3x3 max pooling with stride of 3, followed by a fully connected layer of 1000 rectified linear hidden units (ReLU's). The input images were pre-processed by whitening and dropout noise is applied to the input and fully connected layer.

What is plotted:
Both plots show the training cost as a function of iterations over the entire dataset for networks using various optimization techniques like Adam, AdaGrad and SGD. The left plot shows the trend over the first 3 epochs, whereas the right plot shows the same trend over 45 epochs.

What we observe:
From the left plot, we observe that during the initial phase of training, both Adam (with without dropout) and AdaGrad quickly lower the training cost. However, as the epochs progress, Adam SGD converge considerably faster than others, and also lower cost significantly more than the other networks. In fact, AdaGrad variants show the least reduction in cost over iterations.

Interpretation:
Adam optimization technique is able to converge significantly faster than other stochastic optimization techniques and lowers training cost.

5 Question 6 **1 / 1**

✓ **- 0 pts** Correct