Quiz 2

Karan Vora (username: kv2154)

Attempt 1

Written: Mar 9, 2023 5:32 PM - Mar 9, 2023 5:51 PM

Submission View

Released: Apr 2, 2023 10:11 AM

Question 1 **2 / 2 points**

Consider the Batch Normalizing Transform, applied to activation x over a mini-batch.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- ✓ Batch norm can be effective with any batch size, including 1.
- ✓ Batch norm can enable faster convergence as it allows working with larger learning rates.
- ✓ The trainable parameters are gamma (scale) and beta (shift).
- ✓ Batch norm can be applied with any activation function.
- ✓ It may be possible that learned values of gamma and beta give back the original activation value (before applying batch norm transformation), i.e., y_i will be the same as x_i
- ✓ Batch norm can reduce internal covariance shift.

Question 2 **1.2 / 2 points**

Let A(z) denote an activation function with input z and A'(z) be its derivative. Write the name of the appropriate activation function(s) for which each of the following statements is true for all values of z. For each statement below your response can be one of the activations:

Linear, Sigmoid, Tanh, ReLU, Leaky ReLU, Hard Tanh.

When filling in the name of the activation function please start the name in capital and write it in exactly the same format as mentioned above.

(1). A(-z) = 1-A(z) _____

(2). A(-z) = -A(z) _____

(3). A'(z) = A(z)(1-A(z)) _____

(4). _____ (z) = 2 (5). _____ (2z) - 1

Note: In blanks 4 and 5 you are required to provide two different activation functions who are related to each other by the given equation.

Answer for blank # 1:	Sigmoid	✓(20 %)
Answer for blank # 2:	Tanh	✓(20 %)
Answer for blank # 3:	Sigmoid	✓(20 %)
Answer for blank # 4:	Hard Tanh and ReLU	✗ **(Tanh, tanh)**
Answer for blank # 5:	Sigmoid and Tanh	✗ **(Sigmoid, sigmoid)**

Question 3 **0.8 / 1 point**

Mark all that is/are true about activation functions.

- ✓ While ReLU can also prevent dead neurons it is not preferred over Leaky ReLU as ReLU can often cause vanishing gradient problems.
- ➡ ✓ The derivative of sigmoid with z as input is maximum at z=0 while its value is maximum at z=0.25.
- ➡ ✗ Derivative of Tanh is maximum at z=0 and its value is 1.
- ✓ Leaky ReLU always prevents dead neurons while training.
- ✓ Tanh, sigmoid, and ReLU all squash the input to a value in the positive quadrant.

Question 4 **1 / 1 point**

One of the reason AdaDelta was developed was to improve AdaGrad weakness of learning rate converging to zero with increase of time.

- ✓ ◯ True
- ◯ False

Question 5 **1.667 / 2 points**

Consider an example of training using Nesterov momentum. Here alpha(t) is the learning rate at iteration t and beta is the momentum parameter.

$$V_t = \beta\, V_{t-1} - \alpha(t)\frac{\partial L(W_{t-1}+\beta V_{t-1})}{\partial W_{t-1}}$$
$$W_t = W_{t-1} + V_t$$

$$\alpha(t) = \begin{cases} 0.1 & t < T \\ 0 & t \geq T \end{cases}$$

For this example identify the correct weight updates and momentum updates from below. Your selection should be in terms of A, B, ...

A. $W_{T+1} = W_{T-1} + \beta V_T$
B. $W_{T+1} = W_{T-1} + (1+\beta) V_T$
C. $W_T = W_{T-1} + \beta V_T$
D. $W_T = W_{T-1} + \beta(1+\beta) V_{T-1}$
E. $V_T = \beta V_{T-1}$
F. $V_{T+1} = \beta V_T - \alpha(T+1) \frac{\partial L(W_T + \beta V_T)}{\partial W_T}$

- ✗ ☐ C
- ➡ ✓ ☐ F
- ✓ ☐ A
- ✓ ☐ D
- ➡ ✓ ☐ E
- ➡ ✓ ☐ B

Question 6 **0 / 1 point**

Which of the following is false regarding torch.multiprocessing?

- ○ The multiprocessing module of PyTorch is more efficient than standard Python multiprocessing.
- ✗ ○ We can further improve the performance by using multiprocessing.Queue which utilizes multiple threads to serialize and send objects.
- ➡ ○ In Pytorch multiprocessing, we need to make a deep copy of tensors before sending.
- ○ In Pytorch multiprocessing, we copy and serialize tensors in a shared memory area and only pass handles.

Question 7 **1 / 1 point**

Which of the following is/are true?

- ✓ ☐ Linear regression output can not assume all values however Logistic regression can.
- ✓ ☐ Linear regression finds the best-fitting straight line which is known as a regression line.
- ✓ ☐ Logistic Regression is used to model the probability of a binary event.

Question 8 **0.333 / 1 point**

Which of the following mentions the correct meaning of the hyperparameters is dataloader method?

- ✗ ☐ Shuffle: A bool which says if we want to reshuffle data at each batch or not
- ✗ ☐ Batch Size: The size of the entire dataset
- ➡ ✓ ☐ Sampler: Show the sample you want to use e.g., SequentialSampler or RandomSampler

Question 9 **1 / 1 point**

Shared memory is a memory area that the OS (eg Linux) maps on the address space of the processes, allowing in this way to be simultaneously accessed by multiple programs with an intent to provide communication among them or avoid redundant copies.

- ✓ ○ True
- ○ False

Question 10 **1.333 / 2 points**

Pinned and Persistent memory. Which of the following is true?

- ➡ ✓ ☐ To use enable Pinned memory we need to use pin_memory=True while creating the dataloader.
- ➡ ✗ ☐ Persistent memory is the device memory that will be in the context across multiple kernel launches.
- ✓ ☐ Pinned memory is the host memory that needs to be swapped back to disk.

Question 11 **0.5 / 1 point**

Which of the following is/are False?

- ➡ ✓ ☐ RELU is very susceptible to vanishing gradient problem.
- ✗ ☐ Activation functions and Weight initializers are hyperparameters in Deep Learning.
- ✗ ☐ Improper initialization of weights can lead to vanishing or exploding gradients.
- ➡ ✓ ☐ An activation function can never cause vanishing or exploding gradients.

Question 12 **1.5 / 2 points**

Which of the following is/are true for Normalization?

- ➡ ✓ ☐ Batch normalization enables training with larger learning rates and allows the model to generalize better and converge faster.

- ✓ The parameters of normalization are calculated using training data.
- ✗ While validating and testing we should not apply the same normalization to the test and valid set as we did in the training set so that our model learns better.
- ✓ Batch normalization reduces internal covariance shift which is the change in the distribution of network activations due to change in network parameters during training.

Question 13 **0.75 / 1 point**

Which of the following holds true for Learning Rate?

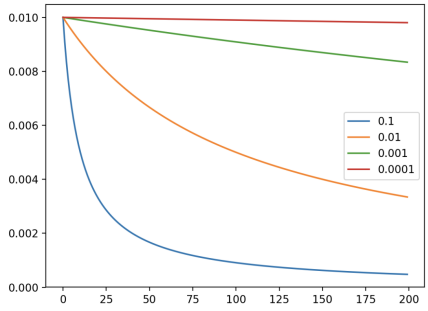
- ✓ None of the above
- ✓ A large learning rate may allow the algorithm to come close to a good solution but will then oscillate around the point or even diverge.
- ✓ A low learning rate may cause the algorithm to take a too long time to come even close to an optimal solution.
- ✗ The optimal solution for learning rates is to start with a higher value to find a good starting point for the weights and then move to a smaller value to converge to a minima.

Question 14 **1 / 2 points**

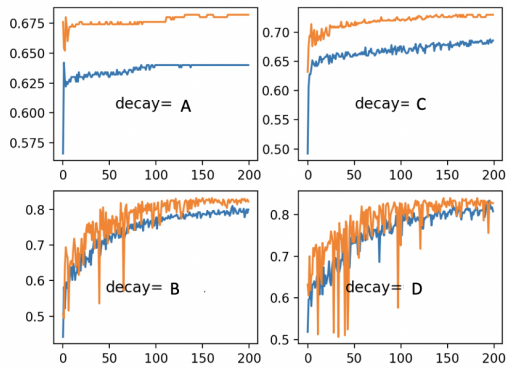
While learning rate usually decays using a learning rate decay schedule as training progresses, it needs to be done wisely as if the decay is too aggressive, it may prevent any learning. To understand this an experiment was done in which four different decay schedules we used. Look at the code below to decay the learning rate used in this experiment.

```
# demonstrate the effect of decay on the learning rate
from matplotlib import pyplot

# learning rate decay
def decay_lrate(initial_lrate, decay, iteration):
    return initial_lrate * (1.0 / (1.0 + decay * iteration))

decays = [1E-1, 1E-2, 1E-3, 1E-4]
lrate = 0.01
n_updates = 200
for decay in decays:
    # calculate learning rates for updates
    lrates = [decay_lrate(lrate, decay, i) for i in range(n_updates)]
    # plot result
    pyplot.plot(lrates, label=str(decay))
pyplot.legend()
pyplot.show()
```

Running the above code creates a line plot showing learning rates over updates for different decay values.



The following picture shows the training and test accuracy for the 4 training jobs, each with one of the four learning rate schedules. Here each schedule is identified by the decay value. Each figure is labeled with an alphabet for the decay value.



The four decay values are:

0.1, 0.01, 0.001, 0.0001
Write the decay value in this manner:
(1). A _____
(2). B _____
(3). C _____
(4). D _____

Answer for blank # 1: 0.0001 ✗ **(0.1)**
Answer for blank # 2: 0.001 ✓(25 %)
Answer for blank # 3: 0.01 ✓(25 %)

Answer for blank # 4: 0.1 ✗ (0.0001)

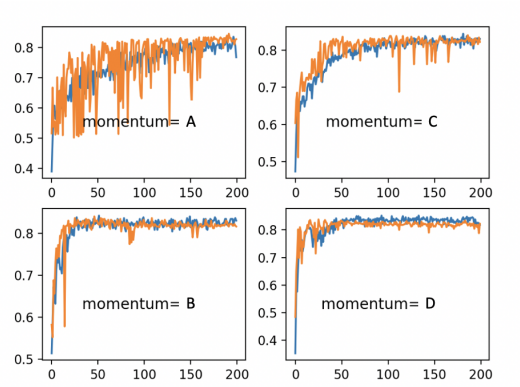
Question 15 **0 / 1 point**

Select which of the following is true for start methods for a new process.

- ✗ ☐ The fork method starts a new fresh process with minimal resources inherited.
- ✗ ☐ In the Spawn method, a new process is created which inherits all resources.
- ➡ ✗ ☐ Usually, Spawn and fork-Server methods are slower but they are a safer option.
- ➡ ✗ ☐ The fork-Server method creates a new server process that forks new child processes. This helps in keeping the original process safer.

Question 16 **0.5 / 1 point**

The following 4 figures show test and training set accuracy with different momentum values. All other parameters are same in the four experiments. The momentum value for each experiment is written as A, B, C, or D in the figure.



Each experiment corresponds to one of the following 4 values for momentum:

0, 0.5, 0.9, 0.99
Match each alphabet with the right value of momentum.

(1). A _____
(2). B _____
(3). C _____
(4). D _____

Answer for blank # 1: 0 ✓(25 %)
Answer for blank # 2: 0.5 ✗ (0.9)
Answer for blank # 3: 0.9 ✗ (0.5)
Answer for blank # 4: 0.99 ✓(25 %)

Question 17 **0.25 / 1 point**

Which of the following is true for Batch Size?

- ➡ ✗ ☐ We can estimate an optimal batch size that is proportional to the learning rate as long as batch size <<< training data.
- ✗ ☐ Large batch size leads to faster convergence, better generalization, and offers to utilize higher learning rates.
- ➡ ✓ ☐ Decreasing the learning rate has the same effect as increasing batch size.
- ➡ ✗ ☐ When you have hardware constraints you can work with a large batch size by delaying gradient/weight updates to happen every n iterations instead of updating every iteration.

Question 18 **0.8 / 1 point**

Which of the following is not true for the Computation Graph?

- ➡ ✓ ☐ Computation Graph is the same as a neural network.
- ➡ ✗ ☐ The computation graph represents the components of a function and can have cycles in it.
- ✓ ☐ In PyTorch, the computation graph is constructed at run-time whereas in Tensorflow it is built at compile-time.
- ✓ ☐ Computation Graph allows scope for optimization as we can parallelize the computations present in it.
- ✓ ☐ In PyTorch, Autograd builds the Computation Graph dynamically.

Question 19 **0.75 / 1 point**

Which of the following is true related to torch.cuda?

- ✗ ☐ There are methods where we can move objects to GPU.
- ✓ ☐ We cannot switch between other devices.
- ➡ ✓ ☐ Keeps track of existing GPUs.
- ➡ ✓ ☐ torch.cuda sets up a GPU device and runs operations on it.

Attempt Score: 16.383 / 25 - D
Overall Grade (highest attempt): 16.383 / 25 - D

Done