Discord's username: txdat
Google classroom: dattranx105@gmail.com

1 Question 1

a

- The formular of f with x, y, z is $f(x, y, z) = \delta((0.5x + 0.1y) * 3.0z)$ with δ is the sigmoid function.
- $f(x = -0.6, y = 0.5, z = -0.4) = \delta(0.3) = 0.5744425$

b

Set $S = (0.5x + 0.1y) * 3.0z$
We have $f(x) = \delta(x) = \frac{1}{1+e^{-x}} \to \frac{df}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = f(x) * (1 - f(x))$
- $\frac{df}{dx} = \frac{df}{dS} * \frac{dS}{dx} = f(S) * (1 - f(S)) * 1.5z = -0.146675$
- $\frac{df}{dy} = \frac{df}{dS} * \frac{dS}{dy} = f(S) * (1 - f(S)) * 0.3z = -0.029335$
- $\frac{df}{dz} = \frac{df}{dS} * \frac{dS}{dz} = f(S) * (1 - f(S)) * (0.5 * x + 0.1 * y) = -0.061114578$

2 Question 2

Set $y = 0$, $\hat{y} = f(x_1, x_2) = \delta(S) = \delta(x_1 * w_1 + x_2 * w_2)$ and $L = 0.5 * (\hat{y} - y)^2$ with δ is the sigmoid function.

a

With $w_1 = 0.5, w_2 = -0.5, x_1 = 5, x_2 = 10 \to \hat{y} = \delta(-2.5) = 0.075858$
- $\frac{dL}{dw_1} = \frac{dL}{d\hat{y}} * \frac{d\hat{y}}{dS} * \frac{dS}{dw_1} = (\hat{y} - y) * \hat{y} * (1 - \hat{y}) * x_1 = 0.0265897$
- $\frac{dL}{dw_2} = \frac{dL}{d\hat{y}} * \frac{d\hat{y}}{dS} * \frac{dS}{dw_2} = (\hat{y} - y) * \hat{y} * (1 - \hat{y}) * x_2 = 0.0531794$

b

With $\alpha = 0.4$
- $w_1' = w_1 - \alpha * \frac{dL}{dw_1} = 0.5 - 0.4 * 0.0265897 = 0.489364$
- $w_2' = w_2 - \alpha * \frac{dL}{dw_2} = -0.5 - 0.4 * 0.0531794 = -0.521272$

3 Question 3

a

The effect of learning rate over-optimizing the objective function?
When the learning rate is high, the value of the loss function oscillates around the (local or global) optimum value, but it may not reach it or fails to converge,

so the solution may be not optimal. In some cases, it may overfit with the training data and not be generalized.

b

The learning rate schedulers are methods to adjust the learning rate during the training process. An example is decreasing the learning rate during the training process after some iterations for better converge.

There are some useful learning rate schedulers like: *StepLR*, *ConstantLR*, *LinearLR*, *CosineAnnealingLR*, *OneCycleLR*, ...[1] [2]

4 Question 4

The effect of batch size over training and optimizing model performance?

There are three common types of batch processing in deep learning: batch (whole dataset), mini-batch (some randomized samples), and stochastic (one sample). With batch training, whole dataset is used in one iteration to compute gradient and update model's parameters. It can provide more accurate estimations and more generalized but requires more expensive computation cost. It may not useful with the large of data in training deep learning models. With stochastic training, it is the fastest, computation efficient method to estimate and update the model's parameters, but it is usually effected by the noise during training process. With mini-batch training, a small number of samples are sampled and used in one iteration, it balances the computation efficiency and accurancy.

5 Question 5

The working principles of the optimizer (Lion) from the paper 'Symbolic Discovery of Optimization Algorithms' [3]

- The paper presents a method to formulate algorithm discovery and apply it to optimization algorithms.

- It differs from other adaptive algorithms (like *AdamW*, ...) by only tracking the momentum and using the sign operation to calculate (uniform) updates.

- The authors define the search space that has main inputs are weight w, gradient g, and learning rate lr, and main output are the update to weight. They use 45 common math functions (including linear algebra) to build the sequence of statements and mutate them. They also propose efficient search techniques (evolution and restart, pruning, and using proxy tasks) to address the challenges from the search space.

The math of the Lion optimizer

[1] https://pytorch.org/docs/stable/optim.htmlhow-to-adjust-learning-rate
[2] https://www.kaggle.com/code/isbhargav/guide-to-pytorch-learning-rate-scheduling
[3] https://arxiv.org/pdf/2302.06675v4.pdf

- It only tracks the momentum without extra step *Bias correction* like *Adam*.

- It uses uniform update magnitudes across all dimensions $(\eta * sign(C))$ instead of fraction of bias-corrected mean and variance $(\frac{\hat{m_t}}{\sqrt{\hat{v_t}}+\epsilon})$ like *AdamW*. Therefore, Lion requires a smaller learning rate and larger weight decay to maintain the weight decay strength.

The advantages of using Lion over Adam and other optimizers.

- Through all Lion's evaluations, it is the best performing optimizer in most part of them, and is expected to train better models than other optimizers.

- Lion converges faster and is more memory-efficient than Adam and others because it tracks only momentum.