CSCI 5922 problem set 1

Jake Krol

January 2025

1 Q1

1.1 Q1a

The **fundamental difference between parameters and hyper-parameters** is that parameter values are learned and updated during algorithmic training while hyper-parameter values are, typically, manually set and fixed during training. Specifically, **hyper-parameters** are evaluated by measuring performance of a model on the **validation set**. Meanwhile, **parameters** are updated by algorithmic learning over the **training dataset**, such as backpropagation and stochastic gradient descent.

1.2 Q1b

Item	P or HP?
A weight matrix **w**	P
The learning rate	HP
A bias term **b**	P
The minibatch size	HP
The non-linear activation function	HP
The optimizer	HP

Justification of table answers.

- **A weight matrix** is a **parameter** since the values of the matrix are updated during algorithmic training and not set manually.
- **The learning rate** is a **hyper-parameter** since this value is typically fixed during training. Although, the learning rate can vary with a tech-

nique called "learning rate decay". However, the decay is a function of training time, and the learning rate parameter is not fit based on the training data.

- **A bias term** is a **parameter** since its value is updated during training and not set manually.
- **The minibatch size** is a **hyper-parameter** since this is a manually set value which determines how many samples in an epoch will be used to aggregate a gradient for backpropagation. The minibatch size is not updated during training, and it is therefore not a learned parameter.
- **The non-linear activation functions** of a neural network are **hyper-parameters** since these functions are manually selected and do not change during training.
- **The optimizer** is the algorithm which decides how a model will use gradients to update parameter values, and optimizers are defined manually and not updated during training, making the optimizer a **hyper-parameter**.

1.3 Q1c

- **The number of hidden layers (depth) is another hyper-parameter** of a neural network since the number is manually defined and does not change during training.
- **The number of units/neurons per hidden layer (width) is another hyper-parameter** of a neural network, too. Similar to model depth, the number of units per layer does not change due to training and is typically set manually.

2 Q2

2.1 Q2a

The model is **overfitting** after 5 epochs since validation accuracy begins to decrease after 5 epochs, despite training loss continously decreasing.

I would set the **epochs hyper-parameter to 5** to achieve the best generalization of the model with respect to validation set accuracy.

2.2 Q2b

The model is **underfitting** since train loss is continuously decreasing and validation accuracy is continuously increasing, yet too slowly to reach a minimum loss and maximum validation accuracy after 100 epochs.

I would **increase the magnitude of the learning rate hyper-parameter** to increase the magnitude of gradients. Increasing the magnitude of gradients

can move parameters closer to their optimal value faster and this can help achieve lower loss and higher validation accuracy in 100 epochs, or less.

3 Q3

3.1 Q3a

3.1.1 Q3ai Forward pass

Hidden layer neurons (H_1, H_2) are computed by a weighted sum, then a leaky ReLU activation. For example, H_1

$$\begin{aligned}\mathbf{z_1} = X_1 \cdot W_1[1] + X_2 \cdot W_1[2] + B_1 = \\ -1 \cdot 1 + 1 \cdot -1 + 0 = \\ \mathbf{-2} \\ \mathbf{H_1} = max(-2, \frac{1}{10} \cdot -2) = \mathbf{-\frac{2}{10}}\end{aligned}$$

Do a similar computation with W_2 and B_2 parameters to get $\mathbf{z_2} = \mathbf{2}$ and $\mathbf{H_2} = \mathbf{2}$.

Next, compute weighted sum of output node (z_3) using hidden layer outputs, W_3, and B_3.

$$\begin{aligned}\mathbf{z_3} = H_1 \cdot W_3[1] + H_2 \cdot W_3[2] + B_3 = \\ -\frac{2}{10} \cdot 1 + 2 \cdot 1 + (-1) = \mathbf{0.8}\end{aligned}$$

Finally, do sigmoid activation to compute $\hat{Y}$

$$\hat{\mathbf{Y}} = \sigma(z_3) = \frac{1}{1 + e^{-z_3}} = \mathbf{0.69}$$

3.1.2 Q3aii

Compute binary cross entropy loss using $\hat{Y}$ from forward pass and ground truth label Y.

$$\begin{aligned}\mathbf{L_{BCE}}(\hat{\mathbf{Y}}, \mathbf{Y}) = -Ylog(\hat{Y}) - (1 - Y)log(1 - \hat{Y}) = \\ 0 - 1 \cdot log(1 - 0.69) = \mathbf{1.171}\end{aligned}$$

3.1.3 Q3aiii

$\hat{Y}$	L	$\nabla\mathbf{W_3}$	$\nabla\mathbf{B_3}$	$\nabla\mathbf{W_2}$	$\nabla\mathbf{B_2}$	∇H_2	$\nabla\mathbf{W_1}$	$\nabla\mathbf{B_1}$	∇H_1	$\nabla\mathbf{X_1}$	$\nabla\mathbf{X_2}$
0.69	1.171	$[-0.138, 1.38]$	0.69	$[-0.69, 0.69]$	0.69	0.69	$[-0.069, 0.069]$	0.069	0.69	-.621	0.276

Let z_1, z_2, z_3 be net input (weighted sums plus biases) into neurons $H_1, H_2, neuron_{output}$. First, take derivative of L_{BCE} with respect to (w.r.t) $\hat{Y}$

$$\frac{\mathbf{dL}}{\mathbf{d\hat{Y}}} = \frac{\hat{Y} - Y}{\hat{Y}(1-\hat{Y})} = \frac{0.69}{0.69 \cdot 0.31} = \mathbf{3.23}$$

Next, compute derivative of loss w.r.t z_3.

$$\frac{\mathbf{dL}}{\mathbf{dz_3}} = \frac{dL}{d\hat{Y}} \cdot \frac{d\hat{Y}}{dz_3}$$

$$\text{where, } \frac{d\hat{Y}}{dz_3} = \sigma'(z_3) = \sigma(z) \cdot (1 - \sigma(z)) = 0.213$$

$$\text{so, } \frac{\mathbf{dL}}{\mathbf{dz_3}} = 3.23 \cdot 0.213 = \mathbf{0.69}$$

Partial derivatives of weights in weighted sum functions (all z functions) are simply the other operand it's multiplied by. And, we use the chain rule to relate this this result back to the loss function. For example,

$$\frac{\mathbf{dL}}{\mathbf{dW_3[1]}} = \frac{dL}{dz_3} \cdot \frac{dz_3}{dW_3[1]}$$

$$\text{where, } \frac{dz_3}{dW_3[1]} = H_1 = -\frac{2}{10}$$

$$\text{so, } \frac{\mathbf{dL}}{\mathbf{dW_3[1]}} = 0.69 \cdot -\frac{2}{10} = \mathbf{-0.138}$$

Loss derivative w.r.t biases is simple since the derivative of the net input function w.r.t a bias is just 1. So, the gradient passes through. For example,

$$\frac{\mathbf{dL}}{\mathbf{dB_3}} = \frac{dL}{dz_3} \cdot \frac{dz_3}{dB_3}$$

$$\text{where, } \frac{dz_3}{dB_3} = 1$$

$$\text{so, } \frac{\mathbf{dL}}{\mathbf{dB_3}} = \mathbf{0.69}$$

The derivative of hidden units (H_1, H_2) is taken w.r.t the net input of output neuron, z_3. Again, z_3 is a weighted sum so the weight derivative w.r.t H is just the associated weight.

$$\frac{\mathbf{dL}}{\mathbf{dH_1}} = \frac{dL}{dz_3} \cdot \frac{dz_3}{dH_1}$$

$$\text{where, } \frac{dz_3}{dH_1} = W_3[1]$$

$$\text{so, } \frac{dL}{dH_1} = 0.69 \cdot 1 = \mathbf{0.69}$$

Similarly, $\frac{\mathbf{dL}}{\mathbf{dH_2}} = \mathbf{0.69}$ since $W_3[2] = W_3[1] = 1$.

Furthermore, $\frac{dL}{dz_1}$ and $\frac{dL}{dz_2}$ showcase how leaky ReLUs derivatives are computed.

$$\frac{\mathbf{dL}}{\mathbf{dz_2}} = \frac{dL}{dH_2} \cdot \frac{dH_2}{dz_2}$$
$$\text{where, } \frac{dH_2}{dz_2} = 1 \text{ since } z_2 = 2 > 0$$
$$\text{so, } \frac{dL}{dz_2} = 0.69 \cdot 1 = \mathbf{0.69}$$

Similarly, for the net input of hidden layer neuron 2,

$$\frac{\mathbf{dL}}{\mathbf{dz_1}} = \frac{dL}{H_1} \cdot \frac{dH_1}{dz_1}$$
$$\text{where, } \frac{dH_1}{dz_1} = \alpha = 0.1 \text{ since } z_1 = -z < 0$$
$$\text{so, } \frac{dL}{dz_1} = 0.69 \cdot 0.1 = \mathbf{0.069}$$

... we process the W_1, W_2, B_1, B_2 exactly the same as W_3, B_3 except we use the net input functions z_1, z_2 corresponding to the their associated weights and biases to get.

$$\frac{\mathbf{dL}}{\mathbf{dW_1[1]}} = \frac{dL}{dz_1} \cdot \frac{dz_1}{dW_1[1]} = 0.069 \cdot X[1] = -\mathbf{0.069}$$
$$\frac{\mathbf{dL}}{\mathbf{dW_1[2]}} = \frac{dL}{dz_1} \cdot \frac{dz_1}{dW_1[2]} = 0.069 \cdot X[2] = \mathbf{0.069}$$
$$\frac{\mathbf{dL}}{\mathbf{dB_1}} = \frac{dL}{dz_1} \cdot \frac{dz_1}{dB_1} = 0.069 \cdot 1 = 0.069$$
$$\frac{\mathbf{dL}}{\mathbf{dW_2[1]}} = \frac{dL}{z_2} \cdot \frac{dz_2}{dW_2[1]} = 0.69 \cdot -1 = -\mathbf{0.69}$$
$$\frac{\mathbf{dL}}{\mathbf{dW_2[2]}} = \frac{dL}{z_2} \cdot \frac{dz_2}{dW_2[2]} = 0.69 \cdot 1 = \mathbf{0.69} \frac{\mathbf{dL}}{\mathbf{dB_2}} = \frac{dL}{dz_2} \cdot \frac{dz_2}{dB_2} = 0.69 \cdot 1 = \mathbf{0.69}$$

Finally, partial derivative of inputs w.r.t loss function requires summing the incoming gradients both from z_1, z_2 for each X_i.

$$\frac{\mathbf{dL}}{\mathbf{dX[1]}} = (\frac{dL}{dz_2} \cdot \frac{dz_2}{dX[1]}) + (\frac{dL}{dz_1} \cdot \frac{dz_1}{dX[1]}) = 0.69 \cdot W_2[1] + 0.069 \cdot W_1[1] = -0.69 + 0.069 = -\mathbf{0.621}$$
$$\frac{\mathbf{dL}}{\mathbf{dX[2]}} = (\frac{dL}{dz_2} \cdot \frac{dz_2}{dX[2]}) + (\frac{dL}{dz_1} \cdot \frac{dz_1}{dX[2]}) = 0.69 \cdot W_2[2] + 0.069 \cdot W_1[2] = 0.69 \cdot \frac{1}{2} + 0.069 \cdot -1 = \mathbf{0.276}$$

3.2 Q3b

3.2.1 Q3bi

The forward pass for the model described in Q3b is the following. Also, I denote the pre-sigmoid value at the output as R (meaning, $\hat{Y} = \sigma(R)$.

First, compute the hidden layers.

$$\mathbf{H_1} = W_1(X_1 + X_2) = 1.5(2 + (-1)) = \mathbf{1.5}$$
$$\mathbf{H_2} = X_1X_2X_3 + B_1 = 2 \cdot -1 \cdot 1 + 1 = \mathbf{-1}$$
$$\mathbf{H_3} = min(X_1, H_1) = min(2, 1.5) = \mathbf{1.5}$$
$$\mathbf{H_4} = W_2(\frac{H_2}{H_1}) = -3(\frac{-1}{1.5}) = \mathbf{2}$$
$$\mathbf{H_5} = max(H_4, H_1) = max(2, 1.5) = \mathbf{2}$$

Compute R, the pre-sigmoid value of the output node. Then, do sigmoid of R to get predicted output, $\hat{Y}$

$$\mathbf{R} = H_3 - H_5 = 1.5 - 2 = \mathbf{-0.5}$$
$$\mathbf{\hat{Y}} = \sigma(R) = \frac{1}{1 + e^{-R}} = \frac{1}{1 + e^{-(-0.5)}} = \mathbf{0.378}$$

Therefore, the model's predicted output is **0.378**.

3.2.2 Q3bii

Next, the binary cross entropy loss is computed using the predicted output, $\hat{Y} = 0.378$, and the ground truth label, $Y = 1$.

$$\mathbf{L_{BCE}}(\mathbf{\hat{Y}}, \mathbf{Y}) = -Ylog(\hat{Y}) - (1 - Y)log(1 - \hat{Y}) =$$
$$-log(0.378) - 0 = \mathbf{0.974}$$

3.2.3 Q3biii

$\hat{Y}$	L	$\bigtriangledown H_5$	$\bigtriangledown H_4$	$\bigtriangledown H_3$	$\bigtriangledown H_2$	$\bigtriangledown H_1$	$\bigtriangledown \mathbf{W_2}$	$\bigtriangledown \mathbf{W_1}$	$\bigtriangledown \mathbf{B_1}$	$\bigtriangledown X_3$	$\bigtriangledown X_2$	$\bigtriangledown X_1$
0.378	0.974	0.622	0.622	-0.622	-1.24	-1.45	-0.415	-1.45	-1.24	2.49	-4.67	-0.934

I drew a computational graph of the forward and backward operations and values. And, I will write out the computations.

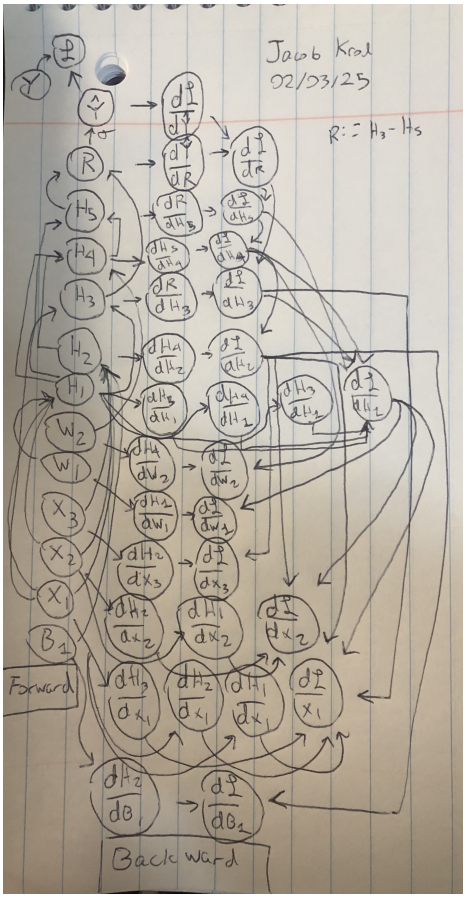

Figure 1: Computational graph of forward and backward pass of Q3b architecture.

To start backpropagation of the Q3b model, we first take the partial derivative of the loss function, L_{BCE} w.r.t $\hat{Y}$.

$$\frac{\mathbf{dL}}{\mathbf{d\hat{Y}}} = \frac{\hat{Y} - Y}{\hat{Y}(1-\hat{Y})} = \frac{0.378 - 1}{0.378(1-0.378)} = \mathbf{-2.65}$$

Next, I define a helper variable $R = H_3 - H_5$ which is the input to the sigmoid activation. It's derivative in the $\hat{Y}$ function is the derivative of the sigmoid function, then the chain rule relates the derivative the loss function.

$$\frac{\mathbf{dL}}{\mathbf{dR}} = \frac{dL}{d\hat{Y}} \cdot \frac{d\hat{Y}}{dR}$$

$$\text{where, } \frac{d\hat{Y}}{dR} = \sigma(R) * (1 - \sigma(R) = 0.235$$

$$\text{so, } \frac{dL}{dR} = -2.65 \cdot 0.235 = \mathbf{-0.622}$$

Next, partial derivative of H_3, H_5 is simplified by the helper intermediate variable $R = H_3 - H_5$.

$$\frac{\mathbf{dL}}{\mathbf{dH_5}} = \frac{dL}{dR} \cdot \frac{dR}{dH_5}$$

$$\text{where, } \frac{dR}{dH_5} = -1$$

$$\text{so, } \frac{dL}{dH_5} = -0.622 \cdot -1 = \mathbf{0.622}$$

$$\frac{\mathbf{dL}}{\mathbf{dH_3}} = \frac{dL}{dR} \cdot \frac{dR}{dH_3}$$

$$\text{where, } \frac{dR}{dH_3} = 1$$

$$\text{so, } \frac{dL}{dH_3} = \mathbf{-0.622}$$

The derivative of H_5 w.r.t H_4 requires taking the derivative of $H_5 = max(H_4, H_1)$. For max functions, the partial derivative w.r.t a parameter is 1 if that value is the greatest, else 0.

$$\frac{\mathbf{dL}}{\mathbf{dH_4}} = \frac{dL}{dH_5} \cdot \frac{dH_5}{dH_4}$$

$$\text{where, } \frac{dH_5}{dH_4} = 1 \text{ since } H_4 > H_1$$

$$\text{so, } \frac{dL}{dH_4} = \mathbf{0.622}$$

W_2 is an input only for H_4. Its partial derivative in the loss function is

$$\frac{\mathbf{dL}}{\mathbf{dW_2}} = \frac{dL}{dH_4} \cdot \frac{dH_4}{dW_2}$$

$$\text{where, } \frac{dH_4}{dW_2} = \frac{H_2}{H_1}$$

$$\text{so, } \frac{dL}{dW_2} = 0.622 \cdot \frac{-1}{1.5} = \mathbf{-0.415}$$

Similarly, H_2 is an input only for H_4.

$$\frac{\mathbf{dL}}{\mathbf{dH_2}} = \frac{dL}{dH_4} \cdot \frac{dH_4}{dH_2}$$

$$\text{where, } \frac{dH_4}{dH_2} = W_2 \cdot \frac{1}{H_1}$$

$$\text{so, } \frac{dL}{dH_2} = 0.622 * -3 * \frac{1}{1.5} = \mathbf{-1.24}$$

Since H_1 is an input to three functions in the forward pass, the contributions must be summed.

$$\frac{\mathbf{dL}}{\mathbf{dH_1}} = (\frac{dL}{dH_2} \cdot \frac{dH_5}{dH_1}) + (\frac{dL}{dH_4} \cdot \frac{dH_4}{dH_1}) + (\frac{dL}{dH_3} \cdot \frac{dH_3}{dH_1})$$

$$\text{where, } \frac{dH_5}{dH_1} = 0$$

$$\text{and, } \frac{dH_4}{dH_1} = W_2 \cdot H_2 \cdot -(\frac{1}{H_1^2}) = -3 \cdot -1 \cdot -(\frac{1}{1.5^2}) = -1.33$$

$$\text{and, } \frac{dH_3}{dH_1} = 1 \text{ since } H_1 < X_1$$

$$\text{so, } \frac{dL}{dH_1} = (-1.24 \cdot 0) + (0.622 \cdot -1.33) + (-0.622 \cdot 1) = \mathbf{-1.45}$$

W_1 is parameter only for H_1. Its partial derivative in the loss function is

$$\frac{\mathbf{dL}}{\mathbf{dW_1}} = \frac{dL}{dH_1} \cdot \frac{dH_1}{dW_1}$$

$$\text{where, } \frac{dH_1}{dW_1} = X_1 + X_2 = 1$$

$$\text{so, } \frac{dL}{dW_1} = -1.45 \cdot 1 = \mathbf{-1.45}$$

B_1 is a parameter only for H_2. Its partial derivative in the loss function just carries the gradient from H_2 since the derivative of the H_2 w.r.t B_1 is just 1.

$$\frac{\mathbf{dL}}{\mathbf{dB_1}} = \frac{dL}{dH_2} \cdot \frac{dH_2}{dB_1}$$

$$\text{where, } \frac{dH_2}{dB_1} = 1$$

$$\text{so, } \frac{dL}{dB_1} = \mathbf{-1.24}$$

Finally, the gradient for the inputs. For X_1 and X_2, the gradient in the loss function for each function they contribute to in the computational graph needs to be summed.

$$\mathbf{\frac{dL}{dX_3}} = \frac{dL}{dH_2} \cdot \frac{dH_2}{dX_3}$$

$$\text{where, } \frac{dH_2}{dX_3} = X_1 \cdot X_2$$

$$\text{so, } \frac{dL}{dX_3} = -1.24 \cdot 2 \cdot -1 = \mathbf{2.49}$$

$$\mathbf{\frac{dL}{dX_2}} = (\frac{dL}{dH_2} \cdot \frac{dH_2}{dX_2}) + (\frac{dL}{dH_1} \cdot \frac{dH_1}{dX_2})$$

$$\text{where, } \frac{dH_2}{dX_2} = X_1 \cdot X_3$$

$$\text{and, } \frac{dH_1}{dX_2} = W_1$$

$$\text{so, } \frac{dL}{dX_2} = (-1.24 \cdot X_1 \cdot X_3) + (-1.45 \cdot W_1) = \mathbf{-4.67}$$

$$\mathbf{\frac{dL}{dX_1}} = (\frac{dL}{dH_3} \cdot \frac{dH_3}{dX_1}) + (\frac{dL}{dH_2} \cdot \frac{dH_2}{dX_1}) + (\frac{dL}{dH_1} \cdot \frac{dH_1}{dX_1})$$

$$\text{where, } \frac{dH_3}{dX_1} = 1 \text{ since } X_1 < H_1$$

$$\text{and, } \frac{dH_2}{dX_1} = X_2 \cdot X_3$$

$$\text{and, } \frac{dH_1}{dX_1} = W_1$$

$$\text{so, } \frac{dL}{dX_1} = (-0.622 \cdot 1) + (-1.24 \cdot -1 \cdot 1) + (-1.45 \cdot 1.5) = \mathbf{-0.934}$$

4 Extra credit

4.1 EC.a

It can be shown that $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} = 2\sigma(2x) - 1$ using algebra.
First, apply the transformation to sigmoid function

$$\mathbf{2\sigma(2x) - 1} = \frac{2}{1 + e^{-2x}} - 1$$

$$\text{set 1 to } \frac{1 + e^{-2x}}{1 + e^{-2x}} \text{ and subtract}$$

$$= \frac{2 - (1 + e^{-2x})}{1 + e^{-2x}} = \mathbf{\frac{1 - e^{-2x}}{1 + e^{-2x}}}$$

Next, it can be shown that $\frac{1-e^{-2x}}{1+e^{-2x}}$ is equivalent to $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$.

First, rearrange the numerator. Use the trick of substituting 1 to rearrange the fraction.

$$1 - e^{-2x} = 1 - \frac{1}{e^{2x}} = \frac{e^{2x} - 1}{e^{2x}}$$

Similarly, rearrange the denomiator of the resulting $2\sigma(2x) - 1$ expression.

$$1 + e^{-2x} = 1 + \frac{1}{e^{2x}} = \frac{e^{2x} + 1}{e^{2x}}$$

Substitute the rearrangements back into the $2\sigma(2x) - 1$ expression. e^{2x} cancels out.

$$\frac{\frac{e^{2x}-1}{e^{2x}}}{\frac{e^{2x}+1}{e^{2x}}} = \frac{\mathbf{e^{2x} - 1}}{\mathbf{e^{2x} + 1}} = \mathbf{tanh(x)}$$

4.2 EC.b

Using $tanh(z)$ activation and L_{MSE} for Q3a, given that $z = 0.8$

$$\mathbf{tanh(0.8) = 0.664}$$

$$\mathbf{L_{MSE}} = \frac{1}{2}(0.664 - (-1))^2 = \mathbf{1.385}$$

Using $Y = -1$ instead of $Y = 0$ makes sense for binary classification when using **tanh since the range is** $[-1, 1]$. Since a binary classifier should separate the predicted value of positive and negative examples, it makes sense to train a model such that negative examples are as far away from positive examples as possible, $Y = -1$ for negative examples. Meanwhile, **sigmoid ranges from** $[0, 1]$ which motivates the use of $Y = 0$ for negative examples.

Next, using $tanh$ for final output neuron and L_{MSE} for loss in Q3b, given that the input to the final activation function is $-\frac{1}{2}$.

$$\mathbf{tanh(-\frac{1}{2}) = -0.462}$$

$$\mathbf{L_{MSE}} = \frac{1}{2}(-0.462 - (1))^2 = \mathbf{1.069}$$

4.3 EC.c