Submission Date: 29th November 2023

1 Activation Functions

1. Why do we use non-linear activation functions?
2. Compute $\frac{\partial g}{\partial z}$ with the following activation functions:

$$g = \mathrm{ReLU}(z) = \max(z, 0) \tag{1}$$

$$g = \mathrm{sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

$$g = \tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{3}$$

   Try to simplify the results, if possible.

3. Think of the advantages and disadvantages of the activation functions mentioned in point 2. In which cases would we prefer one over the other two?[1]
4. Consider you have a sample x with true label $y = A$, and the possible labels are A, B, C (multi-class classification problem). Given a prediction model that outputs the unnormalized log probability z per class with the following values $z_A = 7, z_B = 1, z_C = 2.2$: *i)* compute the Softmax output, *ii)* the predicted class, and *iii)* the log-likelihood loss.

2 Neural Networks

1. What is the difference between using basis functions (recall the non-linear functions in the previous lectures) and using neural networks on the input features?
2. You are given a dataset of 80 examples with 6 features and the outcomes (labels) {'Spring', 'Summer', 'Fall', 'Winter'}. Suppose we use a neural network with 3 hidden layers, each layer having 32 units. What is the number of weights (including biases) of the neural network?
3. Given the following dataset:

x	y
1	0
2	0.3
10	1

[1] To get a feeling on how activation functions can influence the outcome of a neural network you can go to https://playground.tensorflow.org/.

- You are given a neural network with 2 layers, 1 unit per layer and with ReLU as an activation function. In particular, $w_{1,1}^{(1)} = -0.1$, $w_{1,1}^{(2)} = 0.1$, the biases are initialized to 0 and the mean squared error is used as a loss function. Should you increase or decrease the value of $w_{1,1}^{(1)}$ to decrease the loss?
- (Bonus) Compute the gradient of the loss w.r.t. the weight $\frac{\partial J(w)}{\partial w_{1,1}^{(1)}}$. Based on the derivative should you increase or decrease the value of $w_{1,1}^{(1)}$?

4. Warmup: PyTorch installation

   PyTorch is a mature python library for deep learning. In this exercise you have to install PyTorch and verify the correct installation.

   To install PyTorch run `pip install torch`.

5. You will now implement a simple neural network whose output approximates the output of the XOR operation. At this point we are not concerned to generalize; we only care to learn a function that can classify each pair in $X = \{[0,0]^T, [0,1]^T, [1,0]^T, [1,1]^T\}$ to one of the classes in $Y = \{0,1\}$. Note that our input X is a matrix with dimensions $N \times D$ for N examples and D data dimensions.

   (a) Now you will define a 2-layer sequential model (Linear, ReLU, Linear) with 2 hidden units. 2-layer sequential model means it has an input layer, a hidden layer and an output layer. The input layer is not counted. Having 2 hidden units in a 2-layer sequential model means it has 2 units in its hidden layer. This model's output will therefore be computed as

   $f(x) = \max\{0, XW_1 + b_1\}W_2 + b_2$

   and we will be using

   $W_1 = \begin{bmatrix} 3.21 & -2.34 \\ 3.21 & -2.34 \end{bmatrix}$, $b_1 = \begin{bmatrix} -3.21 \\ 2.34 \end{bmatrix}$,

   $W_2 = \begin{bmatrix} 3.19 & -2.68 \\ 4.64 & -3.44 \end{bmatrix}$, $b_2 = \begin{bmatrix} -4.08 \\ 4.42 \end{bmatrix}$

   These parameters are actually the minimizers of some cost, e.g. one of the cost functions shown in the lecture. In the next lecture, you will learn how backpropagation is used to find these parameters using gradient descent.

   For this part you have to:

   - instantiate the sequential model.
   - propagate the input examples through the network to get the output.

   To handle categorical data, the `one_hot_encoding` utility function is provided in file *assignment_06.py*. The function is used to convert the input labels from class numbers to one-hot encodings. You will be required to use it now.

Fill in the gaps for function `run_model_on_xor` in file `assignment_06.py`.

Then, complete function `create_2unit_net` in file `assignment_06.py`.

Run the file `run_2unit_model.py` to see the outputs.

(b) (Bonus) Instantiate a new 2-layer sequential model with 3 hidden units this time. Assign the correct values to the model parameters such that the output of the neural network remains the same as in the model with 2 hidden units.

Fill in the gaps in function `create_3unit_net` in file `assignment_06.py`.

Run the file `run_3unit_models.py` to see the outputs.