CS/ECE/ME532 Assignment 10

1. Neural net functions

a) Sketch the function generated by the following 3-neuron ReLU neural network.

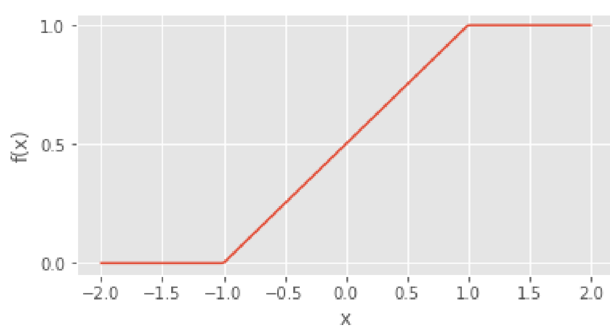
$$f(x) = 2(x - 0.5)_+ - 2(2x - 1)_+ + 4(0.5x - 2)+$$

where $x \in \mathbb{R}$ and where $(z)_+ = max(0, z)$ for any $z \in \mathbb{R}$. Note that this is a single-input, single-output function. Plot $f(x)$ vs x by hand.

b) Consider the continuous function depicted below. Approximate this function with ReLU neural network with 2 neurons. The function should be in the form

$$f(x) = \sum_{j=1}^{2} v_j(w_j x + b_j)_+$$

Indicate the weights and biases of each neuron and sketch the neural network function.



c) A neural network f_w can be used for binary classification by predicting the label as $\hat{y} = \text{sign}(f_w(\mathbf{x}))$. Consider a setting where $\mathbf{x} \in \mathbb{R}^2$ and the desired classifier is -1 if both elements of $\mathbf{x}$ are less than or equal to zero and $+1$ otherwise. Sketch the desired classification regions in the two-dimensional plane, and provide a formula for a ReLU network with 2-neurons that can produce the desired classification. For simplicity, assume in this questions that $\text{sign}(0) = -1$.

2. Gradients of a neural net. Consider a 2 layer neural network of the form $f(\boldsymbol{x}) = \sum_{j=1}^{J} v_j(\mathbf{w}_j^T \boldsymbol{x})_+$. Suppose we want to train our network on a dataset of N samples $\mathbf{x}_i$ with corresponding labels y_i, using a least squares loss function $\mathcal{L} = \sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2$. Derive the gradient descent update steps for the input weights $\mathbf{w}_j$ and output weights v_j.

3. **Compressing neural nets.** Large neural network models can be approximated by considering low rank approximations to weight matrices. The neural network $f(\boldsymbol{x}) = \sum_{j=1}^{J} \boldsymbol{v}_j(\mathbf{w}_j^T \boldsymbol{x})_+$ can be written as

$$f(\boldsymbol{x}) = \boldsymbol{v}^T(\mathbf{W}\boldsymbol{x})_+.$$

where $\mathbf{v}$ is a $J \times 1$ vector of the output weights and $\mathbf{W}$ is a $J \times d$ matrix with ith row $\mathbf{w}_j^T$. Let $\sigma_1, \sigma_2, \dots$ denote the singular values of $\mathbf{W}$ and assume that $\sigma_i \leq \epsilon$ for $i > r$. Let f_r denote the neural network obtained by replacing $\mathbf{W}$ with its best rank r approximation $\hat{\mathbf{W}}_r$. Assuming that $\boldsymbol{x}$ has unit norm, find an upper bound to the difference $\max_x |f(\boldsymbol{x}) - f_r(\boldsymbol{x})|$. (Hint: for any pair of vectors $\mathbf{a}$ and $\mathbf{b}$, the following inequality holds $\|\mathbf{a}_+ - \mathbf{b}_+\|_2 \leq \|\mathbf{a} - \mathbf{b}\|_2$).

4. **Face Emotion Classification with a three layer neural network**. In this problem we return to the face emotion data studied previously. You may find it very helpful to use code from an activity (or libraries such as Keras and Tensorflow).

   a) Build a classifier using a full connected three layer neural network with logistic activation functions. Your network should

   - take a vector $\boldsymbol{x} \in \mathbb{R}^{10}$ as input (nine features plus a constant offset),
   - have a single, fully connected hidden layer with 32 neurons
   - output a scalar $\widehat{y}$.

   Note that since the logistic activation function is always positive, your decision should be as follows: $\widehat{y} > 0.5$ corresponds to a 'happy' face, while $\widehat{y} \leq 0.5$ is not happy.

   b) Train your classifier using stochastic gradient descent (start with a step size of $\alpha = 0.05$) and create a plot with the number of epochs on the horizontal axis, and training accuracy on the vertical axis. Does your classifier achieve 0% training error? If so, how many epoch does it take for your classifier to achieve perfect classification on the training set?

   c) Find a more realistic estimate of the accuracy of your classifier by using 8-fold cross validation. Can you achieve perfect test accuracy?