Introduction to Machine Learning
Homework 7: Neural Networks

Prof. Sundeep Rangan and Yao Wang

1. Consider a neural network on a 3-dimensional input $\mathbf{x} = (x_1, x_2, x_3)$ with weights and biases:

$$W^{\rm H} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad b^{\rm H} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} \quad W^{\rm O} = [1, 1, -1, -1], \quad b^{\rm O} = -1.5.$$

   Assume the network uses the threshold activation function (1)

$$g_{\rm act}(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0. \end{cases} \tag{1}$$

   and the threshold output function (2):

$$\hat{y} = \begin{cases} 1 & z^{\rm O} \geq 0 \\ 0 & z^{\rm O} < 0. \end{cases} \tag{2}$$

   (a) Write the components of $\mathbf{z}^{\rm H}$ and $\mathbf{u}^{\rm H}$ as a function of (x_1, x_2, x_3). For each component j, indicate where in the (x_1, x_2, x_3) hyperplane $u_j^{\rm H} = 1$.

   (b) Write $z^{\rm O}$ as a function of (x_1, x_2, x_3). In what region is $\hat{y} = 1$? (You can describe in mathematical formulae).

2. Consider a neural network used for regression with a scalar input x and scalar target y,

$$z_j^{\rm H} = W_j^{\rm H} x + b_j^{\rm H}, \quad u_j^{\rm H} = \max\{0, z_j^{\rm H}\}, \quad j = 1, \ldots, N_h$$
$$z^{\rm O} = \sum_{k=1}^{N_h} W_k^{\rm O} u_k^{\rm H} + b^{\rm O}, \quad \hat{y} = g_{\rm out}(z^{\rm O}).$$

   The hidden weights and biases are:

$$\mathbf{W}^{\rm H} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{b}^{\rm H} = \begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix}.$$

   (a) What is the number N_h of hidden units? For each $j = 1, \ldots, N_h$, draw $u_j^{\rm H}$ as a function of x over some suitable range of values x. You may draw them on one plot, or on multiple plots.

(b) Since the network is for regression, you may choose the activation function $g_{\text{out}}(z^{\text{O}})$ to be linear. Given training data (x_i, y_i), $i = 1, \ldots, N$, formally define the loss function you would use to train the network?

(c) Using the output activation and loss function selected in part (b), set up the formulation to determine output weights and bias, $\mathbf{W}^{\text{O}}$, b^{O}, for the training data below. You should be able to find a closed form solution. Write a few line of python code to solve the problem.

x_i	-2	-1	0	3	3.5
y_i	0	0	1	3	3

(d) Based on your solution for the output weights and bias, draw $\hat{y}$ vs. x over some suitable range of values x. Write a few line of python code to do this.

(e) Write a function `predict` to output $\hat{y}$ given a vector of inputs `x`. Assume `x` is a vector representing a batch of samples and `yhat` is a vector with the corresponding outputs. Use the activation function you selected in part (b), but your function should take the weights and biases for both layers as inputs. Clearly state any assumptions on the formats for the weights and biases. Also, to receive full credit, you must not use any for loops.

3. Consider a neural network that takes each input $\mathbf{x}$ and produces a prediction $\hat{y}$ given

$$\begin{aligned} z_j &= \sum_{k=1}^{N_i} W_{jk} x_k + b_j, \quad u_j = 1/(1+\exp(-z_j)), \quad j = 1, \ldots, M, \\ \hat{y} &= \frac{\sum_{j=1}^{M} a_j u_j}{\sum_{j=1}^{M} u_j}, \end{aligned} \tag{3}$$

where M is the number of hidden units and is fixed (i.e. not trainable). To train the model, we get training data $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$.

(a) Rewrite the equations (3) for the batch of inputs $\mathbf{x}_i$ from the training data. That is, correctly add the indexing i, to the equations.

(b) If we use a loss function,

$$L = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2,$$

draw the computation graph describing the mapping from $\mathbf{x}_i$ and parameters to L. Indicate which nodes are trainable parameters.

(c) Compute the gradient $\partial L / \partial \hat{y}_i$ for all i.

(d) Suppose that, in backpropagation, we have computed $\partial L / \partial \hat{y}_i$ for all i, represented as $\partial L / \partial \hat{\mathbf{y}}$. Describe how to compute the components of the gradient $\partial L / \partial \mathbf{u}$.

(e) Suppose that we have computed the gradient $\partial L / \partial \mathbf{u}$, describe how would you compute the gradient $\partial L / \partial \mathbf{z}$.

(f) Suppose that we have computed the gradient $\partial L / \partial \mathbf{z}$, describe how would you compute the gradient $\partial L / \partial W_{jk}$ and $\partial L / \partial b_j$.

(g) Put all above together, describe how would you compute the gradient $\partial L / \partial W_{jk}$ and $\partial L / \partial b_j$.

(h) Write a few lines of python code to implement the gradients $\partial L / \partial \mathbf{u}$ (as in part (d)), given the gradient $\partial L / \partial \hat{\mathbf{y}}$. Indicate how you represent the gradients. Full credit requires that you avoid for-loops.