CSE 616: Assignment #1 (Due April 9, 2022)

Suppose we have a small dense neural network as is shown in fig. 1. The

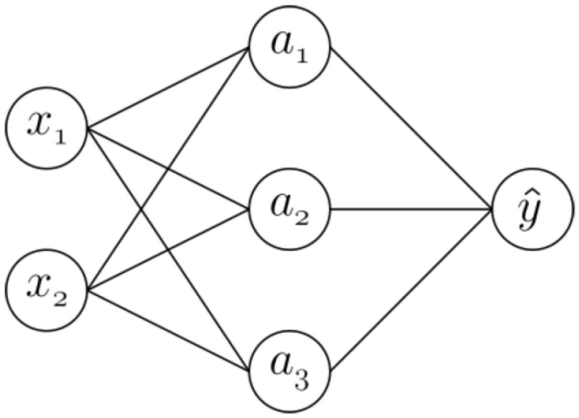

Figure 1: A small dense neural network

input vector $x = [x_1, x_2]^\top$ and the associated ground truth y, are

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad y = 32.$$

In the first layer we have the following weight and bias parameters

$$\begin{pmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \\ 2 & -1 & 1 \end{pmatrix}, \quad \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

In the second layer we have the following weight and bias parameters

$$\begin{pmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \\ w_{31}^{[2]} \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \quad b_1^{[2]} = 1.$$

1a

Compute the value of the network output, $\hat{y}$, when the activation functions in the first and second layer are identity functions.

1b

Compute the value of the network output, $\hat{y}$, when the activation functions in the first and the second layer are ReLU functions.

1c

Given a squared error loss function $J = (\hat{y} - y)^2$, and *identity functions* as activations in all nodes (as in task **1a** above), compute the value of the following four partial derivatives

$$\frac{\partial J}{\partial b_1^{[2]}},$$
$$\frac{\partial J}{\partial w_{21}^{[2]}},$$
$$\frac{\partial J}{\partial b_2^{[1]}},$$
$$\frac{\partial J}{\partial w_{13}^{[1]}}.$$

1d

Given everything as in task **1c**, perform one step with gradient descent with learning rate 2, and compute the updated values of the two parameters

$$b_2^{[1]},$$
$$w_{13}^{[1]}.$$

1e

Splitting the available dataset is a common way to estimate the out-of-sample error. If you check multiple models on the test set and select the best performing model, will this be a good indicator of the out-of-sample error? Justify your answer.

2

In the case where $f : \mathbb{R}^m \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, and $x \in \mathbb{R}^n$, the derivative of f

$$\begin{aligned} f(g(x)) &= f(g_1(x), \dots, g_m(x)) \\ &= f(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)) \end{aligned}$$

w.r.t. one of the components of x, can be given by a generalisation of the above chain rule

$$\frac{\partial f}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial f}{\partial g_j} \frac{\partial g_j}{\partial x_i}.$$

Compute the derivatives $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$ when

$$\begin{cases} f &= \sin g_1 + g_2^2 \\ g_1 &= x_1 e^{x_2} \\ g_2 &= x_1 + x_2^2. \end{cases}$$

3

1. Compute the derivative of the function $f(z)$ with respect to z $\left(\text{i.e., } \frac{df}{dz}\right)$, where

$$f(z) = \frac{1}{1 + e^{-z}}$$

2. Compute the derivative of the function $f(w)$ with respect to w_i, where $w, x \in \mathbb{R}^{\mathrm{D}}$ and

$$f(w) = \frac{1}{1 + e^{-w^T x}}$$

3. Compute the derivative of the loss function $J(w)$ with respect to w, where

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} \left| w^T x^{(i)} - y^{(i)} \right|$$

4. Compute the derivative of the loss function $J(w)$ with respect to w, where

$$J(w) = \frac{1}{2} \left[\sum_{i=1}^{m} \left(w^T x^{(i)} - y^{(i)} \right)^2 \right] + \lambda \|w\|_2^2$$

5. Compute the derivative of the loss function $J(w)$ with respect to w, where

$$J(w) = \sum_{i=1}^{m} \left[y^{(i)} \log \left(\frac{1}{1+e^{-w^T x^{(i)}}} \right) + \left(1 - y^{(i)}\right) \log \left(1 - \frac{1}{1+e^{-w^T x^{(i)}}} \right) \right]$$

6. Compute $\nabla_w f$, where $f(w) = \tanh\left[w^T x\right]$.

Programming:

1. You will use the Wine Quality dataset. Use only the *red wine* data. The goal is to find the quality score of some wine based on its attributes. Write your code in a script *neural_net.py*.
   Do not use any DL framework.

1) First, download the *winequality-red.csv* file, load it, and divide the data into a training and test set using approximately 50% for training. Standardize the data, by computing the mean and standard deviation for each feature dimension using the training set only, then subtracting the mean and dividing by the stdev for each feature and each sample. Append a 1 for each feature vector, which will correspond to the bias that our model learns. Set the number of hidden units, the number of iterations to run, and the learning rate.
2) Call the *backward* function to construct and train the network. Use 1000 iterations and 30 hidden neurons.
3) Then call the *forward* function to make predictions and compute the root mean squared error between predicted and ground-truth labels. Report this number in a file *report.pdf/docx*
4) Experiment with three different values of the learning rate. For each, plot the error over time (output by backward above). Include these plots in your report.