Homework 2

Neural Networks

Computer Vision, SE-803, CS-867 (Spring 2022)

DUE: Sunday, May 29, 2022, 11:59pm

Problem: Designing a Neural Networks (14 points)

Note: For each of these problems, write out the equation required to calculate each value in terms of the variables (a_j, z_j, b_k, etc.) before proceeding to calculate the numerical value. Round up the final answers to four decimal places and drop trailing zeros. You will see that some questions have zero marks. These are the questions you have to do on your own since the aim is not to burden you rather show you that these are the minimum necessary steps to design a neural network. Learn to enjoy doing it. It's the most power tool of last 20 years.

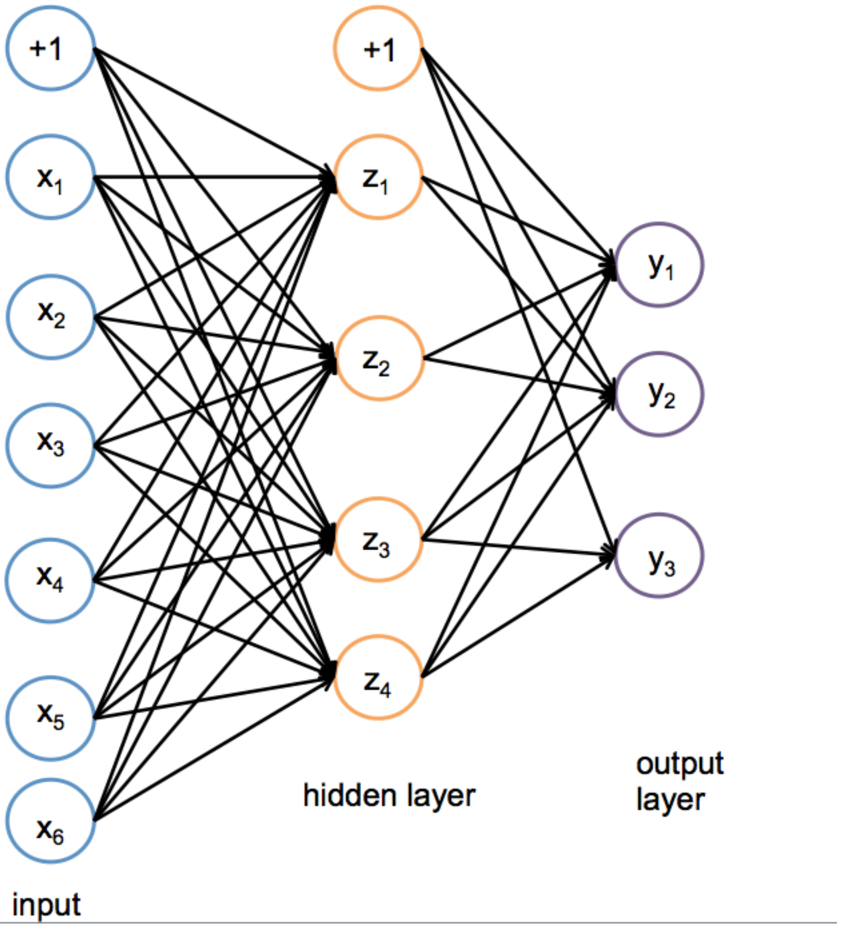

Figure 1: A One Hidden Layer Neural Network

Network Overview

Consider the neural network with one hidden layer shown in Figure 1. The input layer consists of 6 features $\mathbf{x} = [x_1, ..., x_6]^T$, the hidden layer has 4 nodes $\mathbf{z} = [z_1, ..., z_4]^T$, and the output layer is a probability distribution $\hat{\mathbf{y}} = [y_1, y_2, y_3]^T$ over 3 classes. We also add a bias to the input, $x_0 = 1$ and the hidden layer $z_0 = 1$, both of which are fixed to 1.

We adopt the following notation:

1. Let $\mathbf{W}^{(1)}$ the matrix of weights from the inputs to the hidden layer.
2. Let $\mathbf{W}^{(2)}$ the matrix of weights from the hidden layer to the output layer.
3. Let $w_{i,j}^{(1)}$ represent the weight going *to* the node z_i in the hidden layer *from* the node x_j in the input layer (e.g. $w_{1,2}^{(1)}$ is the weight from x_2 to z_1)
4. Let $w_{k,i}^{(2)}$ represent the weight going *to* the node y_k in the output layer *from* the node z_i in the hidden layer.
5. We will use a *sigmoid activation function (σ)* for the hidden layer and a *softmax* for the output layer.

Network Details

Equivalently, we define each of the following.

The input:

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T \tag{1}$$

Linear combination at first (hidden) layer:

$$a_i = w_{i,0}^{(1)} + \sum_{j=1}^{6} w_{i,j}^{(1)} x_j, \quad i \in \{1, \dots, 4\} \tag{2}$$

Activation at first (hidden) layer:

$$z_i = \sigma(a_i) = \frac{1}{1 + \exp(-a_i)}, \quad i \in \{1, \dots, 4\} \tag{3}$$

Linear combination at second (output) layer:

$$b_k = w_{k,0}^{(1)} + \sum_{i=1}^{4} w_{k,i}^{(1)} z_i, \quad k \in \{1, \dots, 3\} \tag{4}$$

Activation at second (output) layer:

$$\hat{y}_k = \frac{\exp(b_k)}{\sum\limits_{l=1}^{3} \exp(b_l)}, \quad k \in \{1, \dots, 3\} \tag{5}$$

Note that the linear combination equations can be written equivalently as the product of the weight matrix with the input vector. We can even fold in the bias term $w_0^{(1)}$ by thinking of $x_0 = 1$, and fold in $w_0^{(2)}$ by thinking of $z_0 = 1$.

Loss

We will use cross entropy loss, $\ell(\hat{\mathbf{y}}, \mathbf{y})$. If $\mathbf{y}$ represents our target (true) output, which will be a one-hot vector representing the correct class, and $\hat{\mathbf{y}}$ represents the output of the network, the loss is calculated as:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{3} y_k \log(\hat{y}_k) \tag{6}$$

1. **[6 pts]** In the following questions you will derive the matrix and vector forms of the previous equations which define our neural network. These are what you should hope to program in order to avoid excessive loops and large run times.
   When working these out it is important to keep a note of the vector and matrix dimensions in order for you to easily identify what is and isn't a valid multiplication.
   Suppose you are given a training example: $\mathbf{x}^{(1)} = [x_1, x_2, x_3, x_4, x_5, x_6]^T$ with **label class 2**, so $\mathbf{y}^{(1)} = [0, 1, 0]^T$. We initialize the network weights as:

$$\mathbf{W^{(0)}} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & w_{4,5} & w_{4,6} \end{bmatrix}^{(1)}$$

$$\mathbf{W^{(1)}} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \end{bmatrix}^{(2)}$$

   We want to also consider the bias term and the weights on the bias terms ($w_{j,0}^{(1)}$ and $w_{k,0}^{(2)}$). To account for these we can add a new column to the beginning of our initial weight matrices.

$$\mathbf{W} = \begin{bmatrix} w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} \\ w_{4,0} & w_{4,1} & w_{4,2} & w_{4,0} & w_{4,4} & w_{4,5} & w_{4,6} \end{bmatrix}^{(1)}$$

$$\mathbf{W} = \begin{bmatrix} w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \end{bmatrix}^{(2)}$$

And we can set our first value of our input vectors to always be 1 ($x_0^{(i)} = 1$), so our input becomes:

$$\mathbf{x}^{(1)} = [1, x_1, x_2, x_3, x_4, x_5, x_6]^T$$

(a) **[1 pt]** By examining the shapes of the initial weight matrices, how many neurons do we have in the first hidden layer of the neural network? (Not including the bias neuron)

(b) **[1 pt]** How many output neurons will our neural network have?

(c) **[1 pt]** What is the vector **a** whose elements are made up of the entries a_i in equation (3). Write your answer in terms of w and $x^{(1)}$.

(d) **[1 pt]** What is the vector **z** whose elements are made up of the entries z_i in equation (4)? Write your answer in terms of **a**.

(e) **[1 pt]** **Select one:** We cannot take the matrix multiplication of our weights $\mathbf{W}^{(2)}$ and our vector $\mathbf{z}$ since they are not compatible shapes. Which of the following would allow us to take the matrix multiplication of $\mathbf{W}^{(2)}$ and $\mathbf{z}$ such that the entries of the vector $\mathbf{b} = \mathbf{W}^{(2)} * \mathbf{z}$ are equivalent to the values of b_k in equation (5)?

- ○ Remove the last column of $\mathbf{W}^{(2)}$
- ○ Remove the first row of $\mathbf{z}$
- ○ Append a value of 1 to be the first entry of $\mathbf{z}$
- ○ Append an additional column of 1's to be the first column of $\mathbf{W}^{(2)}$
- ○ Append a row of 1's to be the first row of $\mathbf{W}^{(2)}$
- ○ Take the transpose of $\mathbf{W}^{(2)}$

(f) **[1 pt]** What are the entries of the output vector $\hat{\mathbf{y}}$? Your answer should be written in terms of b_1, b_2, b_3.

2. **[0 pts]** We will now derive the matrix and vector forms for the backpropagation algorithm.

$$\frac{d\ell}{d\mathbf{W}^{(1)}} =$$

The mathematics which you have to derive in this section jump significantly in difficultly, you should always be examining the shape of the matrices and vectors and making sure that you are comparing your matrix elements with calculations of individual derivatives to make sure they match (e.g. the element of the matrix $(\frac{d\ell}{dw})_{2,1}$ should be equal to $\frac{d\ell}{dw_{2,1}}$). Recall that ℓ is our loss function defined in equation (6)

(a) **[0 pts]** The derivative of the softmax function with respect to b_k is as follows:

$$\frac{d\hat{y}_l}{db_k} = \hat{y}_l(\mathbb{I}[k = l] - \hat{y}_k)$$

where $\mathbb{I}[k = l]$ is an indicator function such that if $k = l$ then it it returns value 1 and 0 otherwise. Using this, write the derivative $\frac{d\ell}{db_k}$ in a smart way such that you do not need this indicator function. Write your solutions in terms of $\hat{\mathbf{y}}, \mathbf{y}$.

(b) **[0 pts]** What are the elements of the vector $\frac{d\ell}{db}$? (Recall that $\mathbf{y}^{(1)} = [0, 1, 0]^T$)

(c) **[0 pt]** What is the derivative $\frac{d\ell}{d\mathbf{W}^{(2)}}$? Your answer should be in terms of $\frac{d\ell}{d\mathbf{b}}$ and $\mathbf{z}$.

(d) **[0 pts]** What is the derivative $\frac{d\ell}{d\mathbf{a}}$ in terms of $\frac{d\ell}{d\mathbf{z}}$ and $\mathbf{z}$

(e) **[0 pt]** What is the matrix $\frac{d\ell}{d\mathbf{W}^{(1)}}$? Your answer should be in terms of $\frac{d\ell}{d\mathbf{a}}$ and $x^{(1)}$.

Prediction

When doing prediction, we will predict the argmax of the output layer. For example, if $\hat{\mathbf{y}}$ is such that $\hat{y}_1 = 0.3,\ \hat{y}_2 = 0.2,\ \hat{y}_3 = 0.5$ we would predict class 3 for the input $\mathbf{x}$. If the true class from the training data $\mathbf{x}$ was 2 we would have a one-hot vector $\mathbf{y}$ with values $y_1 = 0,\ \ y_2 = 1,\ \ y_3 = 0$.

1. [8 pts] We initialize the weights as:

$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 & 1 & -1 & -1 & 0 & -1 \\ 3 & 1 & 0 & 1 & 0 & 2 \\ 1 & 2 & -1 & 0 & 2 & -1 \\ 2 & 0 & 2 & 1 & -2 & 1 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} 3 & -1 & 2 & 1 \\ 1 & -1 & 2 & 2 \\ 1 & -1 & 1 & 1 \end{bmatrix}$$

And weights on the bias terms ($w_{j,0}^{(1)}$ and $w_{j,0}^{(2)}$) are initialized to 1.

You are given a training example $\mathbf{x}^{(1)} = [1, 1, 0, 0, 1, 1]^T$ with label class 2, so $\mathbf{y}^{(1)} = [0, 1, 0]^T$. Using the initial weights, run the feed forward of the network over this training example (without rounding during the calculation) and then answer the following questions.

(a) **[1 pt]** What is the value of a_1?

(b) [**1 pt**] What is the value of z_1?

(c) [**1 pt**] What is the value of a_3?

(d) [**1 pt**] What is the value of z_3?

(e) [**1 pt**] What is the value of b_2?

(f) [**1 pt**] What is the value of $\hat{y}_2$?

(g) [**1 pt**] Which class value we would predict on this training example?

(h) [**1 pt**] What is the value of the total loss on this training example?

2. [0 pts] Now use the results of the previous question to run backpropagation over the network and update the weights. Use the learning rate $\eta = 1$.

   Do your backpropagation calculations without any rounding then answer the following questions: (in your final responses round to four decimal places)

   (a) **[0 pts]** What is the updated value of $w^{(2)}_{2,1}$?

   (b) **[0 pts]** What is the updated weight of the hidden layer bias term applied to y_1 (i.e. $w^{(2)}_{1,0}$)?

   (c) **[0 pts]** What is the updated value of $w^{(1)}_{3,4}$?

   (d) **[0 pts]** If we ran backpropagation on this example for a large number of iterations and then ran feed forward over the same example again, which class would we predict?