CS182 Introduction to Machine Learning, Spring 2023

Homework 3

(Due Tuesday, Apr. 18 at 11:59 pm (CST))

1. [15 points] *(Multi-layer neural network and back-propagation)* Figure 1 below shows an example of a multi-layer neural network, with 1 input layer (2 input units x_1, x_2), h_3 and h_4 are hidden units and 1 output unit h_5. w_{ij} means the weight from unit i to unit j, w_{0j} means the bias, and a_i is the activation function.

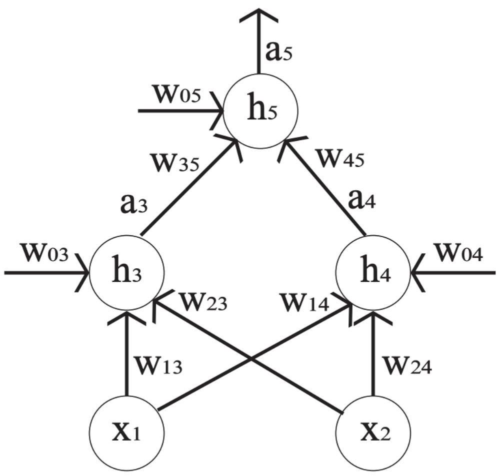

Figure 1: Multi-layer neural network architecture

When an input is passed into a neural network, the signals are passed along the network by following the connections between the units, starting at the input layer and ending in the output layer. This process is referred to as **forward propagation**. In order for the neural network to learn, it must update its weights across its multiple layers. The common approach for neural network learning is called **back-propagation**, which relies on the gradient descent method.

(a) [2.5 points] Imagine you are using backprop to do weight updates for an example for which $x_1 = 1$ and $x_2 = 1$. Use Δw_{ij} to refer to the amount weight w_{ij} changes as the result of the update. If Δw_{03} $= d_3$ and $\Delta w_{04} = d_4$, what are Δw_{13}, Δw_{23}, Δw_{14}, and Δw_{24}?

(b) [7.5 points] This question will have you simulate the process of forward propagation and back-propagation on an example for which $x_1 = 1$ and $x_2 = 1$ and whose desired output is $y = 1$. The values for the various weights are as follows:

- Weight and bias for h_3: $w_{03} = 1$, $w_{13} = 1$, $w_{23} = 1$
- Weight and bias for h_4: $w_{04} = 2$, $w_{14} = 1$, $w_{24} = 1$
- Weight and bias for h_5: $w_{05} = 2$, $w_{35} = 1$, $w_{45} = 1$

For the rest of this question you'll be using the algorithm for forward and back-propagation given in class. Note: **assume the learning rate α is 1.**

What is the output and error after the forward propagation, and what is the gradient and updated value after the back-propagation for each weight with the logistic activation function $g(z) = \frac{1}{1+e^{-z}}$. Recall that $g'(z) = g(z)(1 - g(z))$. Please also give the network's new output and error after the update, and keep your answer to three significant digits after the decimal.

2. [30 points] *(Feed Forward and Backpropagation)*

Network Overview Consider the neural network with one hidden layer shown in Figure 2. The input layer consists of 6 features $= [x_1, ..., x_6]^T$, the hidden layer has 4 nodes $= [z_1, ..., z_4]^T$, and the output layer is a probability distribution $= [y_1, y_2, y_3]^T$ 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.

$\boldsymbol{\alpha}$ is the matrix of weights from the inputs to the hidden layer and $\boldsymbol{\beta}$ is the matrix of weights from the hidden layer to the output layer. $\alpha_{j,i}$ represents the weight going *to* the node z_j in the hidden layer *from* the node x_i in the input layer (e.g. $\alpha_{1,2}$ is the weight from x_2 to z_1), and $\boldsymbol{\beta}$ is defined similarly. We will use a sigmoid activation function for the hidden layer and a softmax for the output layer.

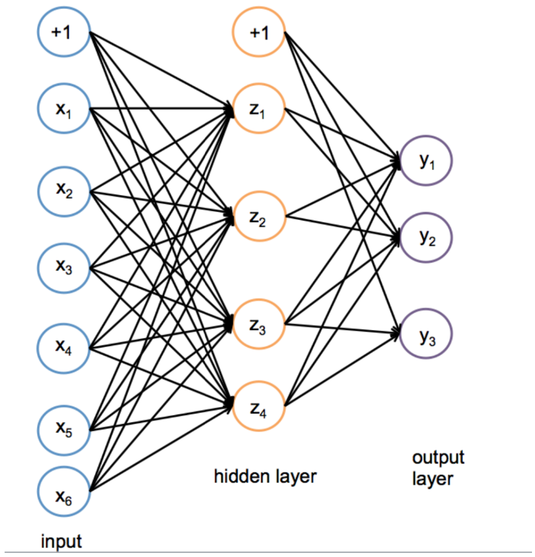

Figure 2: A One Hidden Layer Neural Network

Network Details Equivalently, we define each of the following.

The input:

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

Linear combination at first (hidden) layer:

$$a_j = \alpha_0 + \sum_{i=1}^{6} \alpha_{j,i} * x_i, \ \forall j \in \{1, \dots, 4\} \tag{2}$$

Activation at first (hidden) layer:

$$z_j = \sigma(a_j) = \frac{1}{1 + \exp(-a_j)}, \ \forall j \in \{1, \dots, 4\} \tag{3}$$

Linear combination at second (output) layer:

$$b_k = \beta_0 + \sum_{j=1}^{4} \beta_{k,j} * z_j, \ \forall k \in \{1, \dots, 3\} \tag{4}$$

Activation at second (output) layer:

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

Note that the linear combination equations can be written equivalently as the product of the transpose of the weight matrix with the input vector. We can even fold in the bias term α_0 by thinking of $x_0 = 1$, and fold in β_0 by thinking of $z_0 = 1$.

Loss We will use cross-entropy loss, $\ell(,)$. If represents our target output, which will be a one-hot vector representing the correct class, and ˆ represents the output of the network, the loss is calculated by:

$$\ell(,) = -\sum_{i=1}^{3} y_i \log(\hat{y}_i) \tag{6}$$

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

(a) [8 points] We initialize the weights as:

$$\boldsymbol{\alpha} = \begin{bmatrix} 1 & 2 & -3 & 0 & 1 & -3 \\ 3 & 1 & 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 2 & 1 & -2 & 2 \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} 1 & 2 & -2 & 1 \\ 1 & -1 & 1 & 2 \\ 3 & 1 & -1 & 1 \end{bmatrix}$$

And weights on the bias terms ($\alpha_{j,0}$ and $\beta_{j,0}$) are initialized to 1.
You are given a training example $^{(1)} = [1, 1, 0, 0, 1, 1]^T$ with label class 2, so $^{(1)} = [0, 1, 0]^T$. Using the initial weights, run the feed forward of the network over this example (without rounding during the calculation) and then answer the following questions.

i. What is a_1?

ii. What is z_1?

iii. What is a_3?

iv. What is z_3?

v. What is b_2?

vi. What is $\hat{y}_2$?

vii. Which class would we predict on this example?

viii. What is the total loss on this example?

(b) [10 points] Now use the results of the previous question to run backpropagation over the network and update the weights. Use learning rate $\eta = 1$.

Do your backpropagation calculations without rounding then answer the following questions, then in your responses, round to 4 decimal places.

i. What is the updated value of $\beta_{2,1}$?

ii. What is the updated weight of the hidden layer bias term applied to y_1 (i.e. $\beta_{1,0}$)?

iii. What is the updated value of $\alpha_{3,4}$?

iv. What is the updated weight of the input layer bias term applied to z_2 (i.e. $\alpha_{2,0}$)?

v. 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?

(c) [12 points] Let us now introduce regularization into our neural network. For this question, we will incorporate L2 regularization into our loss function $\ell(\hat{y}, y)$, with the parameter λ controlling the weight given to the regularization term.

i. Write the expression for the regularized loss function of our network after adding L2 regularization (**Hint:** Remember that bias terms should not be regularized!)

ii. Compute the regularized loss for training example $x^{(1)}$ (assume $\lambda = 0.01$ and use the weights before backpropagation)

Suppose the weight initialization for α is changed to the following:

$$\boldsymbol{\alpha} = \begin{bmatrix} 10 & 20 & -30 & 0 & 10 & -30 \\ 30 & 10 & 20 & 10 & 0 & 20 \\ 20 & 20 & 20 & 20 & 20 & 10 \\ 10 & 0 & 20 & 10 & -20 & 20 \end{bmatrix}$$

β and bias terms are not changed.

iii. Report the non-regularized loss for the network on training example $x^{(1)}$

iv. Report the regularized loss for the network on training example $x^{(1)}$ ($\lambda = 0.01$)

v. For a network which uses the regularized loss function, write the gradient update equation for $\alpha_{j,i}$. You may use $\frac{\partial \ell(\hat{y},y)}{\partial \alpha_{j,i}}$ to denote the gradient update w.r.t non-regularized loss and η to denote the learning rate.

vi. Based on your observations from previous questions, **select all statements which are true**:

☐ The non-regularized loss is always higher than the regularized loss

☐ As weights become larger, the regularized loss increases faster than non-regularized loss

☐ On adding regularization to the loss function, gradient updates for the network become larger

☐ When using large initial weights, weight values decrease more rapidly for a network which uses regularized loss

☐ None of the above

3. [20 points] *(Update Rules)* Consider a two-class classification problem with L training samples (x_1, y_1),...,(x_L, y_L), where the input $\mathbf{x}_l \in \mathbb{R}^N, l = 1, \ldots, L$ contains N features. Suppose we use the logistic regression following the batch learning scheme, where the output is computed as:

$$\hat{y}_l = \text{sigmod}\left(\mathbf{w}^T \mathbf{x}_l\right)$$

Derive the update rule for parameter w under the following settings:

(a) [10 points] $y_0 \in \{0, 1\}, l = 1, \ldots, L$, where $P(y_l = 1 \mid \hat{y}_l) = \text{sigmod}\left(\mathbf{w}^T \mathbf{x_l}\right)$ and $P(y_l = 0 \mid \hat{y}_0) = 1 - P(y_0 = 1 \mid \hat{y}_l)$

(b) [10 points] $y_l \in \{-1, 1\}, l = 1, \ldots, L$, where $P(y_l = 1 \mid \hat{y}_l) = \text{sigmod}\big(\mathbf{w}^T \mathbf{x_l}\big)$ and $P(y_l = -1 \mid \hat{y}_l) = \text{sigmod}\big(-\mathbf{w}^T \mathbf{x_l}\big)$