10-315 Intro to Machine Learning HW6

INSTRUCTIONS

- **Due: Tuesday, 24 March 2020 at 11:59 PM EDT.**
- **Format:** Complete this pdf with your work and answers. Whether you edit the latex source, use a pdf annotator, or hand write / scan, make sure that your answers (tex'ed, typed, or handwritten) are within the dedicated regions for each question/part. If you do not follow this format, we may deduct points.
- **How to submit:** Submit a pdf with your answers on Gradescope. Log in and click on our class 10-315, click on the appropriate *Written* assignment, and upload your pdf containing your answers. Don't forget to submit the associated *Programming* component on Gradescope if there is any programming required.
- **Policy:** See the course website for homework policies and Academic Integrity.

Name	
Andrew ID	
Hours to complete (both written and programming)?	

For staff use only

Q1	Q2	Total
/26	/10	/36

Q1. [26 pts] Example Feed Forward and Backpropagation

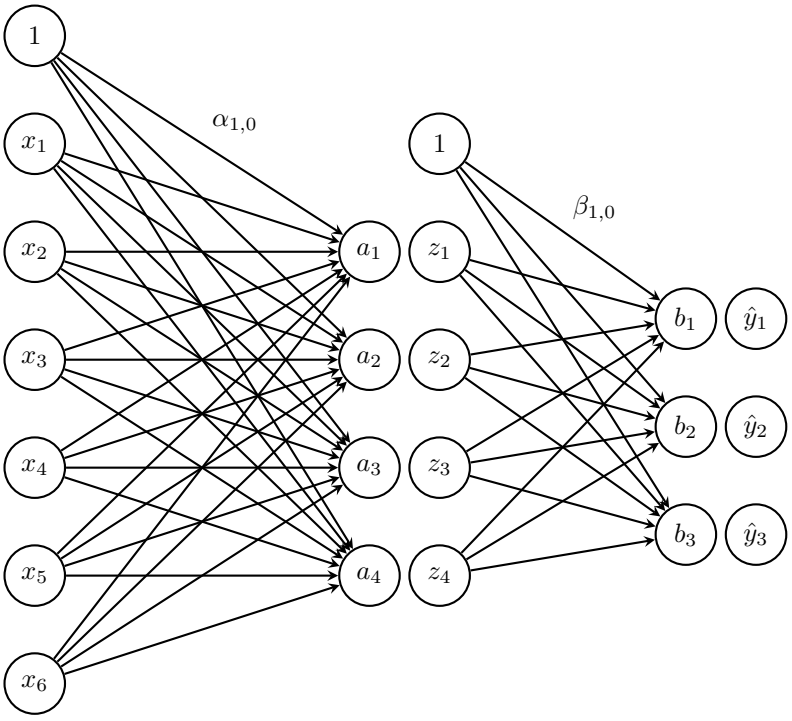

Figure 1: A One Hidden Layer Neural Network

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 $\mathbf{y} = [y_1, y_2, y_3]^T$ 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.

$\boldsymbol{\alpha}$ 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.

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 the first (hidden) layer:

$$a_j = \alpha_{j,0} + \sum_{i=1}^{6} \alpha_{j,i} * x_i, \ \forall j \in \{1, \ldots, 4\} \tag{2}$$

Activation at the first (hidden) layer:

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

Linear combination at the second (output) layer:

$$b_k = \beta_{k,0} + \sum_{j=1}^{4} \beta_{k,j} * z_j, \ \forall k \in \{1, \ldots, 3\} \tag{4}$$

Activation at the second (output) layer:

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

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 α_0 by thinking of $x_0 = 1$, and fold in $\beta_{j,0}$ 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 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 by:

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

For the below questions use natural log in the equation.

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 $\mathbf{y}$ with values $y_1 = 0$, $y_2 = 1$, $y_3 = 0$.

(a) In the following questions you will derive the matrix and vector forms of the previous equations which define out neural network. These are what you should hope to program in order to keep your program under the Autolab time-out.

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 an 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:

$$\boldsymbol{\alpha}^* = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} & \alpha_{4,5} & \alpha_{4,6} \end{bmatrix}$$

$$\boldsymbol{\beta}^* = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \beta_{1,3} & \beta_{1,4} \\ \beta_{2,1} & \beta_{2,2} & \beta_{2,3} & \beta_{2,4} \\ \beta_{3,1} & \beta_{3,2} & \beta_{3,3} & \beta_{3,4} \end{bmatrix}$$

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

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{1,0} & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} \\ \alpha_{2,0} & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} \\ \alpha_{3,0} & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} \\ \alpha_{4,0} & \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,0} & \alpha_{4,4} & \alpha_{4,5} & \alpha_{4,6} \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{1,0} & \beta_{1,1} & \beta_{1,2} & \beta_{1,3} & \beta_{1,4} \\ \beta_{2,0} & \beta_{2,1} & \beta_{2,2} & \beta_{2,3} & \beta_{2,4} \\ \beta_{3,0} & \beta_{3,1} & \beta_{3,2} & \beta_{3,3} & \beta_{3,4} \end{bmatrix}$$

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

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

(i) [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)

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

(iii) [1 pt] What is the vector $\mathbf{a}$ whose elements are made up of the entries a_j in equation (2). Write your answer in terms of α and $\mathbf{x}^{(1)}$.

(iv) [1 pt] What is the vector $\mathbf{z}$ whose elements are made up of the entries z_j in equation (3)? Write your answer in terms of $\mathbf{a}$.

(v) [1 pt] **Select one:** We cannot take the matrix multiplication of our weights β and our vector $\mathbf{z}$ since they are not compatible shapes. Which of the following would allow us to take the matrix multiplication of β and $\mathbf{z}$ such that the entries of the vector $\mathbf{b} = \beta * \mathbf{z}$ are equivalent to the values of b_k in equation (4)?

- ◯ A) Remove the last column of β
- ◯ B) Remove the first row of $\mathbf{z}$
- ◯ C) Append a value of 1 to be the first entry of $\mathbf{z}$
- ◯ D) Append an additional column of 1's to be the first column of β
- ◯ E) Append a row of 1's to be the first row of β
- ◯ F) Take the transpose of β

(vi) [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.

(b) We will now derive the matrix and vector forms for the backpropagation algorithm.

$$\frac{\partial \ell}{\partial \boldsymbol{\alpha}} = \begin{bmatrix} \frac{\partial \ell}{\partial \alpha_{10}} & \frac{\partial \ell}{\partial \alpha_{11}} & \cdots & \frac{\partial \ell}{\partial \alpha_{1M}} \\ \frac{\partial \ell}{\partial \alpha_{20}} & \frac{\partial \ell}{\partial \alpha_{21}} & \cdots & \frac{\partial \ell}{\partial \alpha_{2M}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \ell}{\partial \alpha_{D0}} & \frac{\partial \ell}{\partial \alpha_{D1}} & \cdots & \frac{\partial \ell}{\partial \alpha_{DM}} \end{bmatrix}$$

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 $(\frac{\partial \ell}{\partial \alpha})_{2,1}$ should be equal to $\frac{\partial \ell}{\partial \alpha_{2,1}}$). Recall that ℓ is our loss function defined in equation (6)

(i) [1 pt] The derivative of the softmax function with respect to b_k is as follows:

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

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 $\frac{\partial \ell}{\partial b_k}$ in a smart way such that you do not need this indicator function? Write your solutions in terms of $\hat{y}_k$ and y_k.

$\frac{\partial \ell}{\partial b_k}$**:**

(ii) [1 pt] What are the elements of the row vector $\frac{\partial \ell}{\partial \mathbf{b}}$ evaluated at $\mathbf{y}^{(1)} = [0, 1, 0]^T$?

$\frac{\partial \ell}{\partial \mathbf{b}}$ **at** $\mathbf{y}^{(1)}$**:**

(iii) [1 pt] What is the derivative $\frac{\partial \ell}{\partial \beta}$? Your answer should be in terms of $\frac{\partial \ell}{\partial \mathbf{b}}$ and $\mathbf{z}$.

You should first consider a single entry in this matrix: $\frac{\partial \ell}{\partial \beta_{kj}}$.

$\frac{\partial \ell}{\partial \beta}$**:**

(iv) [1 pt] Explain in one short sentence why must we go back to using the matrix $\boldsymbol{\beta}^*$ (The matrix $\boldsymbol{\beta}$ without the first column of β) when calculating the matrix $\frac{\partial \ell}{\partial \alpha}$?

(v) [1 pt] What is the derivative $\frac{\partial \ell}{\partial \mathbf{z}}$? Your answer should be in terms of $\frac{\partial \ell}{\partial \mathbf{b}}$ and $\boldsymbol{\beta}^*$

$\frac{\partial \ell}{\partial \mathbf{z}}$**:**

(vi) [1 pt] What is the derivative $\frac{\partial \ell}{\partial a_j}$ in terms of $\frac{\partial \ell}{\partial z_j}$ and z_j

$\frac{\partial \ell}{\partial a_j}$**:**

(vii) [1 pt] What is the matrix $\frac{\partial \ell}{\partial \alpha}$? Your answer should be in terms of $\frac{\partial \ell}{\partial \mathbf{a}}$ and $x^{(1)}$.

$\frac{\partial \ell}{\partial \alpha}$**:**

(c) Now you will put these equations to use in an example with numerical values. **You should use the answers you get here to debug your code.**

You are given a training example $\mathbf{x}^{(1)} = [1, 1, 0, 0, 1, 1]^T$ with **label class 2**, so $\mathbf{y}^{(1)} = [0, 1, 0]^T$. We initialize the network weights as:

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

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

We want to also consider the bias term and the weights on the bias terms ($\alpha_{j,0}$ and $\beta_{j,0}$). Lets say they are all initialized to 1. To account for this we can add a column of 1's to the beginning of our initial weight matrices.

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

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

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

$$\mathbf{x}^{(1)} = [1, 1, 1, 0, 0, 1, 1]^T$$

Using the initial weights, run the feed forward of the network over this example (rounding to 4 decimal places during the calculation) and then answer the following questions.

Showing your work in these questions is optional, but it is recommended to help us understand where any misconceptions may occur.

(i) [1 pt] What is a_1?

a_1:	**Work:**

(ii) [1 pt] What is a_2?

a_2:	**Work:**

(iii) [1 pt] What is z_1?

z_1:	**Work:**

(iv) [1 pt] What is z_3?

z_3:	**Work:**

(v) [1 pt] What is b_1?

b_1:	**Work:**

(vi) [1 pt] What is b_2?

b_2:	**Work:**

(vii) [1 pt] What is $\hat{y}_2$?

$\hat{y}_2$:	**Work:**

(viii) [1 pt] Which class would we predict on this example? Your answer should just be an integer $\in \{1, 2, 3\}$.

Class:	**Work:**

(ix) [1 pt] What is the total loss on this example?

Loss:	**Work:**

(d) 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 rounding to 4 decimal places then answer the following questions. Showing your work in these questions is optional, but it is recommended to help us understand where any misconceptions may occur.

(i) [1 pt] What is the value of $\frac{\partial \ell}{\partial \beta_{1,0}}$?

$\frac{\partial \ell}{\partial \beta_{1,0}}$:	**Work:**

(ii) [1 pt] What is the updated value of the weight $\beta_{1,0}$?

$\beta_{1,0}$:	**Work:**

(iii) [1 pt] What is the updated value of the weight $\alpha_{3,4}$?

$\alpha_{3,4}$:	**Work:**

(iv) [1 pt] What is the updated weight of the input layer bias term applied to z_2 (i.e. $\alpha_{2,0}$:)?

$\alpha_{2,0}$**:**

Work:

Q2. [10 pts] Programming

The following questions should be completed after you work through the programming portion of this assignment.

For these questions, **use the large dataset**. Use the following values for the hyperparameters unless otherwise specified:

Parameter	Value
Number of Hidden Units	50
Weight Initialization	RANDOM
Learning Rate	0.01

Please submit computer-generated plots for (d) and (f). Include any code required to produce these results in `additional_code.py` when submitting the programming component. Note: we expect it to take about **5 minutes** to train each of these networks.

(a) Hidden Units

(i) [4 pts] Train a single hidden layer neural network using the hyperparameters mentioned in the table above, except for the number of hidden units which should vary among 5, 20, 50, 100, and 200. Run the optimization for 100 epochs each time.

Plot the average training cross-entropy (sum of the cross-entropy terms over the training dataset divided by the total number of training examples) on the y-axis vs number of hidden units on the x-axis. In the **same figure**, plot the average validation cross-entropy.

Plot:

(ii) [1 pt]

Examine and comment on the the plots of training and validation cross-entropy. What is the effect of changing the number of hidden units?

Answer:

(b) Learning Rate

(i) [4 pts] Train a single hidden layer neural network using the hyperparameters mentioned in the table above, except for the learning rate which should vary among 0.1, 0.01, and 0.001. Run the optimization for 100 epochs each time.

Plot the average training cross-entropy on the y-axis vs the number of epochs on the x-axis for the mentioned learning rates. In the **same figure**, plot the average validation cross-entropy loss. Make a separate figure for each learning rate.

Plot LR 0.1**:**

Plot LR 0.01**:**

Plot LR 0.001**:**

(ii) [1 pt] Examine and comment on the the plots of training and validation cross-entropy. How does adjusting the learning rate affect the convergence of cross-entropy of each dataset?

Answer: