Solutions Template

1 Written Questions

Write your solutions to the HW5 Written Questions in this template. You may either enter your solutions into the .tex file directly, or print out the pdf version and write your solutions by hand in the boxes provided. When you have completed all of the written questions, you should upload your solutions to Gradescope in pdf format.

1.1 Example Feed Forward and Backpropagation

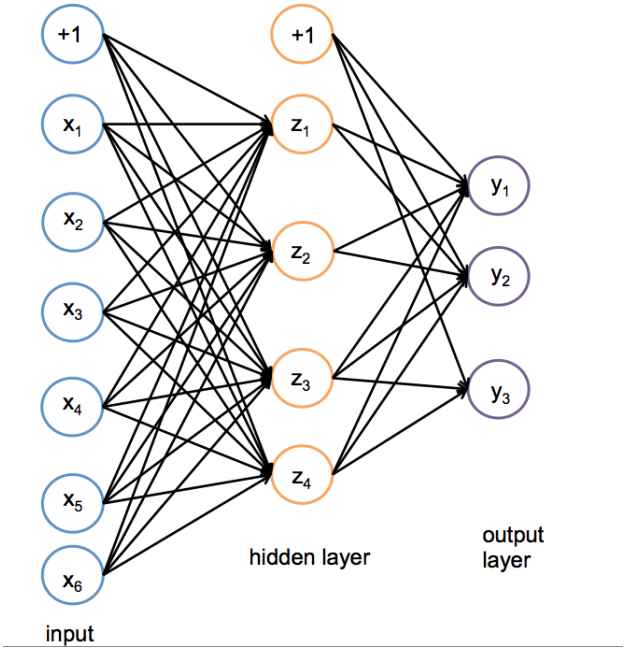


Figure 1.1: A One Hidden Layer Neural Network

Consider the neural network with one hidden layer shown in Figure 1.1. The inputs have 6 features $(x_1, ..., x_6)$, the hidden layer has 4 nodes $(z_1, ..., z_4)$, and the output is a probability distribution (y_1, y_2, y_3) over 3 classes . We also add a bias to the input, x_0, as well as to the hidden layer, z_0 and set them 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.

Equivalently, we define each of the following. The input:

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6) \tag{1.1}$$

Linear combination at first (hidden) layer:

$$a_j = \alpha_0 + \sum_{i=1}^{6} \alpha_{j,i} * x_i, \ \ \forall j \in \{1, \ldots, 6\} \tag{1.2}$$

Activation at first (hidden) layer:

$$z_j = \sigma(a_j) = \frac{1}{1 + \exp(-a_j)}, \ \forall j \in \{1, \ldots, 6\} \tag{1.3}$$

Linear combination at second (output) layer:

$$b_k = \beta_0 + \sum_{j=1}^{4} \beta_{k,j} * z_j, \ \forall k \in \{1, \ldots, 3\} \tag{1.4}$$

Activation at second (output) layer:

$$y_k = \frac{\exp(b_k)}{\sum_{l=1}^{3} \exp(b_l)}, \ \forall k \in \{1, \ldots, 3\} \tag{1.5}$$

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 $z_0 = 1$.

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:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{3} y_i \log(\hat{y}_i) \tag{1.6}$$

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.

1. **[4 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 $x^{(1)} = (1, 1, 0, 0, 1, 1)$ with label class 2, so $y^{(1)} = (0, 1, 0)$. Using the initial weights, run the feed forward of the network over this example (without rounding) and then answer the following questions. In your responses, round to four decimal places (if the answer is an integer you need not include trailing zeros). (Note: the superscript (1) simply indicates that a value corresponds to using training example $x^{(1)}$)

(a) What is $a_1^{(1)}$?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

2

(b) What is $z_1^{(1)}$?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

0.8808

(c) What is $a_3^{(1)}$?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

8

(d) What is $z_3^{(1)}$?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

0.9997

(e) What is $b_2^{(1)}$?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

3.6430

(f) What is $\hat{y}_2^{(1)}$?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

0.2615

(g) Which class would we predict on this example?

$class3$

(h) What is the total loss on this example?

(Please include one number rounded to the fourth decimal place, e.g. 0.1234)

1.3412

2. **[5 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 four decimal places

   (a) What is the updated value of $\beta_{2,1}$?

   (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   1.5103

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

   (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.9151

   (c) What is the updated value of $\alpha_{3,4}$?

   (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   2.0000

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

   (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.9989

   (e) Once we've updated all of our weights if we ran feed forward over the same example again, which class would we predict?

   (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   $class2$

3. **[1 points]** Suppose you are now given a collection of training examples $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$. Explain in English why the cross-entropy loss averaged over these examples is exactly the same quantity as the negative average conditional log-likelihood of the data.

   Negative average conditional log-likelyhood is a multiclass cross-entropy. since y is a one-hot vector, one one element of this vector is 1, the rest are zero. Therefore, summation of $y_i \log(\hat{y}_i)$ is simplified to only $\log(\hat{y}_i)$.

1.2 Empirical Questions [10 points]

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:

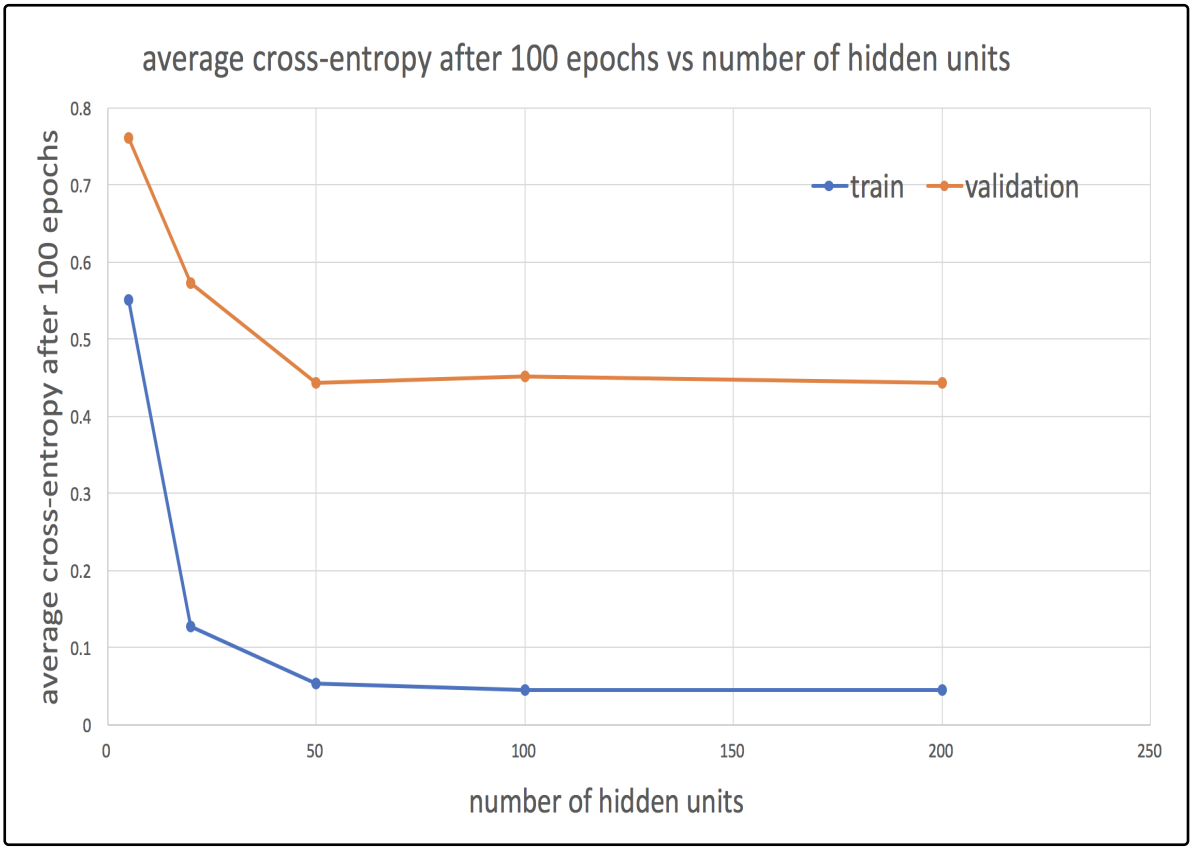
Paramater	Value
Number of Hidden Units	50
Weight Initialization	0.1
Learning Rate	0.01

Table 1.1: Default values of hyperparameters for experiments in Section 1.2.

For the following questions, submit your solutions to Gradescope. Do **not** include any visualization-related code when submitting to Autolab! Note: we expect it to take about **5 minutes** to train each of these networks.

4. **[4 points]** Train a single hidden layer neural network using the hyperparameters mentioned in Table 1.1, 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. On the same figure, plot the average validation cross-entropy.

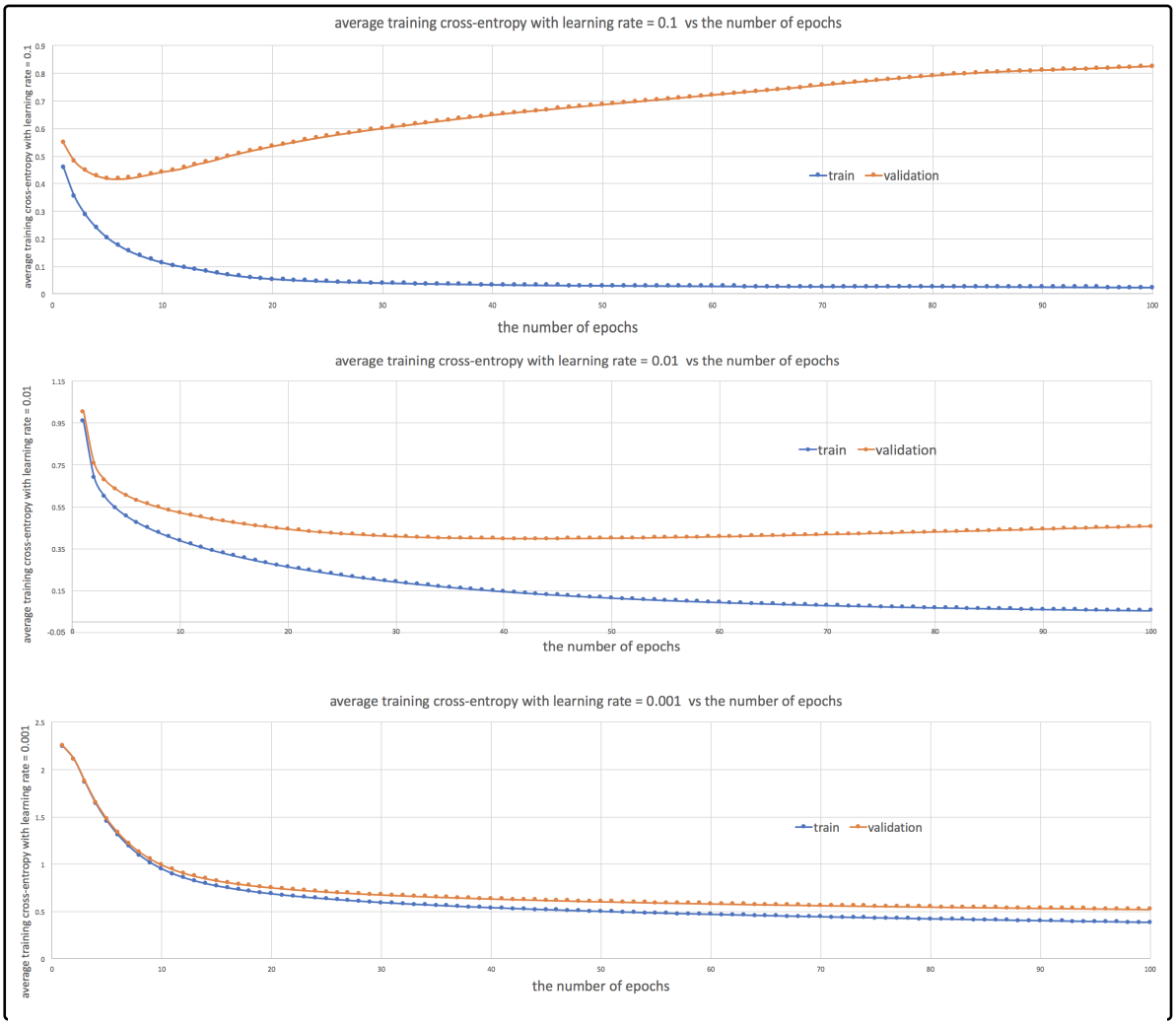


5. **[1 points]** Examine and comment on the the plots of training error and validation error. What effect does changing the number of hidden units have?

training error measured by average cross-entropy is always higher on validation data than that on training data. For hidden units from 0-20, the difference between train/validation error is relatively small, while for hidden units from 20-200, the differences between train/validation errors grows large and become a stable difference.
However, greater number of hidden units give both lower train and validation error than that of smaller number of hidden units.

6. **[4 points]** Train a single hidden layer neural network using the hyperparameters mentioned in Table 1.1, 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. On the same figure, plot the average validation cross-entropy loss. You may make a separate figure for each learning rate.



7. **[1 points]** 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?

As learning rate changes from 0.1 via 0.01, to 0.001, the cross-entropy between training data and validation data changes to converge from diverging as the number of epoches grows up.This indicates a smaller learning rate, in this case 0.001, has a smaller model risk.

8. **Collaboration Questions** After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here. (a) Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details. (b) Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details? (c) Did you find or come across code that implements any part of this assignment ? If so, include full details.

No.