HOMEWORK 5: NEURAL NETWORKS

10-601 Introduction to Machine Learning (Fall 2018)
Carnegie Mellon University
https://piazza.com/cmu/fall2018/10601bd
OUT: Oct 9, 2018[*]
DUE: Oct 20, 2018 11:59 PM
TAs: Aakanksha, Edgar, Sida, Varsha

Summary In this assignment, you will build a handwriting recognition system using a neural network. As a warmup, Section 1 will lead you through an on-paper example of how to implement a neural network. Then, in Section 2, you will implement an end-to-end system that learns to perform handwritten letter classification.

START HERE: Instructions

- **Collaboration Policy**: Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the collaboration policy on the website for more information: http://www.cs.cmu.edu/~mgormley/courses/10601bd-f18/about.html
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/10601bd-f18/about.html
- **Submitting your work:** You will use Gradescope to submit answers to all questions, and Autolab to submit your code. Please follow instructions at the end of this PDF to correctly submit all your code to Autolab.
  - **Gradescope:** For written problems such as derivations, proofs, or plots we will be using Gradescope (https://gradescope.com/). Submissions can be handwritten, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Upon submission, label each question using the template provided. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page.
  - **Autolab:** You will submit your code for programming questions on the homework to Autolab (https://autolab.andrew.cmu.edu/). After uploading your code, our grading

[*] Compiled on Friday 19th October, 2018 at 02:17

scripts will autograde your assignment by running your program on a virtual machine (VM). The software installed on the VM is identical to that on `linux.andrew.cmu.edu`, so you should check that your code runs correctly there. If developing locally, check that the version number of the programming language environment (e.g. Python 2.7/3.5, Octave 3.8.2, OpenJDK 1.8.0, g++ 4.8.5) and versions of permitted libraries (e.g. `numpy` 1.7.1) match those on `linux.andrew.cmu.edu`. (Octave users: Please make sure you do not use any Matlab-specific libraries in your code that might make it fail against our tests.) You have a **total of 10 Autolab submissions**. Use them wisely. In order to not waste Autolab submissions, we recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly first before any Autolab submission.

- **Materials:** Download from Autolab the tar file ("Download handout"). The tar file will contain all the data that you will need in order to complete this assignment.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeXusers, use ■ and ● for shaded boxes and circles, and don't change anything else.

Instructions for Specific Problem Types

For “Select One” questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

~~●~~ Noam Chomsky

For “Select all that apply” questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

□ I don’t know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

~~■~~ I don’t know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-601	10-~~7~~601

1 Written Questions [25 points]

Answer the following questions in the HW5 solutions template provided. Then upload your solutions to Gradescope. You may use LaTeX or print the template and hand-write your answers then scan it in. Failure to use the template may result in a penalty.

Note: For all questions which require numerical answers, round up your final answers to four decimal places. For integers, you may drop trailing zeros.

1.1 Example Feed Forward and Backpropagation [15 points]

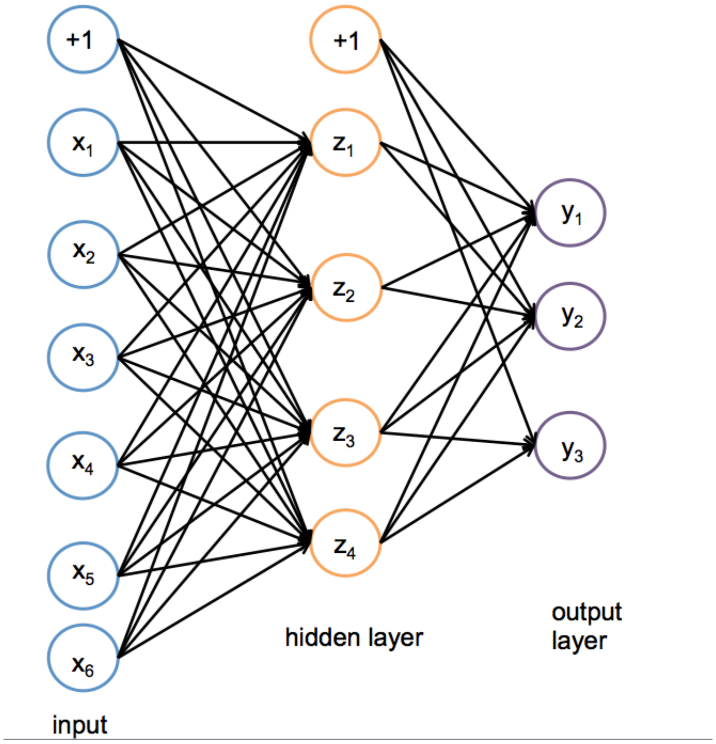


Figure 1.1: A One Hidden Layer Neural Network

Network Overview Consider the neural network with one hidden layer shown in Figure 1.1. 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:

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T \tag{1.1}$$

Linear combination at first (hidden) layer:

$$a_j = \alpha_0 + \sum_{i=1}^{6} \alpha_{j,i} * x_i,\ \forall j \in \{1, \ldots, 4\} \tag{1.2}$$

Activation at first (hidden) layer:

$$z_j = \sigma(a_j) = \frac{1}{1 + \exp(-a_j)},\ \forall j \in \{1, \ldots, 4\} \tag{1.3}$$

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:

$$\hat{y}_k = \frac{\exp(b_k)}{\sum\limits_{l=1}^{3} \exp(b_l)},\ \forall k \in \{1, \ldots, 3\} \tag{1.5}$$

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 $z_0 = 1$.

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:

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

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

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 $\mathbf{x}^{(1)} = [1, 1, 0, 0, 1, 1]^T$ 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 example (without rounding during the calculation) and then answer the following questions.

(a) What is a_1?

$\boxed{2}$

(b) What is z_1?

$\boxed{0.881}$

(c) What is a_3?

$\boxed{8}$

(d) What is z_3?

$\boxed{0.999}$

(e) What is b_2?

$\boxed{3.642}$

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

$\boxed{0.261}$

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

$\boxed{3}$

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

$\boxed{1.341}$

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 $\beta_{2,1}$?

   1.651

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

   0.892

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

   2

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

   0.998

   (e) 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?

   2

3. **[6 points]** Let us now introduce regularization into our neural network. For this question, we will incorporate L2 regularization into our loss function $\ell(\hat{\mathbf{y}}, \mathbf{y})$, with the parameter λ controlling the weight given to the regularization term.

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

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{3} y_i \log(\hat{y}_i) + \lambda||\alpha||^2 + \lambda||\beta||^2$$

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

2.411

   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.

   (c) Report the non-regularized loss for the network on training example $\mathbf{x}^{(1)}$

1.4075

   (d) Report the regularized loss for the network on training example $\mathbf{x}^{(1)}$ (λ = 0.01)

79.698

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

$$\alpha_{j,i} = \alpha_{j,i} - \eta * (\frac{\partial \ell(\hat{\mathbf{y}},\mathbf{y})}{\partial \alpha_{j,i}} + 2\lambda\alpha_{j,i})$$

(f) 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

1.2 Empirical Questions [10 points]

The following questions should be completed after you work through the programming portion of this assignment (Section 2).

For these questions, **use the large dataset**.

Use the following values for the hyperparameters unless otherwise specified:

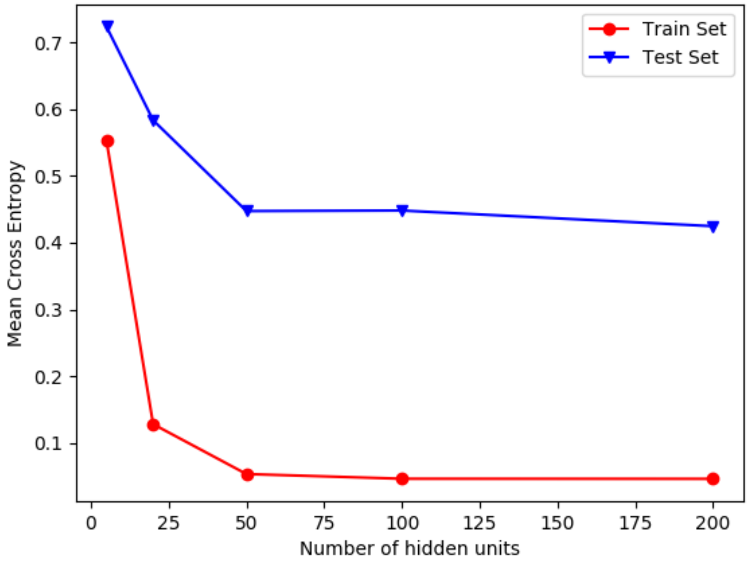
Paramater	**Value**
Number of Hidden Units	50
Weight Initialization	RANDOM
Learning Rate	0.01

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

For the following questions, submit your solutions to Gradescope. Please submit computer-generated plots for Q4 and Q6. 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. In the **same figure**, plot the average test cross-entropy.



5. **[1 points]** Examine and comment on the the plots of training and test cross-entropy. What is the effect of changing the number of hidden units?

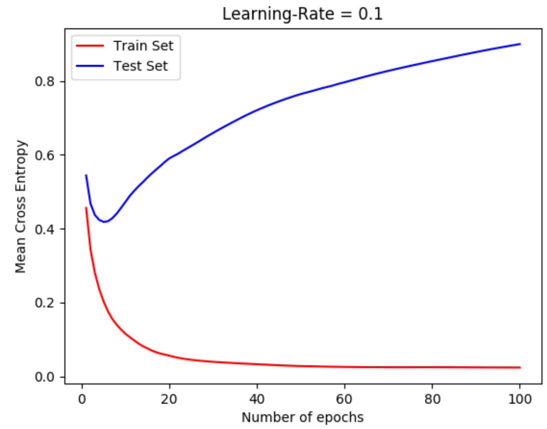
With an increase in the number of hidden units, the mean cross-entropy for both the training and the test set decreases. With 5 hidden units, we were learning a very simple model and hence the mean training cross-entropy is high, but when we take sufficient number of hidden units (20), we see a rapid decrease in the training cross-entropy, signifying that the neural network is learning a sufficiently good representation of the data through weights. We also observe that for the training cross-entropy, the difference with 100 and 200 hidden units is very minimal.

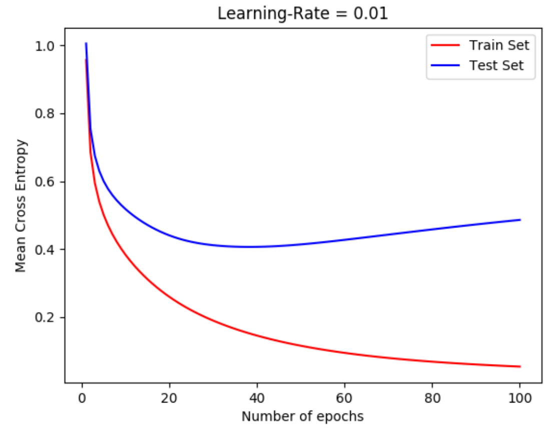
Overall, since the mean training and test cross-entropy stop decreasing after a given number of hidden units (50), it shows that the model is not essentially learning new details with increased complexity and it is okay to use just the sufficient number of hidden layers (here, 50) till we get a good enough error to signify the model representation for the given data.

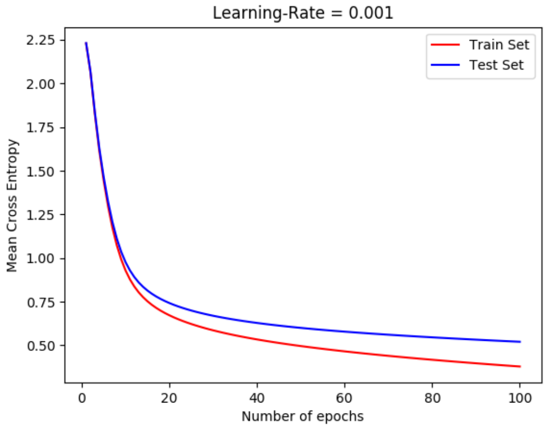
We also see that the test-set mean cross-entropy does not fall below a certain value, siginifying that there is sufficient difference between the training and test set that we can actually consider using the test set for the trained model to check for performance.

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. In the **same figure**, plot the average test 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 test cross-entropy. How does adjusting the learning rate affect the convergence of cross-entropy of each dataset?

Since we are using it to update the weights on statistics of training set, the mean cross-entropy of the training-set always decreases. We also observe that with an decrease in learning rate, the time for convergence increases.

For very large learing rates (0.1), the mean cross-entropy of the test set drastically increases after a certain point, signifying that the model is not learning good representations for generalizing and is overfitting to the training set. The drastic increase in test mean cross-entropy is also signified by a significant drop in train mean cross-entropy.

For very low learning rates (0.001), the mean cross-entropy is decreasing, but very slowly, signifying that we will need a large number of epochs before we get to a sufficiently well learned model for the data.

For learning rate (0.01), we can see that the test set mean cross-entropy decreases consistently and then starts increasing, signifying from here the model is fitting itself to the statistics of the train set. If we consider the case that the test set would have been validation set, we would stop learning at this point and declare the model to be generalizable enough given the distribution of the training and validation (here, test) data.

8. **[0 points]** 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.