HOMEWORK 5: NEURAL NETWORKS

10-301/10-601 Introduction to Machine Learning (Spring 2021)
https://www.cs.cmu.edu/~10601/
DUE: Monday, March 29, 2021 11:59 PM

Summary In this assignment, you will build a handwriting recognition system using a neural network. In the Written component, you will walk through an on-paper example of how to implement a neural network. Then, in the Programming component, you will implement an end-to-end system that learns to perform handwritten letter classification.

START HERE: Instructions

- **Collaboration Policy**: Please read the collaboration policy here: https://www.cs.cmu.edu/~10601
- **Late Submission Policy:** See the late submission policy here: https://www.cs.cmu.edu/~10601
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code. Please follow instructions at the end of this PDF to correctly submit all your code to Gradescope.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please use the provided template. Submissions must be written in LaTeX. Regrade requests can be made, however this gives the staff the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed in the boxes provided. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope (https://gradescope.com). After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). When you are developing, check that the version number of the programming language environment (e.g. Python 3.6.9, OpenJDK 11.0.5, g++ 7.4.0) and versions of permitted libraries (e.g. numpy 1.17.0 and scipy 1.4.1) match those used on Gradescope. You have **a total of 10 Gradescope programming submissions**. Use them wisely. In order to not waste Gradescope submissions, we recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly first before submitting you code to Gradescope.
- **Materials:** The data that you will need in order to complete this assignment is posted along with the writeup and template on Piazza.

Linear Algebra Libraries When implementing machine learning algorithms, it is often convenient to have a linear algebra library at your disposal. In this assignment, Java users may use EJML[a] or ND4J[b] and C++ users Eigen[c]. Details below. (As usual, Python users have NumPy.)

EJML for Java EJML is a pure Java linear algebra package with three interfaces. We strongly recommend using the SimpleMatrix interface. The autograder will use EJML version 0.38. When compiling and running your code, we will add the additional command line argument `-cp "linalg_lib/ejml-v0.38-libs/*:linalg_lib/nd4j-v1.0.0-beta7-libs/*:./"` to ensure that all the EJML jars are on the classpath as well as your code.

ND4J for Java ND4J is a library for multidimensional tensors with an interface akin to Python's NumPy. The autograder will use ND4J version 1.0.0-beta7. When compiling and running your code, we will add the additional command line argument `-cp "linalg_lib/ejml-v0.38-libs/*:linalg_lib/nd4j-v1.0.0-beta7-libs/*:./"` to ensure that all the ND4J jars are on the classpath as well as your code.

Eigen for C++ Eigen is a header-only library, so there is no linking to worry about—just `#include` whatever components you need. The autograder will use Eigen version 3.3.7. The command line arguments above demonstrate how we will call you code. When compiling your code we will include, the argument `-I./linalg_lib` in order to include the `linalg_lib/Eigen` subdirectory, which contains all the headers.

We have included the correct versions of EJML/ND4J/Eigen in the `linalg_lib.zip` posted on the Piazza Resources page for your convenience. It contains the same `linalg_lib/` directory that we will include in the current working directory when running your tests. Do **not** include EJML, ND4J, or Eigen in your homework submission; the autograder will ensure that they are in place.

---

[a] `https://ejml.org`

[b] `https://deeplearning4j.org/docs/latest/nd4j-overview`

[c] `http://eigen.tuxfamily.org/`

Written Questions (44 points)

1 Convolutional Neural Networks

In this problem, consider only the convolutional layer of a standard implementation of a CNN as described in Lecture 12.

1. We are given image X and filter F below.

$X =$

1	0	-2	3	4	1
2	9	5	6	0	-1
0	-3	1	3	4	4
6	5	2	0	6	8
-5	4	-3	1	3	-2
4	1	2	8	9	7

$F =$

-1	-1	-1
-1	8	-1
-1	-1	-1

$Y =$

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

(a) (1 point) Let X be convolved with F using no padding and a stride of 1 to produce an output Y. What is value of j in the output Y?

Answer

8

(b) (1 point) Suppose you had an input feature map of size 6x4 and filter size 2x2, using no padding and a stride of 2, what would be the resulting output size? Write your answer in terms of height × width.

Answer

3×2

2. Parameter sharing is a very important concept for CNN because it drastically reduces the complexity of the learning problem. For the following questions, assume that there is no bias term in our convolutional layer.

(a) (1 point) Which of the following are parameters of a convolutional layer?
Select all that apply:

- ☐ stride size
- ☐ padding size
- ☐ image size
- ☐ filter size
- ■ weights in the filter
- ☐ None of above.

(b) (1 point) Which of the following are hyperparameters of a convolutional layer?
Select all that apply:

- ■ stride size
- ■ padding size
- ☐ image size
- ■ filter size
- ☐ weights in the filter
- ☐ None of above.

(c) (1 point) Suppose for the convolutional layer, we are given black and white images of size 22×22. Using one single 4×4 filter with a stride of 2 and no padding, what is the number of parameters you are learning in this layer?

Answer

16

(d) (1 point) Suppose instead of sharing the same filter for the entire image, you learn a new filter each time you move across the image. Using 4×4 filters with a stride of 2 and no padding, what is the number of parameters you are learning in this layer?

Answer

1600

(e) (1 point) Now suppose you are given a 40×40 colored image, which consists of 3 channels (so your input is a $40 \times 40 \times 3$ tensor), each representing the intensity of one primary color. Without

sharing, using 4×4 filters with a stride of 2 and no padding, what is the number of parameters you are learning in this layer?

Answer

4800

(f) (1 point) Parameter sharing is not usually used for fully-connected layers, but is usually used for convolutional layers. In a sentence, describe a reason why parameter sharing is a good idea for a convolutional layer, besides reduction in problem complexity. Hint: think about applications of CNNs.

Answer

Sharing the parameters facilitate the process of locating a certain image feature everywhere, instead of just a portion, from the original input image.

2 Backpropagation

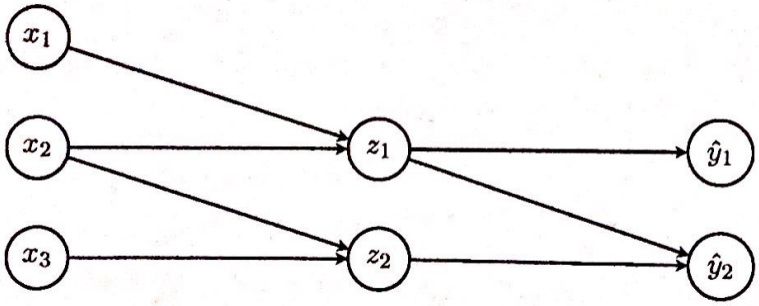


Figure 1: A Directed Acyclic Graph (DAG)

Consider this above Directed Acyclic Graph (DAG). Notice that it looks different than the fully-connected Neural Networks that we have seen before. Recall from lecture that you can perform back propagation on any DAG. We will work through back propagation on this graph.

Let $(\mathbf{x}, \mathbf{y})$ be the training example that is considered, where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ and $\mathbf{y} = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T$. All the other nodes in the graph are defined as:

$$
\begin{aligned}
z_1 &= \text{ReLU}(w_{1,1}x_1 + w_{2,1}x_2) \\
z_2 &= w_{2,2}x_2^2 + w_{3,2}x_3 + b_2 \\
\hat{y}_1 &= \sigma(m_{1,1}z_1^3 + c_1) \\
\hat{y}_2 &= m_{1,2}\sin(z_1) + m_{2,2}\cos(z_2)
\end{aligned}
$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and $\text{ReLU}(x) = \max(0, x)$. Let $\boldsymbol{\theta}$ be the set of all parameters to be learned in this graph. We have that

$$\boldsymbol{\theta} = \{w_{1,1}, w_{2,1}, w_{2,2}, w_{3,2}, m_{1,1}, m_{1,2}, m_{2,2}, b_2, c_1\}$$

For every set of input $\mathbf{x} = (x_1, x_2, x_3)$, we will define objective of the problem as minimizing the loss function

$$J(\boldsymbol{\theta}) = \log\left((y_1 - \hat{y}_1)^2\right) + \log\left((y_2 - \hat{y}_2)^2\right)$$

Assume that you have already gone through the forward pass with inputs $\mathbf{x} = (x_1, x_2, x_3)$ and stored all the relevant values. In the following questions, you will derive the backpropagation algorithm applied to the above DAG.

(a) (1 point) First, we will derive the gradients with repect to the outputs. What are the expressions for $\frac{\partial J}{\partial \hat{y}_1}$? Write your solution in terms of $\hat{y}_1$.

Answer

$-2(y_1 - \hat{y}_1)^{-1}$

(b) Now, we will derive the gradients associated with the last layer, ie nodes y_1, y_2. Note that for the full backpropagation algorithm, you would need to calculate the gradients of the loss function with respect to every parameter $(m_{1,1}, m_{1,2}, m_{2,2}, c_1)$ as well as every input of the layer (z_1, z_2), but we are not asking for all of them in this part. For all of the questions in this part, you should use Chain Rule, and write your solution in terms of values from the forward pass, or **gradients with respect to the outputs of this layer,** $\frac{\partial J}{\partial \hat{y}_1}, \frac{\partial J}{\partial \hat{y}_2}$, because you have already calculated these values. In addition, use the sigmoid function $\sigma(x)$ in your answer instead of its explicit form.

(a) (1 point) What is the expression for $\frac{\partial J}{\partial z_1}$?

Answer

$$\frac{\partial J}{\partial \hat{y}_1} \cdot \sigma(m_{1,1} z_1^3 + c_1)\left[1 - \sigma(m_{1,1} z_1^3 + c_1)\right] \cdot 3 m_{1,1} z_1^2 + \frac{\partial J}{\partial \hat{y}_2} m_{1,2} \cos(z_1)$$

(b) (1 point) What is the expression for $\frac{\partial J}{\partial z_2}$?

Answer

$$= \frac{\partial J}{\partial \hat{y}_2} \cdot m_{2,2} \cdot (-\sin(z_2))$$

(c) (1 point) What is the expression for $\frac{\partial J}{\partial m_{1,1}}$?

Answer

$$\frac{\partial J}{\partial \hat{y}_1} \cdot \sigma(m_{1,1} z_1^3 + c_1)\left[1 - \sigma(m_{1,1} z_1^3 + c_1)\right] \cdot z_1^3$$

(d) (1 point) What is the expression for $\frac{\partial J}{\partial m_{1,2}}$?

Answer

$$\frac{\partial J}{\partial \hat{y}_2} \cdot \sin(z_1)$$

(e) (1 point) What is the expression for $\frac{\partial J}{\partial c_1}$?

Answer

$$\frac{\partial J}{\partial \hat{y}_1} \cdot \sigma(m_{1,1} z_1^3 + c_1)\left[1 - \sigma(m_{1,1} z_1^3 + c_1)\right]$$

(f) (1 point) Lastly, we will derive the gradients associated with the second layer, ie nodes z_1, z_2. Note that for the full backpropagation algorithm, you need to calculate the gradients of the loss function with respect to every parameter (every $w_{i,j}, b_2$). However, we do not need to calculate the gradients with respect to the inputs of this layer (x_1, x_2, x_3), because they are fixed inputs of the model. For all of the questions in this part, you should use Chain Rule, and write your solution in terms of values from the forward pass or **gradients with respect to the outputs of this layer,** $\frac{\partial J}{\partial z_1}, \frac{\partial J}{\partial z_2}$, because you have already calculated these values.

(a) (1 point) What is the expression for $\frac{\partial J}{\partial w_{2,2}}$?

Answer

$\frac{\partial J}{\partial z_2} \cdot x_2^2$

(b) (1 point) What is the expression for $\frac{\partial J}{\partial b_2}$?

Answer

$\frac{\partial J}{\partial z_2}$

(c) (2 points) Recall that $\text{ReLU}(x) = \max(x, 0)$. The ReLU function is not differentiable at $x = 0$, but for backpropagation, we define its derivative as

$$\text{ReLU}'(x) = \begin{cases} 0, x < 0 \\ 1, \text{ otherwise} \end{cases}$$

Now, what is the expression for $\frac{\partial J}{\partial w_{1,1}}$? Explicitly write out the cases.

Answer

$$\frac{\partial J}{\partial w_{1,1}} = \begin{cases} 0, & w_{1,1} x_1 + w_{2,1} x_2 < 0 \\ \frac{\partial J}{\partial z_1} \cdot x_1, & \text{otherwise} \end{cases}$$

3 Backpropagation Through Time

Consider the following one-layer many-to-many Recurrent neural network (RNN),

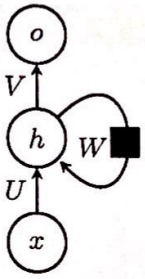


Figure 2: A one hidden layer many-to-many Recurrent Neural Network

Recall from lecture, this is equivalent to

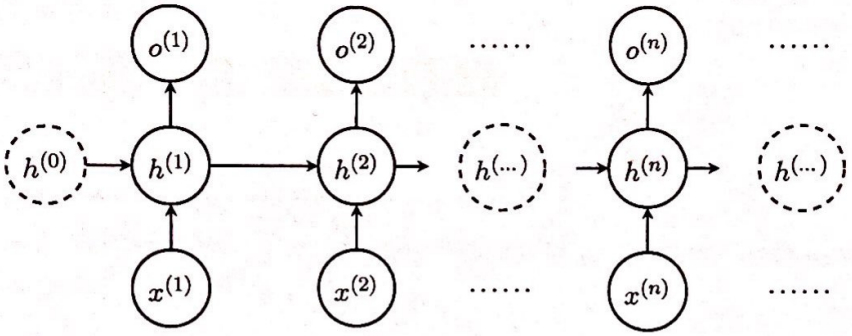


Figure 3: The unfolding of the above RNN in time of the computation involved in its forward computation

Assume $h^{(0)}$ is given. To keep things simple, we consider an RNN with identity activation function and no bias terms. We also assume that $W = I$, the identity matrix. For each timestep $n \geq 1$, the hidden state and output can be expressed as

$$h^{(n)} = f(h^{(n-1)}, x^{(n)}, U) = Ux^{(n)} + h^{(n-1)}$$
$$o^{(n)} = g(h^{(n)}, V) = Vh^{(n)}$$

where $x^{(n)} \in \mathbb{R}^a, h^{(n)} \in \mathbb{R}^b, o^{(n)} \in \mathbb{R}^c, U \in \mathbb{R}^{b \times a}$, and $V \in \mathbb{R}^{c \times b}$. Assume finite time N. The loss can then be expressed as

$$J(\boldsymbol{\theta}) = \sum_{t=1}^{N} l(y^{(t)}, o^{(t)})$$

where $l(y^{(t)}, o^{(t)})$ is the loss at time step t.

For the following questions, express your answer in terms of $x^{(n)}, h^{(n)}, o^{(n)}, U, V$ unless specified otherwise. Some rules of matrix calculus may be helpful: for $J \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n \times m}$,

$$\frac{\partial J}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \\ \dots \\ \frac{\partial J}{\partial x_n} \end{bmatrix} \qquad \frac{\partial J}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial J}{\partial x_{11}} & \frac{\partial J}{\partial x_{12}} & \cdots & \frac{\partial J}{\partial x_{1m}} \\ \frac{\partial J}{\partial x_{21}} & \frac{\partial J}{\partial x_{22}} & \cdots & \frac{\partial J}{\partial x_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial x_{n1}} & \frac{\partial J}{\partial x_{n2}} & \cdots & \frac{\partial J}{\partial x_{nm}} \end{bmatrix}$$

1. Assume $N = 1$. For the following subparts (a)-(d), you may give your answer in terms of $\frac{\partial l}{\partial o^{(1)}}$.

(a) (1 point) What is $\frac{\partial J}{\partial h^{(1)}}$?

Answer

$V^T \cdot \frac{\partial l}{\partial o^{(1)}}$

(b) (1 point) What is $\frac{\partial J}{\partial V}$?

Answer

$\frac{\partial l}{\partial o^{(1)}} \cdot (h^{(1)})^T$

(c) (1 point) What is $\frac{\partial J}{\partial U}$?

Answer

$V^T \cdot \frac{\partial l}{\partial o^{(1)}} \cdot (x^{(1)})^T$

$\frac{\partial J}{\partial h^{(1)}} \cdot \frac{\partial h^{(1)}}{\partial U}$

2. Assume $N = 2$. For the following subparts (a)-(d), you may give your answer in terms of $\frac{\partial l}{\partial o^{(1)}}$ and $\frac{\partial l}{\partial o^{(2)}}$. $h^{(2)} = Ux^{(2)} + h^{(1)}$

(a) (1 point) What is $\frac{\partial J}{\partial h^{(1)}}$?

Answer

$V^T \cdot \frac{dl}{do^{(1)}} + V^T \cdot \frac{dl}{do^{(2)}}$

$\frac{\partial l(y^{(1)}, o^{(1)}) + l(y^{(2)}, o^{(2)})}{\partial h^{(1)}}$

$\frac{\partial l}{\partial o^{(1)}} \frac{\partial o^{(1)}}{\partial h^{(1)}} + \frac{\partial l}{\partial o^{(2)}} \frac{\partial o^{(2)}}{\partial h^{(2)}} \cdot \frac{\partial h^{(2)}}{\partial h^{(1)}}$

(b) (1 point) What is $\frac{\partial J}{\partial V}$?

Answer

$\frac{dl}{do^{(1)}} \cdot (h^{(1)})^T + \frac{dl}{do^{(2)}} \cdot (h^{(2)})^T$

$\frac{\partial l(y^{(1)}, o^{(1)}) + l(y^{(2)}, o^{(2)})}{\partial V}$

$\frac{\partial l}{\partial o^{(1)}} \cdot \frac{\partial o^{(1)}}{\partial V} + \frac{\partial l}{\partial o^{(2)}} \frac{\partial o^{(2)}}{\partial V}$

(c) (1 point) What is $\frac{\partial J}{\partial U}$?

Answer

$V^T \cdot \left[\frac{dl}{do^{(1)}} \cdot (x^{(1)})^T + 2\frac{dl}{do^{(2)}} \cdot (x^{(1)})^T + \frac{dl}{do^{(2)}} (x^{(2)})^T\right]$

$\frac{\partial J}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial U} + \frac{\partial J}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial U}$

$(V^T \frac{dl}{do^{(1)}} + V^T \frac{dl}{do^{(2)}})(x^{(1)})^T + V^T \frac{dl}{do^{(2)}} (x^{(2)})^T + (x^{(1)})^T$

1231 23 12 3 123

3. Now let's generalize the result to an arbitrary time $N > 1$. For the following subparts (a)-(c), you may give your answer in terms of $\frac{\partial l}{\partial o^{(n)}}$, $n = 1, \ldots, N$.

(a) (2 points) What is $\frac{\partial J}{\partial h^{(n)}}$?

Answer

$$\sum_{i=n}^{N} V^T \cdot \frac{dl}{do^{(i)}}$$

(b) (2 points) What is $\frac{\partial J}{\partial V}$?

Answer

$$\sum_{i=1}^{N} \frac{dl}{do^{(i)}} \cdot (h^{(i)})^T$$

(c) (2 points) What is $\frac{\partial J}{\partial U}$?

Answer

$$V^T \cdot \sum_{i=1}^{N} \left(\frac{dl}{do^{(i)}} + \frac{dl}{dx^{(i+1)}} + \cdots \frac{dl}{do^{(n)}}\right) \cdot (x^{(i)} + x^{(2)} + \cdots x^{(i)})^T$$

4 Empirical Questions

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:

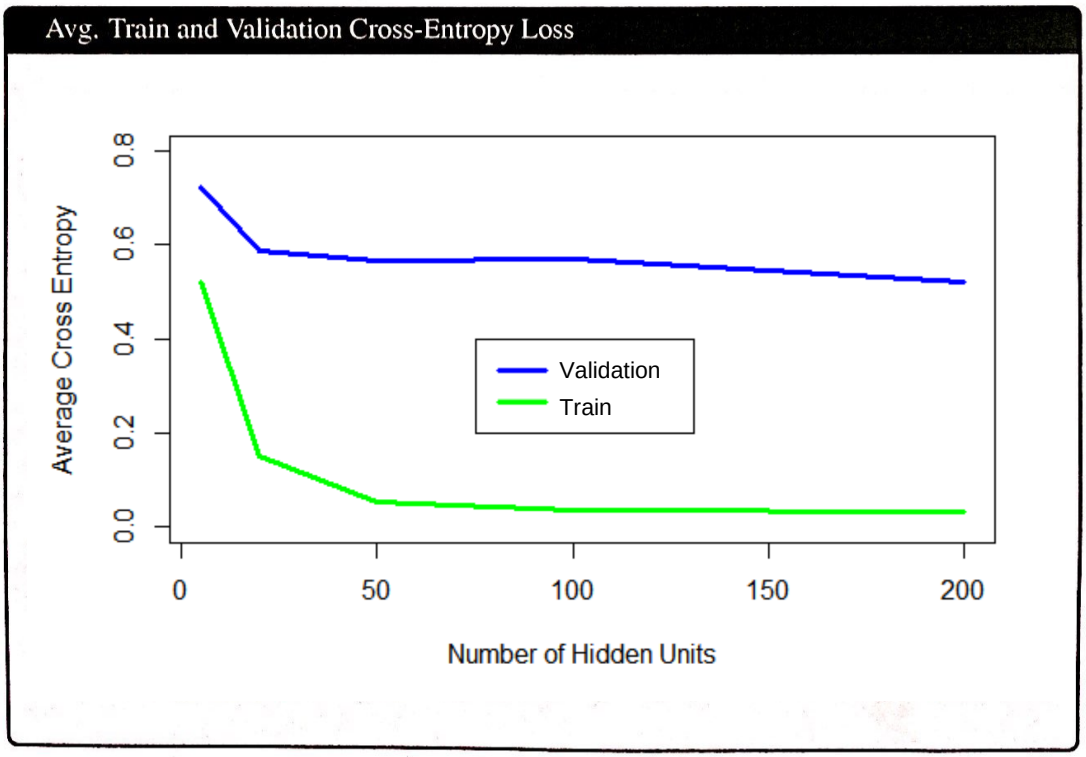
Parameter	Value
Number of Hidden Units	50
Weight Initialization	RANDOM
Learning Rate	0.01

Please submit computer-generated plots for (a)i and (b)i. Note: we expect it to take about **5 minutes** to train each of these networks.

1. Hidden Units

   (a) (2 points) 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.



(b) (2 points) 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

Average cross entropy decreases as the number of hidden units gets larger, and the average cross entropy for validation data is always larger than that of training data.

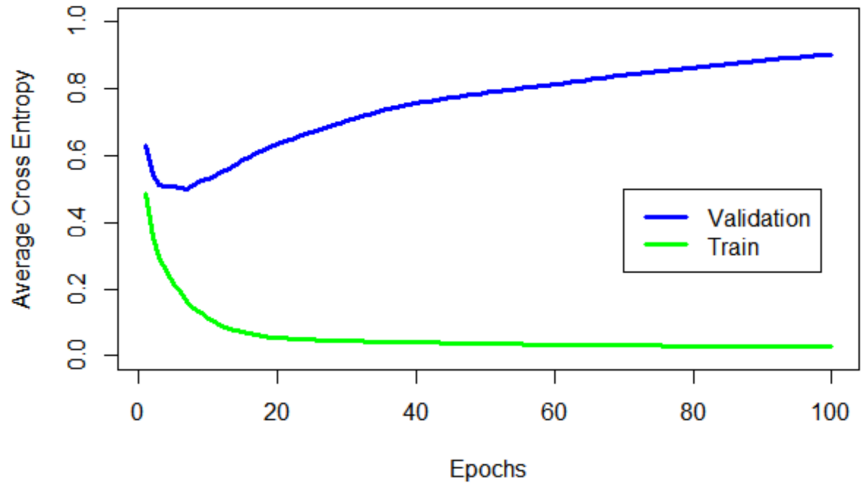
This implies neural network with more hidden units fits the data better.

2. Learning Rate

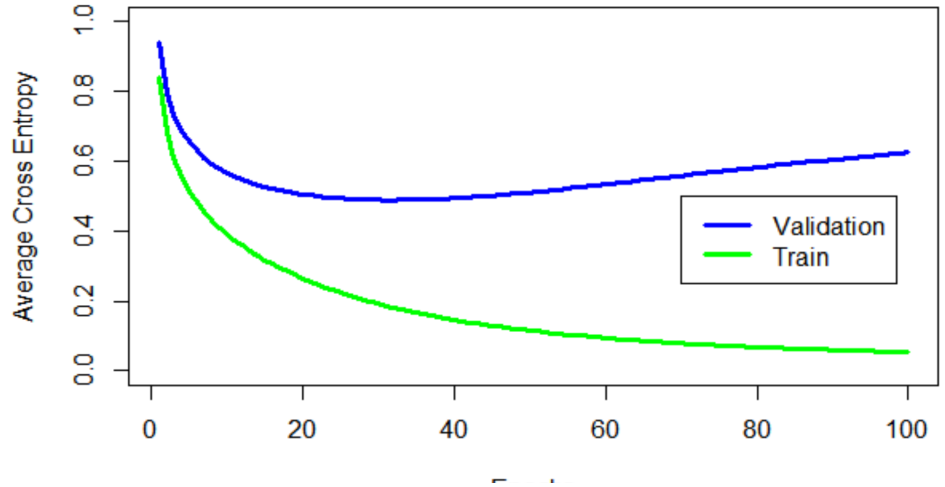
(a) (6 points) Train a single hidden layer neural network using the hyperparameters mentioned in the table above, except for the learning rate which should vary among 0.1, 0.01, and 0.001. Run the optimization for 100 epochs each time.

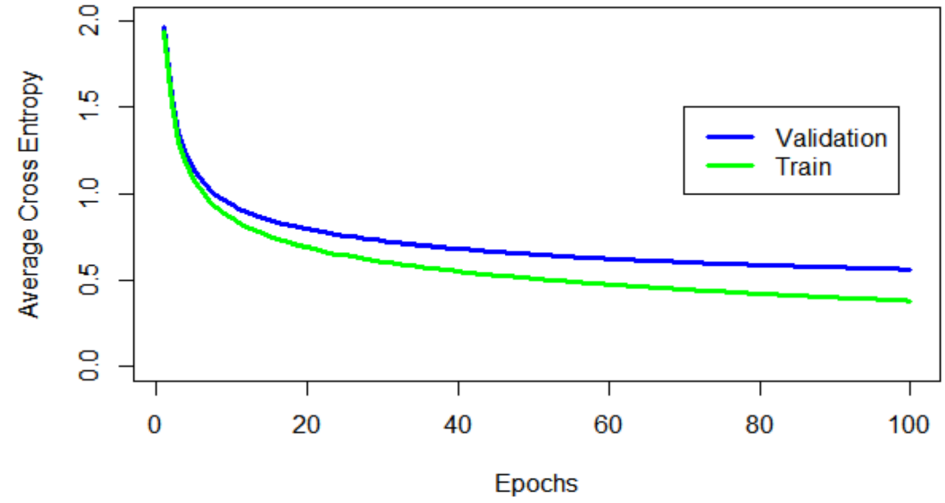
Plot the average training cross-entropy on the y-axis vs the number of epochs on the x-axis for the mentioned learning rates. In the **same figure**, plot the average validation cross-entropy loss. Make a separate figure for each learning rate.

Plot LR 0.1



Plot LR 0.01





(b) (2 points) Examine and comment on the plots of training and validation cross-entropy. How does adjusting the learning rate affect the convergence of cross-entropy on the datasets?

Answer

In all three graphs, cross entropy for training data decreases with epochs, however, the cross entropy for validation data for LR = 0.1 and 0.01 suggests earlier overfitting. We can conclude that, the smaller the learning rate, the closer the mean cross entropies for training and validation datasets get as epochs increase, but it also means that the rate of convergence is lower.

3. Weight Initialization

(a) (2 points) For this exercise, you can work on any data set. Initialize α and β to zero and print them out after the first few updates. For example, you may use the following command to begin:

```
$ python neuralnet.py smallTrain.csv smallValidation.csv \
smallTrain_out.labels smallValidation_out.labels \
smallMetrics_out.txt 1 4 2 0.1
```

Compare the values across rows and columns in α and β. Comment on what you observed. Do you think it is reasonable to use zero initialization? Why or why not?

Answer

In α, some of the rows look quite similar to each other, and all the parameters in α are updated at some point by SGD. In β, there's no obvious pattern although some numbers appears multiple time in the matrix. I think zero initialization is fine, because the softmax layer will be initialized with equal probability for each label.

Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details.
3. Did you find or come across code that implements any part of this assignment ? If so, include full details.

Your Answer

Since I've never worked with numpy before, I frequently checked this page
https://numpy.org/doc/stable/reference/index.html
and the examples on it to implement my numpy functions in this assignment.