CS 615 - Deep Learning

Assignment 4 - Exploring Hyperparameters
Winter 2024

Alec Peterson (ap3842@drexel.edu)

1 Theory

Whenever possible, please leave your answers as fractions so the question of rounding and loss of precision therein does not come up.

1. What would the *one–hot encoding* be for the following set of multi–class labels (5pts)?

$$Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$$

Answer (assuming only the four class labels 0, 1, 2, 3):

$$Y_{encoded} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

2. Given inputs $X = \begin{bmatrix} 1 & 2 & -1 \\ 0 & -4 & -4 \end{bmatrix}$ and the fully connected layer having weights $W = \begin{bmatrix} -1 & 1 & 1 \\ 2 & 2 & 0 \\ 4 & -1 & -3 \end{bmatrix}$, $b = \begin{bmatrix} 1 & 0 & 2 \end{bmatrix}$, what is the output of the following architecture? Show intermediate computations. For simplicity *do not* z-score your inputs. (5pts)

$$Input \rightarrow \text{Fully Connected} \rightarrow \text{Softmax}$$

$OutputofFCL:$
$$\begin{bmatrix} (1)(-1)+(2)(2)+(-1)(4))+1 & (1)(1)+(2)(2)+(-1)(-1)+0 & (1)(1)+(2)(0)+(-1)(-3)+2 \\ (0)(-1)+(-4)(2)+(-4)(4)+1 & (0)(1)+(-4)(2)+(-4)(-1)+0 & (0)(1)+(-4)(0)+(-4)(-3)+2 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 6 & 6 \\ -23 & -4 & 14 \end{bmatrix}$$

$SubsequentOutputofSoftmax:$
$$= \begin{bmatrix} \frac{e^0}{e^0+e^6+e^6} & \frac{e^6}{e^0+e^6+e^6} & \frac{e^6}{e^0+e^6+e^6} \\ \frac{e^{-23}}{e^{-23}+e^{-4}+e^{14}} & \frac{e^{-4}}{e^{-23}+e^{-4}+e^{14}} & \frac{e^{14}}{e^{-23}+e^{-4}+e^{14}} \end{bmatrix}$$

3. Using the same setup as the previous question, what are the gradients to update the fully connected layer's weights (both W and b) if we're using a cross-entropy objective function if we have three (3) total classes as the observations' targets are $Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$? Make sure to show the intermediate gradients being passed backwards to make these computations. (5pts)

4. Given the objective function $J = \frac{1}{4}(x_1w_1)^4 - \frac{4}{3}(x_1w_1)^3 + \frac{3}{2}(x_1w_1)^2$ (*I know you already did this in HW3, but it will be relevant for HW4 as well*):

(a) What is the gradient $\frac{\partial J}{\partial w_1}$ (1pt)?
$\frac{\partial J}{\partial w_1} = \frac{4}{4}(x_1 w_1)^3(x_1) - \frac{12}{3}(x_1 w_1)^2(x_1) + \frac{6}{2}(x_1 w_1)(x_1)$
$= x_1^4 w_1^3 - x_1^3 w_1^2 + 3x_1^2 w_1$

(b) What are the locations of the extrema points for your objective function if $x_1 = 1$? Recall that to find these you set the derivative to zero and solve for, in this case, w_1. (3pts)

$\frac{\partial J}{\partial w_1} = 0 = (1)w_1^3 - 4(1)w_1^2 + 3(1)w_1$
$= w_1^3 - 4w_1^2 + 3w_1 = w_1(w_1^2 - 4w_1 + 3)$
$\implies 0 = w_1(w_1 - 1)(w_1 - 3)$
$\implies w_1 = 0, w_1 = 1, w_1 = 3$

(c) What does J evaluate to at each of your extrema points, again when $x_1 = 1$ (1pts)?

i. $x_1 = 1, w_1 = 0$:
$J = \frac{1}{4}(1 * 0)^4 - \frac{4}{3}(1 * 0)^3 + \frac{3}{2}(1 * 0)^2$
$= 0$

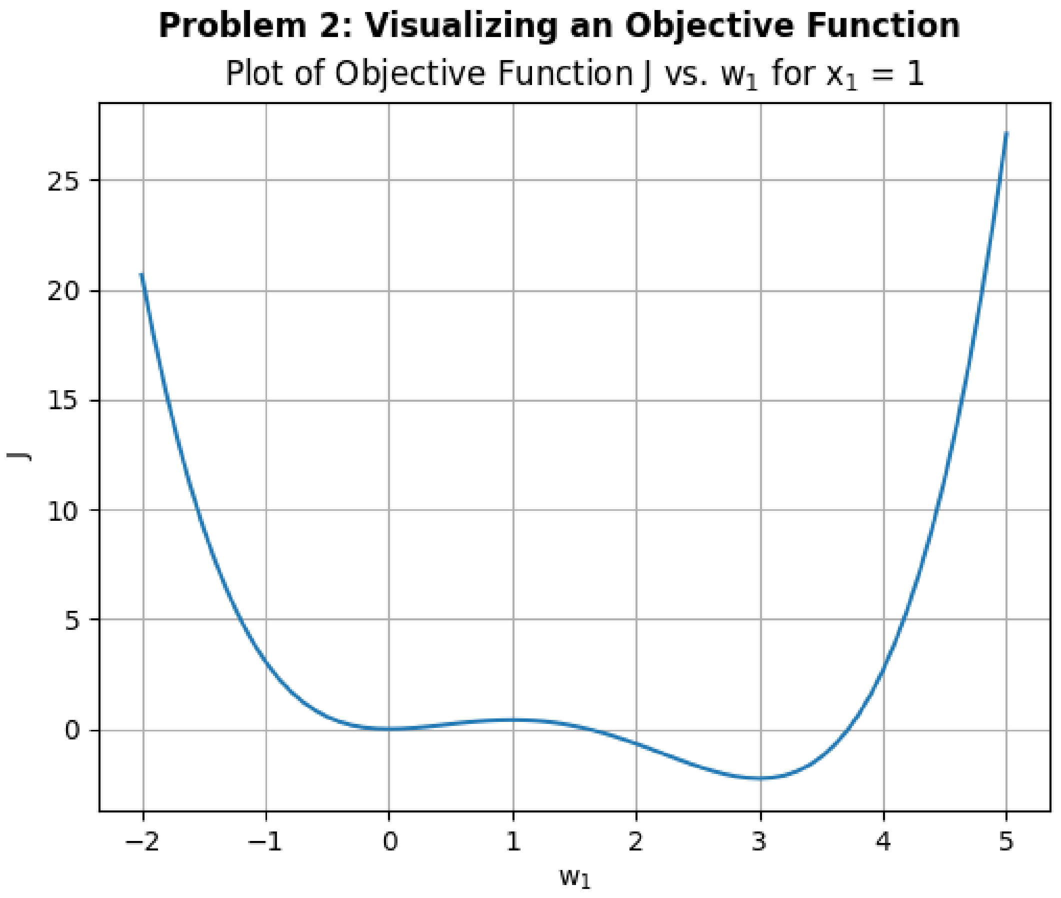
ii. $x_1 = 1, w_1 = 1$:
$J = \frac{1}{4}(1 * 1)^4 - \frac{4}{3}(1 * 1)^3 + \frac{3}{2}(1 * 1)^2$
$= \frac{5}{12}$

iii. $x_1 = 1, w_1 = 3$:
$J = \frac{1}{4}(1 * 3)^4 - \frac{4}{3}(1 * 3)^3 + \frac{3}{2}(1 * 3)^2$
$= -\frac{9}{4}$

2 Visualizing an Objection Function

For the next few parts we'll use the objective function $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$ from the theory section. First let's get a look at this objective function. Using $x_1 = 1$, plot w_1 vs J, varying w_1 from -2 to +5 in increments of 0.1. You will put this figure in your report.

See submitted code and below figure.



3 Exploring Model Initialization Effects

Let's explore the effects of choosing different initializations for our parameter(s). In the theory part you derived the partial of $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$ with respect to the parameter w_1. Now you will run gradient descent on this for four different initial values of w_1 to see the effect of weight initialization and local solutions.

Perform gradient descent as follows:

- Run through 100 epochs.
- Use a learning rate of $\eta = 0.1$.
- Evaluate J at each epoch so we can see how/if it converges.

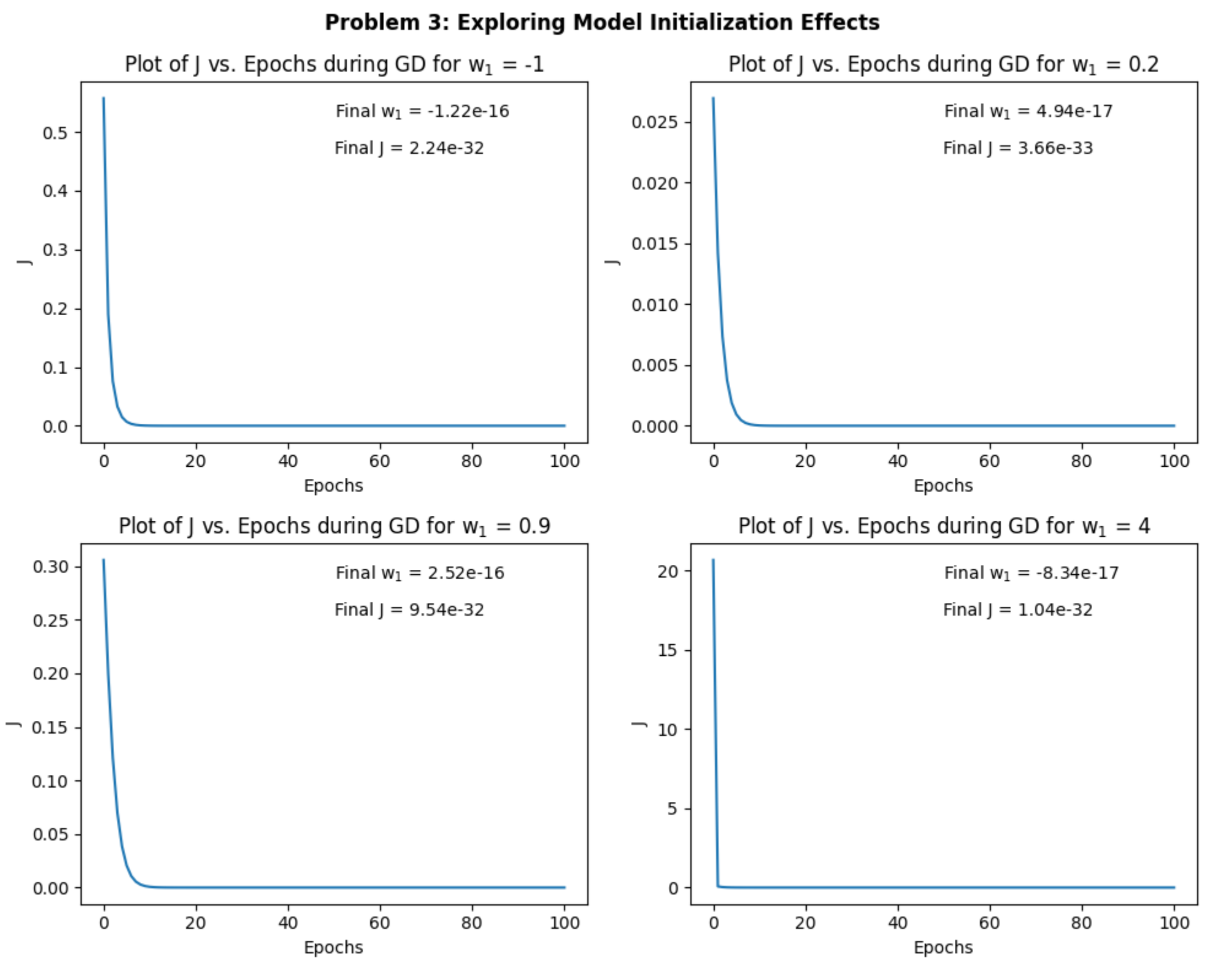
- Assume our only data point is $x = 1$

Do this for initialization choices:

- $w_1 = -1$.
- $w_1 = 0.2$.
- $w_1 = 0.9$.
- $w_1 = 4$.

In your report provide the four plots of epoch vs. J, superimposing on your plots the final value of w_1 and J once 100 epochs has been reached. In addition, based on your visualization of the objective function in Section 2, describe why you think w_1 converged to its final place in each case.

- **See submitted code and below figure.**
- **For $w_1 = -1$, magnitude of slope and step size led to reaching closest local minima at $w_{1,final} \approx 0$**
- **For $w_1 = 0.2$ close to 0, magnitude of slope and step size led to reaching closest local minima at $w_{1,final} \approx 0$**
- **Similarly, for $w_1 = 0.9$, magnitude of slope and step size led to reaching closest local minima at $w_{1,final} \approx 0$**
- **While $w_1 = 4$ started closest to global minima at $w_1 = 3$, it seems like the magnitude of the gradient caused the updated weight to overshoot. The update values for w_1 were : $[-2.0, -.1999, -0.135, ...]$ and step closer and closer to 0.**


Problem 3: Exploring Model Initialization Effects
Plot of J vs. Epochs during GD for w_1 = -1
Final w_1 = -1.22e-16
Final J = 2.24e-32
Plot of J vs. Epochs during GD for w_1 = 0.2
Final w_1 = 4.94e-17
Final J = 3.66e-33
Plot of J vs. Epochs during GD for w_1 = 0.9
Final w_1 = 2.52e-16
Final J = 9.54e-32
Plot of J vs. Epochs during GD for w_1 = 4
Final w_1 = -8.34e-17
Final J = 1.04e-32
J
Epochs

4 Explore Learning Rate Effects

Next we're going to look at how your choice of learning rate can affect things. We'll use the same objective function as the previous sections, namely $J = \frac{1}{4}(x_1w_1)^4 - \frac{4}{3}(x_1w_1)^3 + \frac{3}{2}(x_1w_1)^2$.

For each experiment initialize $w_1 = 0.2$ and use $x = 1$ as your only data point and once again run each experiment for 100 epochs.

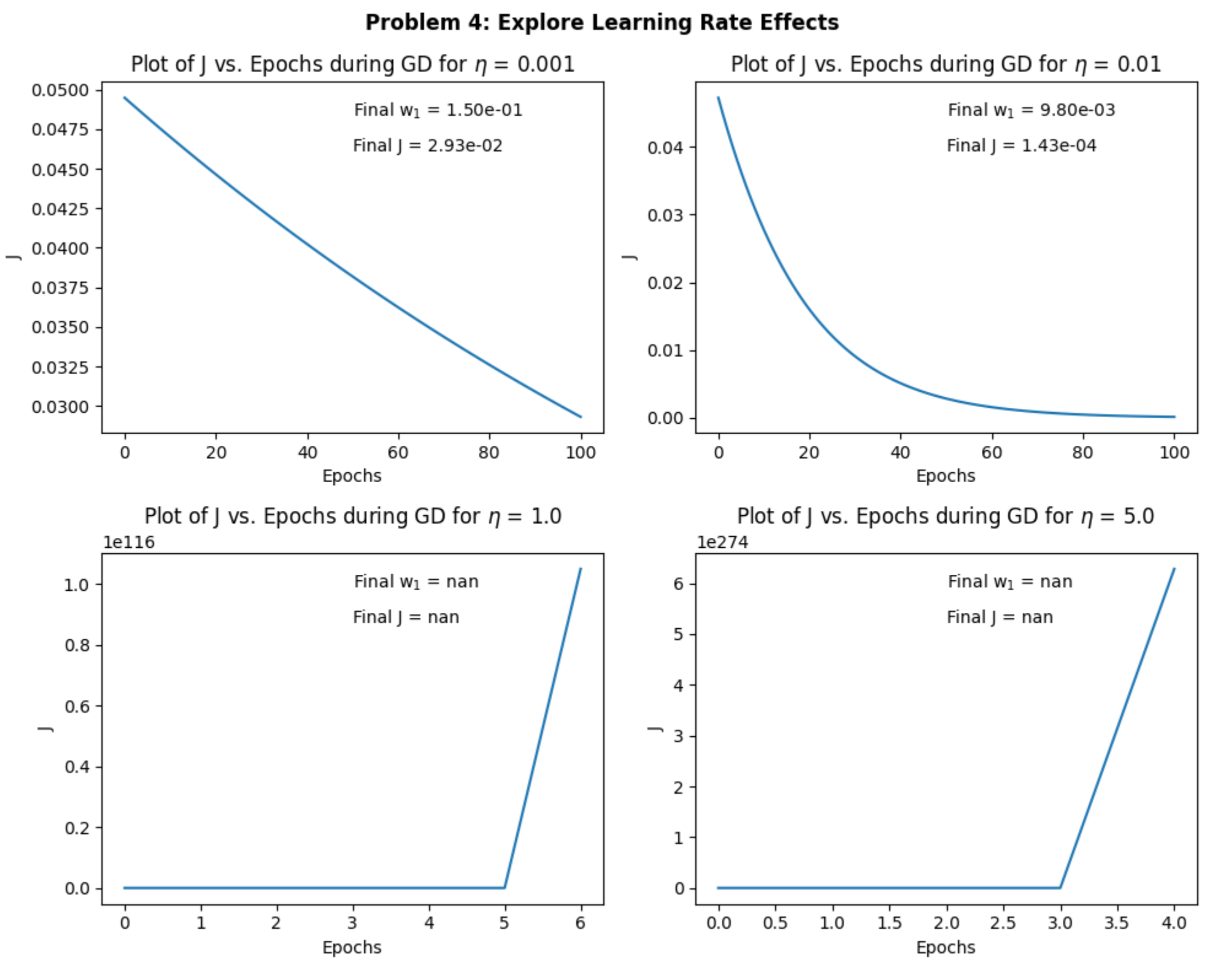
The learning rates for the experiments are:

- $\eta = 0.001$
- $\eta = 0.01$
- $\eta = 1.0$
- $\eta = 5.0$

And once again, create plots of *epoch* vs J for each experiment and superimpose the final values of w_1 and J.

NOTE: Due to the potential of overflow, you likely will want to have the evaluation of your J function in a try/except block where you break out of the gradient decent loop if an exception happens.

See submitted code and below figure.
For learning rates of 1.0 and 5.0, resulted in overflow. Final values did not converge, though J approaches infinity


Problem 4: Explore Learning Rate Effects
Plot of J vs. Epochs during GD for η = 0.001
Final w_1 = 1.50e-01
Final J = 2.93e-02
Plot of J vs. Epochs during GD for η = 0.01
Final w_1 = 9.80e-03
Final J = 1.43e-04
Plot of J vs. Epochs during GD for η = 1.0
1e116
Final w_1 = nan
Final J = nan
Plot of J vs. Epochs during GD for η = 5.0
1e274
Final w_1 = nan
Final J = nan
J
Epochs

5 Adaptive Learning Rate

Finally let's look at using an adaptive learning rate, á la the Adam algorithm.

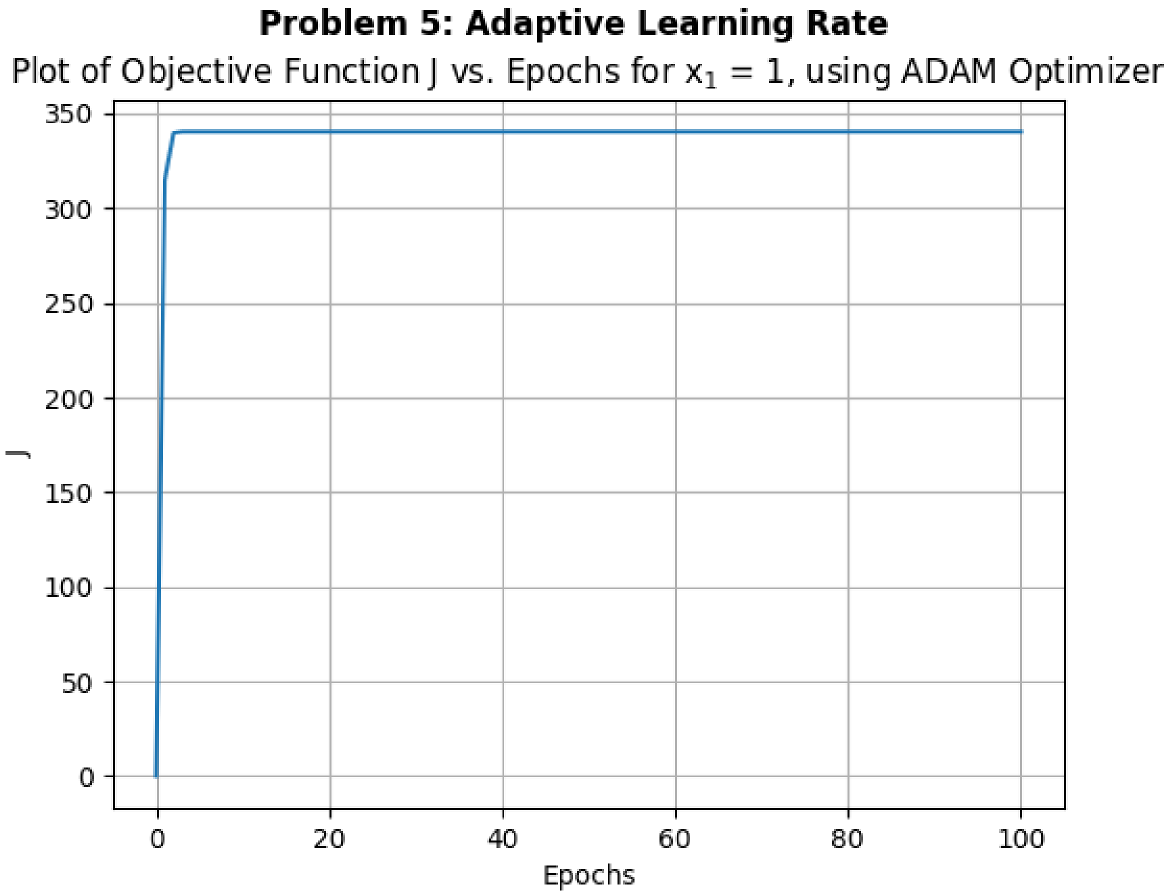
For this part of your homework assignment we'll once again look to learn the w_1 that minimizes $J = \frac{1}{4}(x_1w_1)^4 - \frac{4}{3}(x_1w_1)^3 + \frac{3}{2}(x_1w_1)^2$ given the data point $x = 1$. Run gradient descent *with ADAM* adaptive learning on this objective function for 100 epochs and produce a graph of epoch vs J. Ultimately, you are implementing ADAM from scratch here.

Your hyperparameter initializations are:

- $w_1 = 0.2$
- $\eta = 5$
- $\rho_1 = 0.9$
- $\rho_2 = 0.999$
- $\delta = 10^{-8}$

In your report provide a plot of epoch vs J.

See submitted code and below figure.



6 Multi–Class Classification

Finally, in preparation for our next assignment, let's do multi–class classification. For this we'll use the architecture:

$$\textit{Input} \rightarrow \text{Fully Connected} \rightarrow \text{Softmax} \rightarrow \text{Output w/ Cross–Entropy Objective Function}$$

Download the MNIST dataset from BBlearn and read in the training data. Train your system using the training data, keeping track of the value of your objective function with regards to the training set as you go. In addition, we'll compute the cross entropy loss for the validation as well, the watch for overfitting.

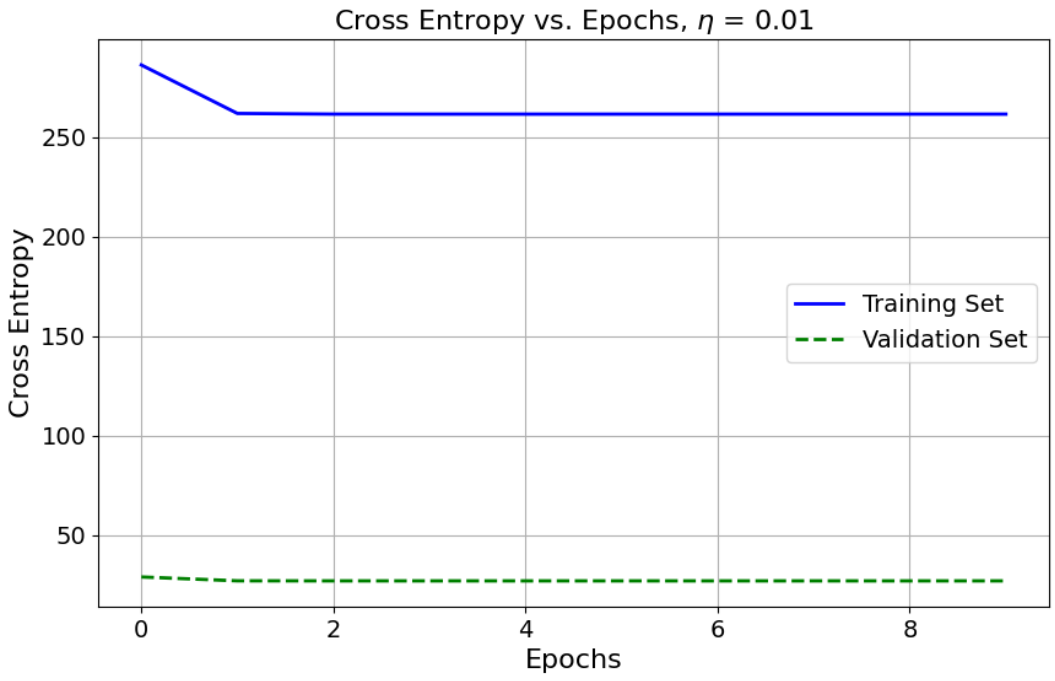
Here's some additional implementation details/specifications:

Implementation Details

- Use *Xavier Initialzation* to initialize your weights and biases.
- Use *ADAM* learning.
- Run your iterations until near-convergence appears (things are mostly flattening out).
- You can decide on your own about things like hyperparameters, batch sizes, z-scoring, etc.. Just report those design decisions in your report and state *why* you made them.

In your final report provide:

- A graph of epoch vs. J for the training data and the validation data. Both plots should be on the same graph with legends indicating which is which.
  See submitted code and below figure.

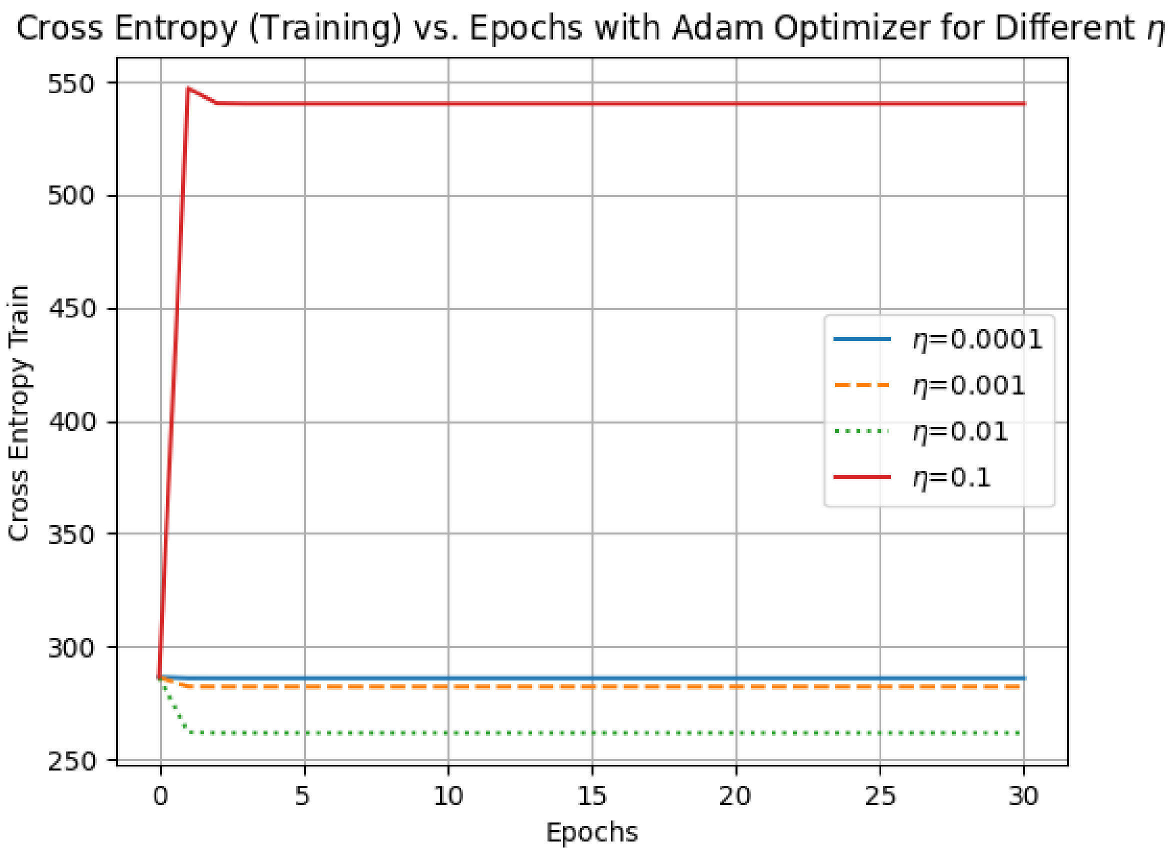


- Your final training and validation *accuracy*. Make sure predict the enumarated class using the *argmax* of the output as well as the original target enumerated classes.

  - $TrainingAccuracy = 0.165$
  - $ValidationAccuracy = 0.160$

- Your hyperparameter design decisions and why you made them.

  - **Learning rate of 0.01 was selected based on lowest converged J vs. epochs relative to other learning rates. See figure below.**
  - **Limited number of epochs required due to quick convergence.**
  - **Z-scoring was used on input values**
  - **Entire dataset was used to calculate gradients for faster convergence. No issues with memory or calculation speed for architecture and dataset size.**


Cross Entropy (Training) vs. Epochs with Adam Optimizer for Different η
Cross Entropy Train
Epochs
η=0.0001
η=0.001
η=0.01
η=0.1