CS 615 - Deep Learning

Assignment 4 - Exploring Hyperparameters
Winter 2023

Introduction

In this assignment we will explore the effect of different hyperparameter choices and apply a multi–class classifier to a dataset.

Programming Language/Environment

As per the syllabus, we are working in Python 3.x and you must constrain yourself to using numpy, matplotlib, pillow and opencv–python add–on libraries.

Allowable Libraries/Functions

In addition, you **cannot** use any ML functions to do the training or evaluation for you. Using basic statistical and linear algebra functions like *mean*, *std*, *cov* etc.. is fine, but using ones like *train*, *confusion*, etc.. is not. Using any ML–related functions, may result in a **zero** for the programming component. In general, use the "spirit of the assignment" (where we're implementing things from scratch) as your guide, but if you want clarification on if can use a particular function, DM the professor on Discord.

Grading

Part 1 (Theory)	10pts
Part 2 (Visualizing an Objective Function)	10pts
Part 3 (Exploring Model Initialization Effects)	20pts
Part 4 (Exploring Learning Rate Effects)	20pts
Part 5 (Adaptive Learning Rate)	20pts
Part 6 (Multi–class classification)	20pts

Table 1: Grading Rubric

Datasets

MNIST Database The MNIST Database is a dataset of hand–written digits from 0 to 9. It contains 60,000 training samples, and 10,000 validation/testing samples, each of which is a 28×28 image.
You have been provided a *CSV* file with the *training data* (we'll worry about the validation data later). This file is arranged so that each row pertains to an observation, and in each row, the first column is the *target class* $\in \{0, 9\}$. The remaining 784 columns are the *features* of that observation, in this case, the pixel values.

For more information about the dataset, you can visit: http://yann.lecun.com/exdb/mnist/

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)?

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

Ans. For this problem we will have to create one hot encoding for the unique digits, for this problem the unique digits are 0, 1, 2, and 3. Hence for these the one hot encoding will be

$$\begin{aligned} 0 &: \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \\ 1 &: \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\ 2 &: \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \\ 3 &: \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

hence for the given array the one hot encoded vector will be:

$$\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

2. Given the objective function $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$ (*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)?
   Ans. $\frac{\partial J}{\partial w_1} = \frac{\partial}{\partial w_1}\left(\frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2\right)$
   Using the power rule, we get $\frac{\partial J}{\partial w_1} = (x_1^4 w_1^3 - 4x_1^3 w_1^2 + 3x_1^2\ w_1)$

   (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)
   Ans. if $x_1 = 1$, the derivative of J with respect to w_1 will be computed against 0,
   $=> w_1(w_1^2 - 4.w_1 + 3) = 0$
   $=> w_1(w_1^2 - 3.w_1 - w_1 + 3) = 0$
   $w_1.(w_1 - 3).(w_1 - 1) = 0$
   So, the value of w_1 which are the point of extrema is
   $w_1 = 0, w_1 = 3, w_1 = 1$

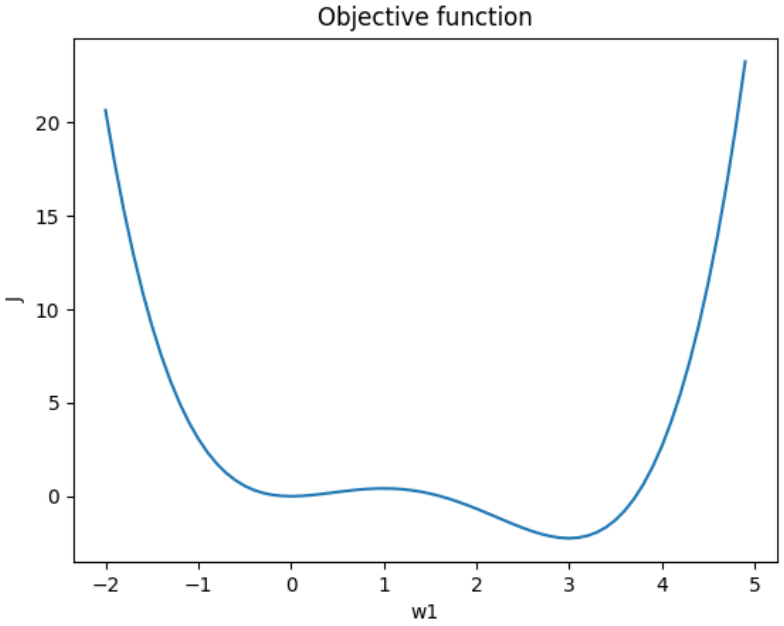
(c) What does J evaluate to at each of your extrema points, again when $x_1 = 1$ (1pts)?

For the value of $w_1 = 0$, $J = 0$

For the value of $w_1 = 1$ $J = 5/12$

For the value of $w_1 = 3$ $J = -9/4$

2 Visualizing an Objection Function



3 Exploring Model Initialization Effects

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:

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

Answer.

Using the analogy from the first theoretical question, we find points of local and global minima with loss values at w1 = 0, J = 0. We can say that the point closest to 0 in our options is -0.2, which gives us pretty good gradient descent results (loss vs epoch). Aside from that, at w1 =3, we got J = -9/4, which is an even lower J value. We can also see some steep results from the options given at w1 = 3.

As a result of the preceding observation, we can conclude that w1 that are closer to the local and global minima points produce better descents and thus accuracy.

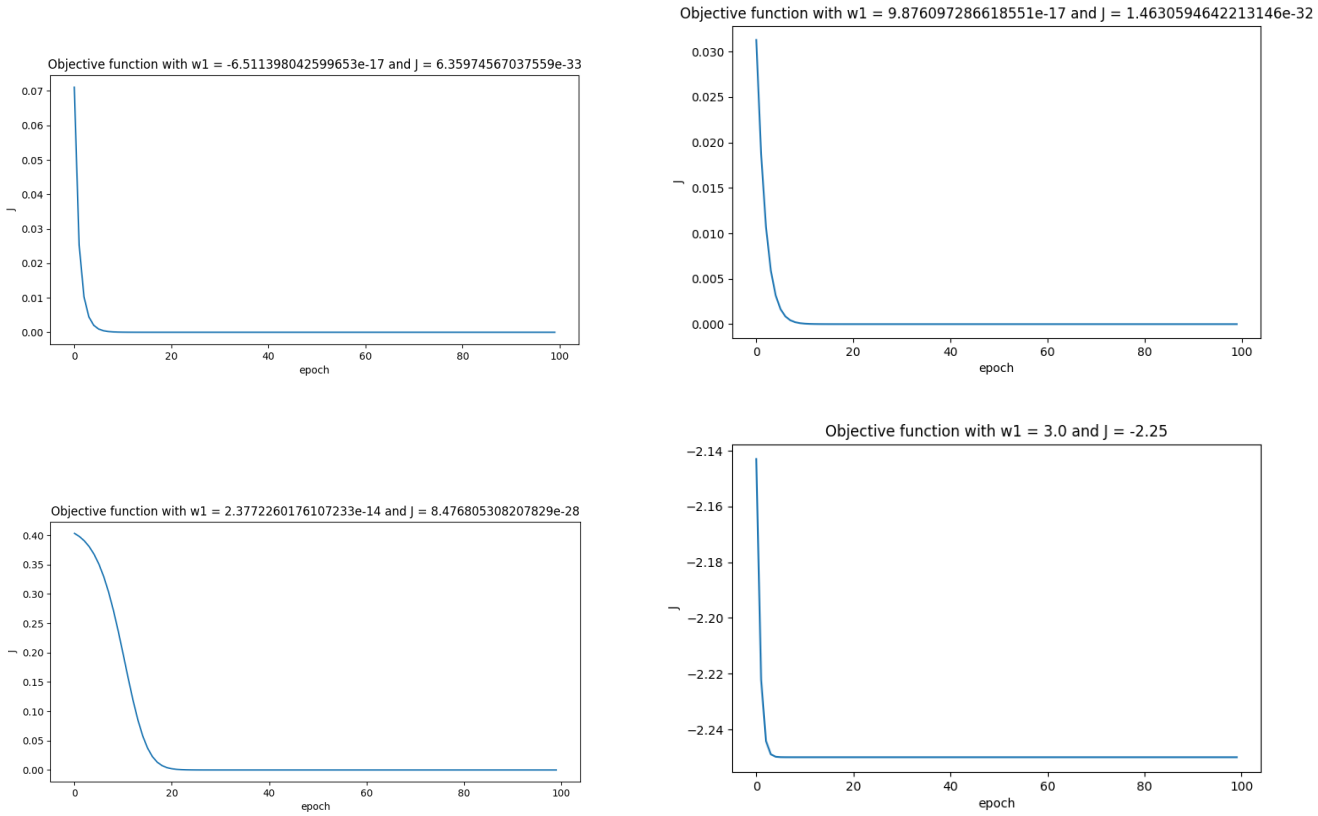


Figure 1: fig 1. is for w1 = -1, fig 2 is for w1 = 0.2, fig 3 is for w1 = 0.9 and fig 4 is for w1 = 4

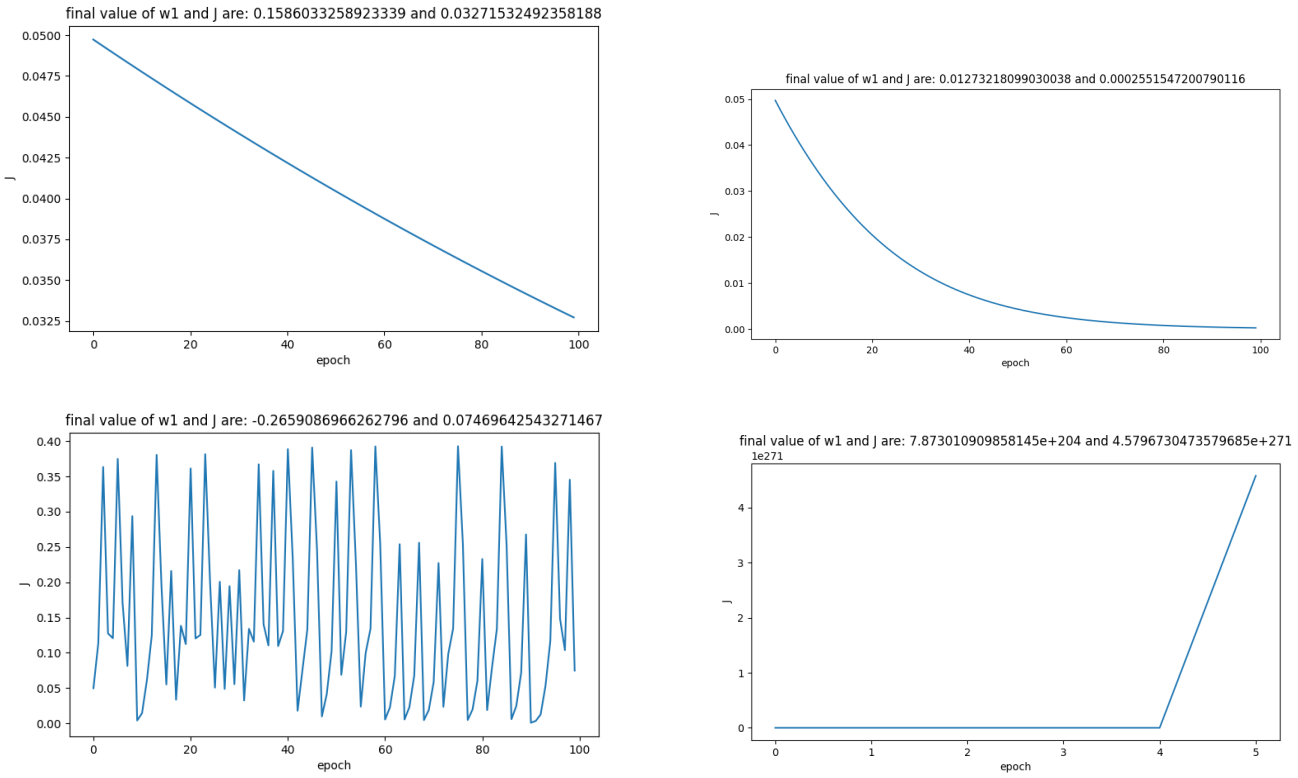


Figure 2: fig 1. is for eta = 0.001, fig 2 is for eta = 0.01, fig 3 is for eta = 1 and fig 4 is for eta = 5

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 $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$.

The learning rates for the experiments are:

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

While attempting to converge the loss function for various values of Loss, answer the following questions. Lower eta values, such as 0.001, cause the function to converge slowly, whereas eta values of 0.01 cause some good steady convergence. However, for larger eta values, the function was unable to converge properly, so I diagnosed an exploding gradient problem over here.

As a result, it is preferable to use a lower learning rate, such as eta = 0.01.

5 Adaptive Learning Rate

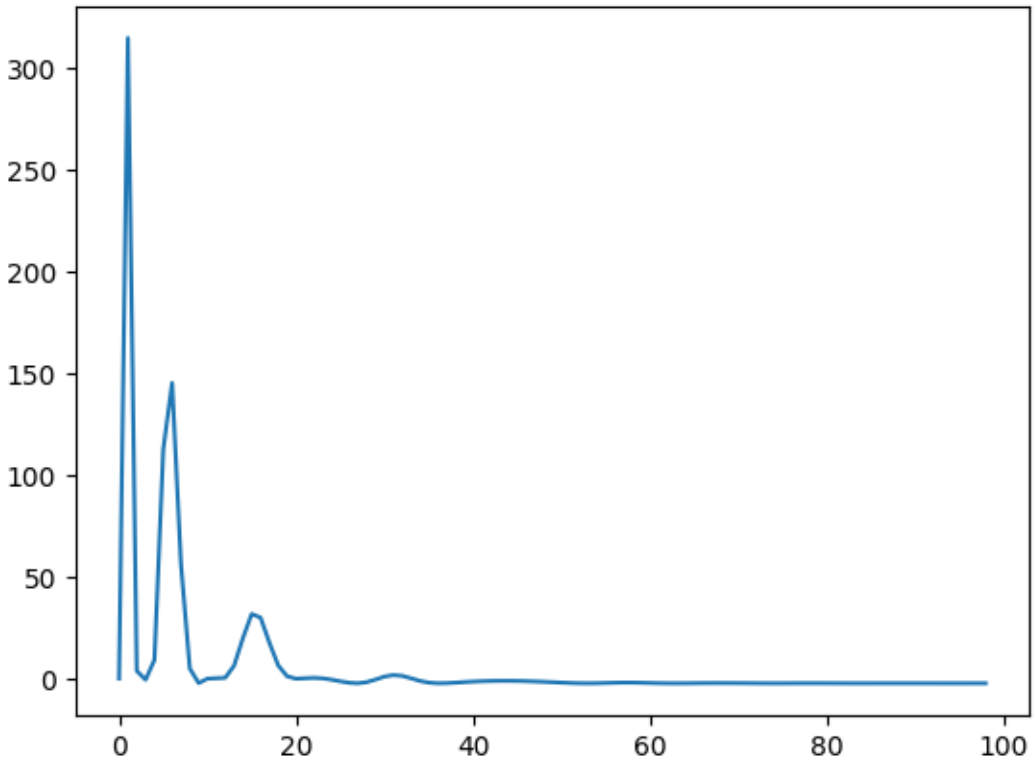
Finally let's look at using an adaptive learning rate, á la the Adam algorithm.

For this part of your homework assignment we'll once again look to learn the w_1 that minimizes $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$ 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.



6 Multi–Class Classification

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

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

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.

Here's some additional implementation details/specifications:

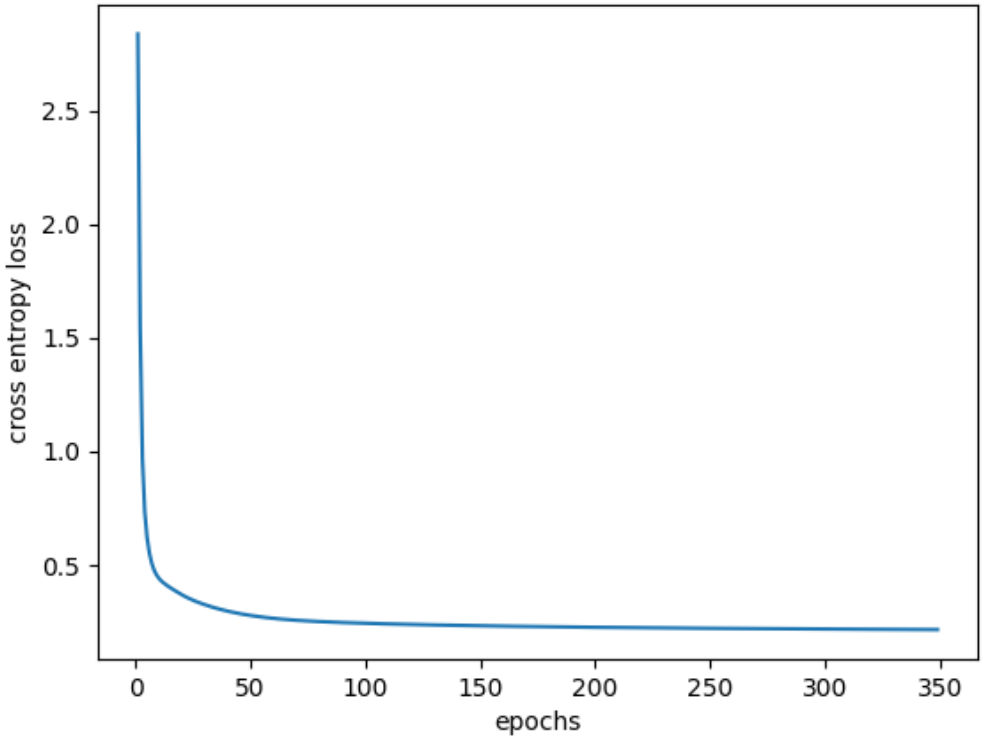
Implementation Details

- Use *Xavier Initialzation* to initialize your weights and biases.
- Use *ADAM* learning.
- 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.

For the following problem I used z-scoring to standardize the data, although It would have been okay even if I did not use zscoring for the data. Apart from this I did not use batch size as I was able to solve the problem quickly without using Stochastic mini-batch gradient descent. For the value of learning rate, I chose learning rate to be 0.01 or 1e-2 because it gave optimum results and fast convergence. Also I used ADAM optimizer for optimization of algorithm with standard parameters like, $rho1 = 0.9, rho2 = 0.999$ and $eta = 1e-8$.

For the calculation of gradient in the Softmax function I used the formula $diag(g(z) - g(z)^T.g(z))$ whereas to compute the backwards for the softmax I used Einstein summation which is a way to use tensor multiplication in an optimum way here I used "$ijk, ik- > ij$" which specifies that we want to contract over the last index of the first array grand and the second index of the second array gradIn, and produce a new array with the remaining indices. The remaining index being ixj or with the notations used in the class NxK matrix.
In your final report provide:

- A graph of epoch vs. J for the training data.
- Your final training *accuracy*.
- Your hyperparameter design decisions and why you made them.



Final Training accuracy turns out to be 94.03 percent when we run it for 350 epochs.

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1:
   (a) Your solutions to the theory question
2. Part 2:
   (a) Your plot.
3. Part 3:
   (a) Your four plots of epoch vs. J with the terminal values of x and J superimposed on each.
   (b) A description of why you think x converged to its final place in each case, justified by the visualization of the objective function.
4. Part 4:
   (a) Your four plots of epoch vs. J with the terminal values of x and J superimposed on each.
5. Part 5:
   (a) Your plot of *epoch* vs J.
6. Part 6:
   (a) Graph of J vs *epoch* for the training data set.
   (b) Final training accuracy.
   (c) Any additional design/hyperparameter decisions, and why.