CS 615 - Deep Learning

Assignment 5 -MLPs
Winter 2022

Introduction

In this assignment we will explore the design and use of multi-layer perceptrons for multi-class classification.

Programming Language/Environment

As per the syllabus, you may work in either Matlab or Python 3.x. If you are working in Python 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 function 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 slack.

Grading

Part 1	Theory	10pts
Part 2	Multi-Class Logistic Regression	30pts
Part 3	ANN	30pts
Part 4	Multi-Layer MLP	30pts
Extra-Credit	Reducing Overfitting	10pts

Table 1: Grading Rubric

Datasets

MNIST Database The MNIST Database is a dataset of hand-written digits from 0 to 9. The *original* dataset contains 60,000 training samples, and 10,000 testing samples, each of which is a 28×28 image.

To keep processing time reasonable, we have extracted 100 observations of each class from the training datase,t and 10 observations of each class from the validation/testing set to create a new dataset in the files *mnist_train_100.csv* and *mnist_valid_10.csv*, respectively.

The files are 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 original dataset, you can visit: http://yann.lecun.com/exdb/mnist/

1 Theory

1. (10pts) In class and the lecture notes, we provided the gradient of the *tanh* function without actually walking through the derivation. For this assignment's only theory question, show the work on how the partial derivative of the tanh function, $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, is $\frac{\partial g(z)}{\partial z} = (1 - g^2(z))$

2 Multi-Class Logistic Regression

In the previous assignment we used a cross-entropy loss function when doing multi-class classification. Just to mix it up, let's see what happens if we use a sigmoid activation function and log-loss objective function to do multi-class classification. Your existing implementation should work "out-of-the-box" to do this, although you will have to *one-hot-encode* your targets.

Given the following architecture, perform gradient descent to learn the weights. Hyperparameter decisions are up to you, although they should be described in your report. In addition, your report should include a plot showing the value of the objective function for the training and validation datasets as a function of the epoch, and the final training and validation accuracies.

The architecture is:

Input $\rightarrow$ Fully-Connected $\rightarrow$ Sigmoid Activation $\rightarrow$ Log Loss Objective

3 Artificial Neural Networks

Next let's add in an additional fully-connected and activation function (which we'll refer to as a *hidden layer* moving forward) so that we have an *artificial neural network*.

Now it's up to you to make some design decisions!

- How many outputs for the first hidden layer?
- What activation functions to use?
- What objective function to use?
- Other decisions related to learning rate, termination, overfitting, etc..

Try at least three different designs, and in your report provide these design decisions and the resulting training and validation accuracies for each.

HINT: You may want to play with the learning rate and/or incorperate ADAM optimization to avoid things like local minima, exploding or vanishing gradients, saddle points, etc...

4 Multi-Layer Perceptron

Now let's allow for multiple hidden layers!

Try at least three architectures, each of which have at least two hidden layers (and at least one that has more than two).

In your report, in addition to any hyperparmameter choices, provide a *table* reporting the training and validation accuracies vs the architecture.

Architecture	Training Accuracy	Validation Accuracy
TODO	TODO	TODO

Table 2: Comparison of accuracies for different MLP architectures

5 Extra Credit: Reducing Overfitting

You may have noticed, that with increased model complexity, so came increased overfitting. To help overcome this, use one of the techniques discussed in class (and in the *Improving Learning* slides) to attempt to reduce overfitting.

In your report provide:

- A description of your technique to reduce overfitting.
- Plot of training and valdiation objective evaluation vs epoch before and after attempting to reduce overfitting.
- Final training and validation accuracies before and after attempting to reduce overfitting.

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) Answer to the theory question.
2. Part 2:
   (a) Any hyperparameter choices.
   (b) Your plot of the objective function for the training and validation sets as a function of the training epoch.
   (c) Your final training and validation accuracies.
3. Part 3:
   (a) A list/table of your design choices and resulting trainining and testing accuracies.
4. Part 4:
   (a) Any hyperparameter choices.
   (b) Table of training and validation accuracies for different MLP architectures.
5. Extra Credit:
   (a) Description of your technique to reduce overfitting.
   (b) Plot of training and validation evalution vs training epoch **before** reducing overfitting.
   (c) Plot of training and validation evalution vs training epoch **after** reducing overfitting.
   (d) Final training and validation accuracies before and after reducing overfitting.