CS 615 - Deep Learning

Assignment 1 - Artificial Neurons
Spring 2020

Introduction

In this assignment we will train artificial neurons using different activation and objective functions for the task of face recognition.

Programming Language/Environment

While you may work in a language of your choosing, if you work in any languages other than Matlab, you **must** make sure you code can compile (if applicable) and run on tux. If you need a package installed on tux reach out to the professor so that he can initialize the request.

Allowable Libraries/Functions

In addition, you **cannot** use any libraries 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 (Intro to Gradient Descent)	20pts
Part 3 (Gradient Descent Logistic Regression)	30pts
Part 4 (Gradient Descent Cross Entropy)	30pts
Report	10pts
TOTAL	100pts

Table 1: Grading Rubric

Datasets

Yale Faces Datasaet This dataset consists of 154 images (each of which is 243x320 pixels) taken from 14 people at 11 different viewing conditions (for our purposes, the first person was removed from the official dataset so person ID=2 is the first person).

The filename of each images encode class information:

$$\text{subject} < ID > . < condition >$$

Data obtained from: http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

1 Theory

1. Given a class probability distribution of $\{0.1, 0.8, 0.1\}$, what is the cross entropy if the correct class is the first class? (3pts)

2. If we're using cross-entropy as our objective function, are we attempting to minimize it or maximize it (2pts)?

3. Given the confusion table in Figure 1:

   - What are the class priors? (3pts)
   - What is the overall accuracy of the system? (2pts)

		True Class			
		1	2	3	4
Predicted Class	1	5	2	3	4
	2	8	12	30	4
	3	0	8	45	4
	4	10	0	5	80

Figure 1: Confusion Matrix

2 Gradient Descent

In this section we want to visualize the gradient descent process for a simple function (you can think of this as our objective function):

$$J = (\theta_1 + \theta_2 - 2)^2$$

Initializing your parameters to zero, determine the gradients $\frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}$ and choose a learning rate and termination criteria.

For each iteration, compute J, and use this to generate a 3D plot of θ_1 vs θ_2 vs J. It likely will look something like Figure 2.

$$\frac{\partial J}{\partial \theta_1} = 2(\theta_1 + \theta_2 - 2)$$

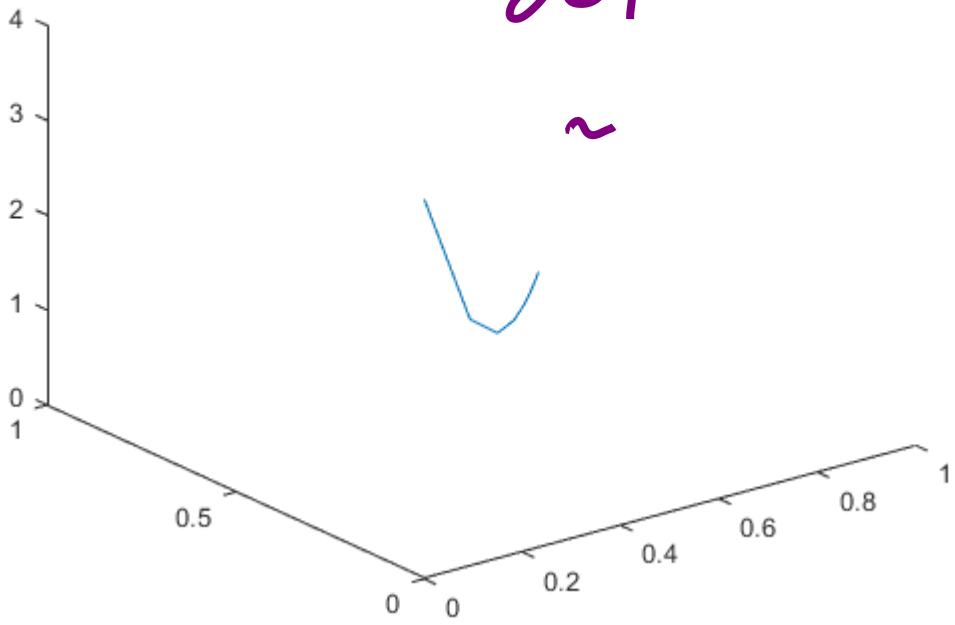

Figure 2: Gradient Descent Progress

3 Gradient Descent Logistic Regression

For the next two programming components we're going to tackle the problem of face recognition using different activation and objective functions.

Download and extract the dataset *yalefaces.zip* from Blackboard. This dataset has 154 images ($N = 154$) each of which is a 243x320 image ($D = 77760$). Since we'll be attempting to do face recognition, we'll be using the *subject* numbers for a class labels.

In order to process this data your script will need to:

1. Read in the list of files
2. Create a 154×1600 data matrix, and a 154×1 target vector such that for each image file
   (a) Read in the image as a 2D array (234x320 pixels)
   (b) Subsample/resize the image to become a 40x40 pixel image (for processing speed). I suggest you use your image processing library to do this for you.
   (c) *Flatten* the image to a 1D array (1x1600)
   (d) Concatenate this as a row of your data matrix.
   (e) Use the **subject** number to enumerate this image's class in your target vector.

Now that you have your data ready, let's separate it into training and testing sets, learn from the training set, and apply it to the testing set!

Write a script that:

1. Randomizes (shuffles) your data.
2. Selects the first 2/3 (round up) of the data for training and the remaining for testing. **However**, make sure that class priors in both the training and testing sets are similar, if not identical.
3. While training:
   (a) Compute the average log likelihood of your training and testing sets.
   (b) Update each parameter using *batch* gradient descent with a logistic activation function and log likelihood objective function.
4. Compute the testing accuracy.
5. Generate the confusion matrix.

Implementation Details

1. Seed the random number generator prior to your algorithm for reproducibility.
2. Standardize the data based on the **training** data. However, if any features have zero standard deviation, either set them to zero or remove them.
3. Don't forget to add in the bias/offset feature (but don't standardize it)!
4. Do **batch** gradient descent

Hyperparameters You'll also have to experiment with different hyper-parameters. These include:

1. How to initialize your model (i.e the initial values of θ.
2. Your learning rate.
3. Your L2 regularization amount.
4. How/when to terminate the training process.

What you will need for your report

1. The values of the hyperparameters that you chose.
2. A graph of the average log likelihood for the training and testing sets as a function of the training iteration number.
3. The testing accuracy.
4. The confusion matrix for the testing data.

4 Gradient Descent w/ Softmax and Cross-Entropy

Repeat everything from the previous section, but now using a softmax activation function and a cross-entropy objective function.

Additional considerations:

- Change your plot and termination criteria to use the average cross entropy.
- In the case that you have an invalid distribution for the output vector (all zeros), set it to be the priors.

What you will need for your report All the same stuff as in the previous part!

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) Hyperparameter choices
   (b) Final Testing Accuracy
   (c) Confusion Matrix
   (d) Plot of average log likelihood for Training and Testing Data vs Gradient Descent iteration number
4. Part 4:
   (a) Hyperparameter choices
   (b) Final Testing Accuracy
   (c) Confusion Matrix
   (d) Plot of average cross entropy for Training and Testing Data vs Gradient Descent iteration number