CS 615 - Deep Learning

Assignment 2 - Artificial Neural Networks
Spring 2020

Introduction

In this assignment we'll implement a multi-layer artificial neural network for the task of image 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 Questions)	15pts
Part 2 (Shallow ANN)	50pts
Part 3 (Multi-Layer ANN)	25pts
Report and Simplicity to Run	10pts
TOTAL	100pts

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. Another common activation function is the hyperbolic tangent function, *tanh*, which is defined as:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{1}$$

   What is $\frac{\partial tanh(x\theta)}{\partial \theta_j}$ (5pts)?

2. Given a network with a single hidden layer, a softmax activation function at the output, a linear activation at the hidden layer, and a cross-entropy objective function what is:

   (a) $\frac{\partial J}{\partial \theta_{kj}}$ (5pts)

   (b) $\frac{\partial J}{\partial \beta_{ij}}$ (5pts)

2 Shallow Artificial Neural Networks

NOTE I advise you to read the last part of this assignment before starting on this part. By completing the last part you will have completed this part as well. However, a shallow ANN is easier to implement than a generic deep ANN, so it's up to you how you want to tackle this assignment.

As with your previous assignment, download and extract the dataset *yalefaces.zip* from Blackboard. This dataset has 154 images ($N = 154$) each of which is a 243x320 image ($D = 77760$). In order to process this data your script will need to:

1. Read in the list of files
2. Create a 154x1600 data matrix 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.

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!

You are to create, train, and test an artificial neural network containing a single hidden layer (which is really just a special case of an arbitrarily deep artificial neural network). You are to do this *from scratch*, not using any ANN library functions.
All non-input layers will use a logistic activation function. Our objective function will be the log likelihood.

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. Standardizes the data based on the **training** data.
4. Trains an artificial neural network using the training data
5. During the training process, computes the training and testing log likelihoods after each iteration. You will use this to plot the log likelihood vs. iteration number.
6. Classifies the testing data using the trained neural network.
7. Outputs the final testing accuracy and the confusion matrix.

Implementation Details

1. Seed the random number generator prior to your algorithm for reproducibility.
2. Don't forget to add in the bias/offset feature (but don't standardize it)!
3. Remove any features that had a standard deviation of zero.
4. Do **batch** gradient descent
5. Terminate when absolute value of the percent change in the average log likelihood is less that 0.1%.
6. To avoid $log(0) = -Inf$, for any probabilities less that 0.0001, set them to 0.0001.

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 regularization amount (use L2 regularization).

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.

3 Multi-Layer ANN

Repeat all of the same stuff as in the previous problem, but now allow a user to provide a vector (array, list, etc..) of weights, which will dictate how many hidden layers there are, and how many nodes are in each hidden layer.
For instance, if I provide the vector $[100, 30, 20]$, my ANN will have three hidden layers (and therefore five total layers including the input layer and output layers). In addition, the first hidden layer (the one closest to the input layer) will have 100 nodes in it, the second hidden layer will have 30 nodes in it, etc...
Run your code for several different network configurations (at least three), and report the testing accuracy for each.

What you will need for your report

1. The values of the hyperparameters that you chose.
2. A table of network configurations and their associated number of iterations till termination, training and testing accuracies.

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) 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
3. Part 3:
    (a) Hyperparameter choices
    (b) Table of results for different network configurations.