CMPE297: Deep Learning

Lab 1 Assignment: ANN/DNN Using TensorFlow

Due: October 15, 2017

It is an individual programming assignment. This lab assignment covers ANN/DNN in TensorFlow This lab assignment is graded based on 30 points and is an individual effort (e.g, no teamwork is allowed).

Prerequisites

- You should have the python and TensorFlow setup ready on your Laptops
- You must know programing language basics, python and it's libraries for deep learning as mentioned in Programming Refresher Assignment, Tensor Flow.

Grading

Problem 1 (15pts) and Questions (5pts), Problem 2 (25 pts) Questions (5pts): total 50 pts.
Late assignments will be accepted, but will be subject to a penalty of -5 points per day late:

- Submissions received at or before the class on the due date can receive maximum

Problem 1 - Improving the accuracy of the MNIST model.

We are able to achieve an accuracy of ~90% with the logistic regression model we discussed in class. The dataset is basically solved and state of the art models reach accuracies above 99%. (http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html). You can use different loss functions, optimizers, even models that you want, as long as
your model is built in TensorFlow. We are expecting accuracy of at least 97%. (hint: implement it with multi-layer ANN and/or dropout and/or efficient initialization functions).
Add tensorboard to observe cost functions.

Questions (problem 1)

1. Describe the techniques you have used for improving accuracy if any.
2. In 2A, try to use high value of learning rate (say 2.5) and observe/record cost functions and discuss results why that is the case. Also use small value of learning rate (say, 1e-10) and discuss the same.

Problem 2 - You are supposed to build a Logistic Regression model to predict whether someone has coronary heart disease.

The data will be found in heart.csv. The first row is the name of the observed variables. There are 10 variables:

- sbp: Systolic blood pressure
- tobacco: Cumulative tobacco consumption, in kg
- ldl: Low-density lipoprotein cholesterol
- adiposity: Adipose tissue concentration
- famhist: Family history of heart disease (1=Present, 0=Absent)
- typea: Score on test designed to measure type-A behavior

- obesity: Obesity
- alcohol: Current consumption of alcohol
- age: Age of subject
- chd: Coronary heart disease at baseline; 1=Yes 0=No

Each following row contains the information of one patient. There are 462 samples in total.

Step 1 – Data Preprocessing

1. Read the input from the csv file using any library of your choice (Ex: pandas.read_csv('xyz.csv')
2. Do encoding and feature scaling if applicable (You can use TensorFlow or sklearn libraries for encoding and feature scaling)
3. Split the dataset into test and trainset to feed the placeholders X and Y for training and testing the network

Step 2 -Create placeholders for features i.e input and output

1. Name the input placeholder as X
2. Name the output placeholder as Y

Step 3 – Create weights and bias variables

1. w is initialized to random variables with mean of 0, stddev of 0.01
2. b is initialized to 0
3. shape of w depends on the dimension of X and Y so that Y = tf.matmul(X, w)
4. shape of b depends on Y

Step 4 – Build model

1. Model that returns logits. You can use the below code
   logits = tf.matmul(X, w) + b

Step 5 – Calculate entropy and loss

1. Use the below code to calculate entropy

   tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y, name='loss')

2. Use **tf.reduce_mean**(calculated entropy) to calculate loss. Store the loss value in a variable 'loss'

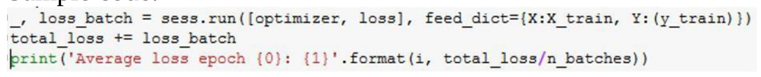
Step 6 – Use optimizer and minimize loss

1. Use one of the optimizers with a learning rate (0.01 or of your choice to yield better results) and minimize loss.
   Sample Example: optimizer = tf.train.AdamOptimizer(learning_rate).minimize(loss)

Step 7 – Use Session

1. Initialize all the variables
2. Calculate number of batches. (training data size / batch size)
3. Train the model for as many times as no of epochs.
4. For each batch, calculate loss and total loss so that you get average loss per epoch.

Sample code:



```
_, loss_batch = sess.run([optimizer, loss], feed_dict={X:X_train, Y:(y_train)})
total_loss += loss_batch
print('Average loss epoch {0}: {1}'.format(i, total_loss/n_batches))
```

Step 8 – Test the model

1. Calculate the predicted output. You can use **tf.nn.softmax(logits)** to get the predictions.
2. Calculate the correct predictions by comparing the predicted output with the actual output. You can use **tf.equal** function for the same.
3. Calculate number of batches and batch accuracy in the same way as training batch accuracy
4. Calculate total number of correct predictions
5. Print the accuracy (Accuracy = total correct predictions/ No of test samples)

Questions (problem 2)

1. Describe the Data preprocessing techniques you have used if any?
2. Describe the activation functions you have used in different layers if it is a multi-layered network? Explain the choice of activation function.
3. Describe the techniques you have used for improving accuracy?
4. Describe the role of number of epochs and batch size in neural networks performance in this specific case (for the heart.csv dataset)

Deliverables Required:

- **Lab 1 report Submission**
  - Introduction: state your goals, choice of neural network
  - Network Design: Describe your chosen network design.
  - Results: Screen image captures of Accuracies for each iteration/ one final accuracy after testing the network.
  - Performance: What was performance? Analyze results and explain why are you getting those results.
  - HTML export of exection
  - The answers to the questions posed

- **Lab 1 Codes Submission**
  - **ipynb** file containing the codes zipped into your last name (ex. Lab1-Smith.zip)

Submission

- **On-line submission**: Submissions shall be made via Canvas in each category