Middle East Technical University Department of Computer Engineering

CENG 498

Introduction to Machine Learning

Spring 2017-2018

Homework 1 - Artificial Neural Network

Due date: 25 03 2018, Sunday, 23:59

1 Introduction

The objective of this assignment is to implement a classifier using an Artificial Neural Network (ANN) and draw its graphs according to specifications. Your task is to train various ANNs with different network architectures and hyper-parameters on two different datasets for classification. You will implement your networks with python using tensorflow library. Based on the provided parameters, you will then plot the training graphs for each different parameter provided to you. You will optimize your hyperparameters and plot the graph for the network with best classification accuracy and give the final classification rate. The homework consists of two parts. Two different datasets provided to you for each part and you will use these to train your networks. All the network and parameter details for training is provided to you for the first part. You need to implement the network according to these specifications. However, in the second part you will create your own network architecture and decide on your parameters.

2 PART 1 - 70 pts

For the first part of the assignment, you will work on banknote forgery classification dataset with three distinct architectures and optimize its parameters at the end. The details are given in the following subsections.

2.1 Dataset

In these dataset, you will be given 5 attributes and the last one is the classification result. There are two classes. Using the four attributes about the certain characteristics of the images of the banknote, you will train a classifier using the dataset. You can examine the details from `https://archive.ics.uci.edu/ml/datasets/banknote+authentication`. The dataset is divided into training and test parts and converted into python arrays. These arrays are encapsulated using pickle and will be provided to you for training and test results. Use the following code to load the dataset:

```
import pickle

bank = pickle.load( open('bank.pkl', 'rb'))

x_train = bank['x_train']
y_train = bank['y_train']
x_test = bank['x_test']
y_test = bank['y_test']
```

These arrays are explained below:

- `x_train` is the training input array of shape (914, 4) where the each of the 4 attributes is a floating point number.
- `y_train` is the training output array of shape(914, 2) where each value is the probability of corresponding input in `x_train` being in one of the classes. The two possibilities are `[0.0, 1.0]`, `[1.0, 0.0]`.
- `x_test` is the test input array of shape (458, 4) with same characteristics as training input.
- `y_test` is the test output array of shape (458, 2) with same characteristics as training output.

You will use training input/output for training your networks and test input/output for testing. If you use test data for training you will not gain any points from this part.

2.2 Networks and Reports

You will implement 3 different network architectures and 3 different activation functions for each architecture. In total you will have 9 different networks and results. After these 9 results, you will optimize your certain hyperparameters and report your best result. For the first 9 networks, following characteristics should be same:

- Activation function at the output layer should **softmax** function and you should use softmax cross entropy as your loss function.
- Since the output shape of the network should match the data, the number of units in the final layer is always 2.
- Learning rate should be 0.01
- Gradient descent should be used for optimization
- Epoch is passed when all the input/output pairs are used once. (session.run() with all input/output pairs in `feed_dict` parameter.)
- During training, the network should print the following at the end of each epoch:

```
Epoch: <epoch> Loss: <loss>
```

- After training is complete, the network should print the classification accuracy using the test data in the format:

```
Test Accuracy: <test_accuracy between 0.0 and 1.0 values>
```

It is important to note that for the test accuracy, your network outputs may not be normalized and can be negative. You can select the highest value as the correct class (see `tf.argmax` function).
Three activation functions for each network is given below:

- Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Tanh:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

- ReLU:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

You can implement them yourself if you want however they are already implemented in the tensorflow library. You are free to use it.
For each of the 9 networks, draw a (training loss)/epoch graph in the corresponding place in the given latex file and report its final accuracy. Small changes can between each run and you can put results from any run into your reports.

2.2.1 First Architecture - 15 pts

First architecture is a two-layer neural network and the input should be fed directly into the first layer. The number of units in the first layer is 3. You need to implement three networks using this architecture. The only difference is the activation function of the first layer and they are given in subsection Networks and Reports. Training should stop at epoch 2000. Fill the corresponding places in your latex report based on your results.

2.3 Second Architecture - 15 pts

First architecture is a three-layer neural network and the input should be fed directly into the first layer. The number of units in the first layer is 4 and the second layer is 3. You need to implement three networks using this architecture. The only difference is the activation function of the first two layers and they are given in subsection Networks and Reports. Training should stop at epoch 4000. Fill the corresponding places in your latex report based on your results.

2.4 Third Architecture - 15 pts

First architecture is a four-layer neural network and the input should be fed directly into the first layer. The number of units in the first three layers are 8, 4, and, 3 in the given order. You need to implement three networks using this architecture. The only difference is the activation function of the first three layers and they are given in subsection Networks and Reports. Training should stop at epoch 8000. Fill the corresponding places in your latex report based on your results.

2.5 Hyperparameter Optimization

(25 pts) In this part, you will use the third architecture and optimize your training hyperparameters. The parameters you are going to optimize are the learning rate of your network and the activation functions of first three layers. Optimization should stop at 2000 epochs. You will only use training data for optimization and explain how you can use this data. Optimization parameters and optimization values are given below:

1. Layer1 activation function: sigmoid, tanh, relu
2. Layer2 activation function: sigmoid, tanh, relu
3. Layer3 activation function: sigmoid, tanh, relu
4. Learning rate: values in [0.01, 0.1] with 0.01 intervals

For each optimization parameters, fill the table with values based on the optimization criteria. After optimization, train and test the best network. Training should stop at 8000 epochs. Draw its training loss/epoch graph and write the final classification accuracy.

3 PART 2 - 30pts

For the second part of this homework, you will be given a more complicated dataset. Using this dataset, you will create your own network with complete control over its architecture, parameters, loss function and optimization algorithm. After determining the network architecture, you will optimize your hyperparameters, train and test your final network. Using these values you will fill the necessary information on the latex report, draw its (training loss/epoch) and write its final classification accuracy. 20 points of this part will be determined by the final classification accuracy ranking among all the submissions. 10 points will be for the implementation and the report.

3.1 Dataset

In these dataset, you will be given 10 attributes and the last one is the classification result. There are two classes. You can examine the details from `https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope`. The dataset is divided into training and test parts and converted into python arrays. These arrays are encapsulated using pickle and will be provided to you for training and test results. Use the following code to load the dataset:

```
import pickle

magic = pickle.load( open('magic.pkl', 'rb'))

x_train = magic['x_train']
y_train = magic['y_train']
x_test = magic['x_test']
y_test = magic['y_test']
```

These arrays are explained below:

- `x_train` is the training input array of shape (12679, 10) where the each of the 10 attributes is a floating point number.
- `y_train` is the training output array of shape(12679, 2) where each value is the probability of corresponding input in `x_train` being in one of the classes. The two possibilities are `[0.0, 1.0], [1.0, 0.0]`.
- `x_test` is the test input array of shape (6341, 10) with same characteristics as training input.
- `y_test` is the test output array of shape (6341, 2) with same characteristics as training output.

4 Specifications

- The codes must be in python and must use tensorflow library. Any other programming language or library will not be accepted. Python 3 is preferable but you are allowed to use Python 2 as well.
- You programs should be able to run on Ubuntu with tensorflow installed.
- Falsifying results, changing the composition of training and test data are strictly forbidden and you will receive 0 if this is the case. Your programs will be examined to see if you have actually reach the results and if it is working correctly.
- You have total of 3 late days for all your homeworks. For each day you have submitted late, you will lose 10 points. If your late days exceed 3, any homework you have submitted late will be 0. Your remaining late submission days will be displayed together with your homeworks on cow.
- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. The violators will be punished according to the department regulations.
- Follow the course page on cow for any updates and clarifications. Please ask your questions on cow instead of e-mailing if the question does not contain code or solution.

5 Submission

Submission will be done via COW. You will submit a tar file called "hw1.tar.gz" that contain all your source code together with your report in a pdf format compiled from the given latex file. Your tar files should not contain any folders. Your source code should consist of 11 files with following names:

1. **arch1sigmoid.py**: For first architecture with sigmoid activation
2. **arch1tanh.py**: For first architecture with tanh activation
3. **arch1relu.py**: For first architecture with relu activation
4. **arch2sigmoid.py**: For second architecture with sigmoid activation
5. **arch2tanh.py**: For second architecture with tanh activation
6. **arch2relu.py**: For second architecture with relu activation
7. **arch3sigmoid.py**: For third architecture with sigmoid activation
8. **arch3tanh.py**: For third architecture with tanh activation
9. **arch3relu.py**: For third architecture with relu activation
10. **arch3opt.py**: For the hyperparameter optimization, training and test
11. **part2.py**: All your implementation for part2 including hyperparameter optimization, training and test