CSCI 184 HW 4 -- Programming

Due Date: June 7, 11:59pm 2024

All assignments MUST have your name, student ID, course name/number at the beginning of your documents.

Your homework MUST be submitted via Camino with the file format and name convention as follows:

For Question Answering part, you can either write by hand or type your answers, but please ensure your submission is **pdf** file with name "**HW#_Name.pdf**".

For programming questions, please upload your code and supporting files in "**HW#_Name.zip**".

If you have any questions, please don't hesitate to contact me :)

Neural Networks – Monks' problem

For this part, we will use the monks' dataset that we have used in HW1, and use Neural Network to do Classification. Specifically, the training and test file is named monks-1.train and monks-1.test. There are six attributes/features (columns 2–7), and binary class labels (column 1). See monks.names for more details.

Requirements:

1. Implement a 3-layer neural network, with the number of nodes and activation function of each layer shown as follows:

	Layer 1	Layer 2	Layer 3
Number of nodes	8	6	1
Activation function	Hyperbolic tangent	ReLU	Sigmoid

You can either use the Sequential and Dense layer from Keras OR scikit-learn MLPClassifier OR implement you own neural network function. Next page includes a simple example of using Keras to construct a 2-layer neural network.

2. Use the cross-entropy function as the loss function as we discussed in the class for the logistic regression.

3. Set the learning rate as 0.01 or change to other values if you find 0.01 is not suitable for this problem.

4. For the two optimizers: stochastic gradient descent and Adam, record the testing accuracy for the following epochs 5, 10, 15, 20, 25, 30, 35, 40, 45, 50. Plot the testing accuracy of these two optimizers with respect to the epochs number.

2-layer NN using Keras with 12 nodes in the hidden layer.

```
from keras.models import Sequential  from keras.layers import Dense
model = Sequential()

model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(1, activation='sigmoid’))
model.compile(loss='binary_crossentropy', optimizer='sgd’)
model.fit(X, y)
```