Machine Learning Assignment 4
Neural Networks
Submission deadline: May 03, 2020, 23:59 IST

In this assignment, you will implement some simple multi layer perceptrons and neural networks.

Dataset : [5 points]

1. Download the SEEDS dataset: Click Here
2. It has 210 data points each with 7 attributes, and 3 classes (1,2,3) denoting three different varieties of wheat. Consider the output as a one-hot vector. E.g., if a datapoint has a class '2', consider the output of that data point as [0 1 0]. Normalize each attribute by **z-score normalization:** for each feature, every value is to be normalized to (value - m)/s, where m is the mean value of the feature, and s is the standard deviation of the feature**.**
3. Consider 80% of the dataset (randomly selected) as training data, and the rest 20% as test data.
4. **Save this training and test data. In the rest of this assignment, you will be asked to develop various NN classifiers, and each classifier should be trained using this training data, and evaluated using this test data. In other words, each NN should be trained and tested using the same datasets.**

You can use any library of your choice for preparing the dataset.

Part 1 : Building your own Neural Network

Build a neural network and perform the classification task with the specifications mentioned in Parts 1A and 1B. Your implementation should have the following modules :

1. *Preprocess:* Use this module to preprocess the data and divide into train and test.
2. *Data loader* : Use this module to load all datasets and create mini batches, with each minibatch having 32 training examples.
3. *Weight initialiser* : This module should initialize all weights randomly between -1 and 1.
4. *Forward pass:* Define the forward() function where you do a forward pass of the neural network.
5. *Backpropagation* : Define a backward() function where you compute the loss and do a backward pass (backpropagation) of the neural network and update all weights.
6. *Training* : Implement a simple mini batch SGD loop and train your neural network, using forward and backward passes.
7. *Predict*: To test the learned model weights to predict the classes of the test set.

1A. NN Specification 1 **[30 points]**

1. No of hidden layers : 1
2. No. of neurons in hidden layer: 32
3. Activation function in the hidden layer : Sigmoid
4. 3 neurons in the output layer.
5. Activation function in the output layer : Softmax
6. Optimisation algorithm : Mini Batch Stochastic Gradient Descent (SGD)
7. Loss function : categorical cross entropy loss
8. Learning rate : 0.01
9. No. of epochs = 200

The sigmoid function is defined as : $\sigma(x) = \frac{1}{1+e^{-x}}$

Its derivative : $\sigma'(x) = \sigma(x)\,(1-\sigma(x))$

You should define the activation functions and their derivatives. **Do not use any inbuilt library for this.**

1B. NN Specification 2 **[45 points]**

1. No of hidden layers : 2
2. No. of neurons in the 1st hidden layer: 64
3. No. of neurons in the 2nd hidden layer: 32
4. Activation function in both the hidden layers : ReLU
5. 3 neurons in the output layer.
6. Activation function in the output layer : Softmax
7. Optimisation algorithm : Mini Batch Stochastic Gradient Descent (SGD)
8. Loss function : categorical cross entropy loss
9. Learning rate : 0.01
10. No. of epochs = 200

The function ReLU is defined as : $f(x) = max\,(0, x)$

Its derivative : $f'(x) = 0,\ if\ x <= 0$

$= 1,\ otherwise$

You should define the activation functions and their derivative. **Do not use any inbuilt library for this.**

Your code must output the following for each of the two specifications :

1. Plot the training accuracy and test accuracy after every 10 epochs (in a single plot)
2. Print the final training accuracy
3. Print the final test accuracy

Note:

- **For Part 1, you should not use any ML library like scikit learn. However, you can use numpy.**
- **Try using numpy array computations in Python, and avoid using loops as it may take a long time to train. Also it may be easier to implement in an object oriented format.**
- **You can also write a generalizable code to implement a Neural Network, where one can specify various hyperparameters such as the number of layers, number of neurons in each layer, what non-linearity to use, what loss function to use, number of epochs to train for, etc. Such a code can be reused for both Part 1A and 1B.**

Part 2 : Implementation with scikit learn [20 points]

Use the MLP implementation of scikit learn: Click here for details
Use the specifications from Part 1A and Part 1B, and use the same training and test data as in Part 1. For each specification, your code must output :

1. Final train accuracy
2. Final test accuracy

Note: Use the same training and test data in Part 1A and 1B, as well as in Part 2.

Submission Instructions

Submit **two (2) code files -- one for Part 1 and one for Part 2**. When we execute your code, we should be able to see all the outputs. Mention the question no. with its corresponding output. Follow this format strictly:

When we run your code for Part 1, your output should be :
Part 1A :
<plot & output>

Part 1B :
<plot & output>

When we run your code for Part 2, your output should be :
Part 2 Specification 1A :
<output>

Part 2 Specification 1B :
<output>

Comment your codes sufficiently, use meaningful variable names. Keep the training and test dataset in a separate folder called “data”. Also submit a README file which will contain the instructions on how to execute your codes.

All source codes, result files and the README file must be uploaded via the course Moodle page, as a single compressed file (.tar.gz or .zip). The compressed file should be named as: {ROLL_NUMBER}_ML_A4.zip or {ROLL_NUMBER}_ML_A4.tar.gz
Example: If your roll number is 16CS60R00, then your submission file should be named as 16CS60R00_ML_A4.tar.gz or 16CS60R00_ML_A4.zip

Note that the evaluators can deduct marks if the deliverables are not found in the way that has been asked for the assignment, or if your codes are not understandable.

You can use one of C / C++ / Java / Python for writing the codes; NO other programming language is allowed. You need to submit raw codes in one of the specified languages, which can be executed on a Linux system from the console. Other platform-specific or software-specific formats such as ipython notebooks are NOT allowed.

You cannot use any library/module meant for Machine Learning for implementing your models, unless specifically allowed in the problem statement. You can use libraries for other purposes, such as generation and formatting of data. Also you should not use any code available on the Web. Submissions found to be plagiarised or having used ML libraries (except for parts where specifically allowed) will be awarded zero marks.

Submission deadline: ~~April 14~~, May 03 2020, 23:59 IST [hard deadline]

For any questions about the assignment, contact the following TAs:
1. Soham Poddar (sohampoddar26 @ gmail . com)
2. Paheli Bhattacharya (paheli.cse.iitkgp @ gmail . com)