CSE/ECE 343/543: Machine Learning
Assignment-3 Neural Networks and CNNs

Max Marks: 100 (Programming:70, Theory:30) **Due Date**: 10/11/2019, 11:59PM

Instructions

- Keep collaborations at high level discussions. Copying/Plagiarism will be dealt with strictly.
- Late submission penalty: As per course policy.
- Your submission should be a single zip file **name_HW3.zip**. Including only the **relevant files** arranged with proper names. A **.pdf report** explaining your codes with relevant graphs and visualization and theory questions.
- Remember to **turn in** after uploading on google classroom.
- **Bonus** of 10 points for submissions before **6 hours of deadline.**
- Resolve all your doubts from TA's in their office hours **two days before the deadline.**

1. (5+5+5+10+5+10+5 = 45 points) **Multi-layer Perceptron**
   Implement a MLP classifier with forward and back-propagation algorithms from scratch. The only library allowed for this question is numpy except for the last part.
   - Create a class **Neural_Net** which requires the following parameters as details:
     - the number of layers (Type: Int) [this includes the input layer and the output layer]. Thus, if this parameter is N, the number of hidden layers are N-2.
     - An array of size N where N is the number of layers passed above. This array contains the number of nodes in each layer. (Type: Array of Ints)
     - the activation function to be used (Type: String). Use this string later to determine which activation function needs to be called in appropriate places.
     - the learning rate (Type: Float)
   - Create the following functions inside the class:
     - **fit**: Accepts input data, input labels, batch size and epochs and returns nothing.
     - **predict**: Accepts input data and returns class wise probabilities for the given data.
     - **score**: Accepts input data and their labels and returns accuracy. Clearly, this function uses the above implemented **predict** function.
     - You may create other helper functions along with other attribute variables inside the class.

- The following activation functions should be implemented along with their gradient calculation:
  - Relu
  - Sigmoid
  - Linear
  - tanh
  - Softmax for output layer
- Ensure that your code is general and can create a network for any number of hidden layers. Do not hard code the architecture. Likewise, the forward and back propagation algorithms should be general too and should work for any number of hidden layers. Use 0.01*Normal(0,1) for initialization of weights for all hidden layers and all the biases should be initialized to 0. Use Cross Entropy Loss as the loss function.
- Use the above mentioned class to create the following architecture [#input, 256, 128, 64, #output] along with Relu activation, learning rate as 0.1, epochs as 100. Train the network on MNIST dataset and report accuracy on the test set. Save the weights of the trained model. Also Plot the training error vs epoch curve.
- Repeat the experiment with Sigmoid, Linear and TanH activation functions. Report the accuracy on test set for the 3 experiments. Save the weights for all 3 trained models. Also Plot the training error vs epoch curve for all the 3 cases.
- Implement the same architecture using sklearn's MLP classifier. Use sklearn's fit, predict and score methods to get the accuracy on test data. Try all the 4 activation functions - Relu, Sigmoid, Linear and TanH. Compare the accuracy achieved with sklearn's MLP vs your own implementation of MLP and comment on differences in accuracies if any.

*A linear activation or identity activation function is f(x) = x.

2. (10+10+5 = 25 points) **Convolutional Neural Networks** Note: You are only allowed to use the PyTorch framework in Python. Explicitly mention each parameter considered (SVM kernel, number of filters, filter size, type of pooling).

   - Train a CNN consisting of convolutional layers, pooling layers and FC layers on the FashionMNIST dataset. Ensure that the total number of layers (including CONV/POOL/FC) do not exceed 6. You are free to choose an arbitrary number of CONV/POOL/FC layers with appropriately chosen filter sizes. Report the accuracy, confusion matrix and loss-per-epoch plots on the training and test sets.
   - Use the trained CNN as a feature extractor by using the outputs of the last FC layer as feature vectors. Now train a kernelized SVM (with appropriate kernel) using these feature vectors as inputs. Report the accuracy, confusion matrix and loss-per-epoch plots on the training and test sets.
   - Compare and give insights based on the metrics obtained for both the trained models.

3. (10 points) Can a neural net of arbitrary depth using just linear activation functions be used to model the XOR truth table? Can you mathematically prove this classifier equivalent to be same as any other classifier?

4. (10 points) What are the different components of a deep Convolutional Neural Network applied to classification? Describe each component based on its need, functionality and advantages over traditional neural network architectures.

5. (10 points) Assume that you are given a black box which creates a multi-layer perceptron. Following are the commands can be used to setup the parameters and layers.
   $Connect(i, j, a, b)$: make a connection from a^{th} neuron of layer i to b^{th} neuron of layer j.
   $Share(i, j, a, b, c, d)$: Keeps the weights of connections $a- > b$ and $c- > d$ same between two adjacent layers.
   Write a psudo code (in text) for creating a CNN with 2 hidden layers for a binary classification task on a 5×5 input image. You may use for loops and may need to define few corner functions for example: $Create_{al}ayer()$.