ECE 457 B: COMPUTATIONAL INTELLIGENCE

ASSIGNMENT #2
(Artificial Neural Networks Module)

Due Date: Please upload your solutions in the Dropbox folder "Assignment#2"

by 4pm on March 12, 2021

NOTE: To help with the adequate marking of your assignments, please follow these rules carefully:

1. All work is expected to be carried out **individually.**
2. The first page of the assignment must have the name and ID of the student
3. The items required must be numbered, labelled, and in numerical order.
4. The solutions should be typewritten. All graphs should be computer generated. You can use existing libraries in Matlab, Python or others (but state the software used and attach a copy of your program code).
5. All pages must be numbered sequentially
6. Show your steps and state any additional assumption you make.
7. Presentation of your results, the organization of your copy, and completion level count for **10% of the total mark.**
8. **The minimum score for the average of all assignments is required to be 50%. Invalid submission or no submission leads to INC in the course.**
9. All copies will be checked by Turnitin similarity software
10. For inquiries related to problem 1 or to the scikit-learn library, contact TA **Ambareesh Ravi**. For inquiries related to problem 2, contact TA **Mahmoud Nasr**, For inquiries related to problem 3, contact TA **Mohita Chaudhary**

For all problems below, you may use any computational framework you wish: TensorFlow (Google), PyTorch (Facebook), Gluon (AWS, Microsoft), Keras (which acts as interface for TensorFlow library), or Scikit. For questions related to these libraries, please contact any of the TAs.

Problem 1 (Nonlinear Classifier) **30 marks**

We need to build a machine learning model to predict whether a patient in the a community has diabetes or not. The members of the community data has been included in a dataset (PIMA Indians Diabetes Database) originating from the National Institute of Diabetes and Digestive and Kidney Diseases. The datasets consists of several medical predictor variables and one target variable (whether you have diabetes or not). Based on certain diagnostic measurements included in the dataset, the objective is to diagnostically predict whether or not a patient has diabetes.

Choose a classifier of your choice to solve the problem (MLP, SVM or others).

1. **(15 marks)** Change the parameters of your classifier and provide a confusion matrix for each of the various classifiers (change C-parameters and kernels for SVM or number of nodes for MLP).
2. **(15 marks)** Write the various performance measure in terms of Accuracy, Precision, Recall, and F1 Measure. Which performance measure is most important in this problem? Why?

Note 1 : Those of you using SVM classifier, investigate for some values for the slack parameter C (seen on slide 16 of the notes) such as {0.1, 1, 10} and use a linear or non-linear kernel to deal with nonlinear separability of the original data (slide 24 of SVM lectures).
Note 2: Feel free to use Scikit-learn (especially for SVM) or other tools to develop your solution: https://scikit-learn.org/stable/.

Problem 2 (**Kohonen Self Organizng Map: Unsupervised Learning**) **30 marks**

We need to design a Kohonen self organizing map (SOM), which gives as an output some shades of color mapped over 100 by 100 grid of neurones. The idea is to cluster colors of the same shade in the same neighborhood using 3D to 2D representation of colors. The training input of the SOM are 24 colors (use shades of red, green, blue, with some yellow, teal and pink) which you can chose from the "RGB Color Table: Basic Colors" section of this page: http://www.rapidtables.com/web/color/RGB_Color.htm

Using a time varying learning rate $a(k) = a(0)\exp(-k/T)$ where k is the current training epoch (starts with epoch 0), $\alpha(0) = 0.8$, and T is the Total number of training epochs equal to 600. Note that the epoch training involves all twenty four input samples for the 24 chosen colors to the network (hint: calibrate the color codes to values between 0 and 1, instead of being between 0 and 255). The initial weights linking to all 10,000 output nodes are randomized. Basically each node has an RGB color of three random values between (0 and 255, which should be normalized to between 0 and 1 in the same way we calibrate the training data as discussed earlier)

For each winning node (i) and its neighborhood (j), we upgrade the weights according to the formulae:

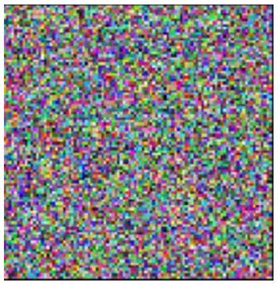
$$\mathbf{w}_{i,j}^{(new)} = \mathbf{w}_{i,j}^{(old)} + \alpha(k) N_{i,j}(k)(\mathbf{x}\text{-}\mathbf{w}_{i,j}^{(old)})$$

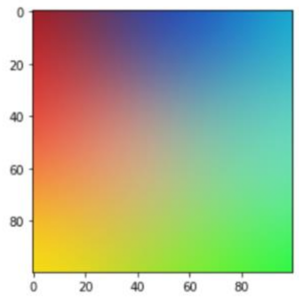
Where $N_{i,j}(k)$ is the topological neighbourhood around the winning unit (i) and is given by□

$N_{i,j}(k) = \exp(-\frac{d_{i,j}^2}{2\sigma^2(k)})$ with $\sigma(k) = \sigma_0 \exp(-k/T)$ and $d_{i,j}$ is the distance between the winning node (i) and surrounding node (j). Initial value of $\sigma_0 = 1$ doesn't provide good visualization of clustering. As such you may use values of $\sigma_0 = 10, 40, 70$.

a) **(25 marks)** Basically, and after training, we need to have as the output of the SOM colors something similar to the picture below, where similar colors cluster around each other's:

   Generate, a figure of the original SOM grid (randomly selected) followed by figures of the SOM grid after 20, 40, 100, 600 epochs for various values of $\sigma_0 = 10, 40, 70$.


Original SOM (random colors)



SOM after 600 epochs training with $\sigma_0 = 40$

b) **(5 marks)** In light of the above simulations, draw your conclusions?

Hint: Tutorial #4 material should assist you with some aspects of this problem. There is also a Matlab library and some code under this link, which you may find useful if your wish: http://www.cis.hut.fi/somtoolbox/

Problem 3 (MLP vs Deep Learning Based CNN): <u>30 marks</u>
Using your preferred deep learning library (you are free to use any library from TensorFlow, PyTorch, or Keras, which were introduced in Tutorials #1, #2, #4), we need to train a

convolutional neural network (CNN) to classify images from the CIFAR10 dataset. Note that most libraries mentioned have utility functions to download and load this dataset (in your code).

Using the API for loading the dataset, will readily divide it into training and testing sets. Randomly sample 20% of the training set in CIFAR dataset and use it as your training set for the purposes of this problem. Use the test set from the original dataset for validation. Normalize your training and testing sets using Min-Max normalization.

The following three networks (MLP and CNNs), could be readily implemented using libraries in TesnorFlow or PyTorch

1- Build a multi-layer perceptron with the following layers:
   - Dense layer with 512 units and a sigmoid activation function
   - Dense layer (output layer) with 10 units (representing 10 classes in the dataset) and a suitable activation function for the classification task
2- (10 marks) Build a Convolutional neural network with the following architecture:
   - 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
   - Flatten layer (to pass to the Fully Connected layers)
   - Fully connected (Dense) layer with 512 units and a sigmoid activation function
   - Fully connected layer with 512 units and a sigmoid activation function
   - Dense layer (output layer) with 10 units (representing 10 classes in the dataset) and a suitable activation function for the classification task
3- Build a Convolutional Neural network with the following architecture:
   - 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
   - 2x2 Maxpooling layer
   - 2D Convolutional layer with 64 filters (size of 3x3) and ReLU activation function
   - 2x2 Maxpooling layer
   - Flatten layer (to pass to the Fully Connected layers)
   - Fully connected (Dense) layer with 512 units and a sigmoid activation function
   - Dropout layer with 0.2 dropout rate
   - Fully connected layer with 512 units and a sigmoid activation function
   - Dropout layer with 0.2 dropout rate
   - Dense layer (output layer) with 10 units and a suitable activation function for the classification task

Use a batch size of 32, utilize Adam as the optimizer and choose an appropriate loss function while monitoring the accuracy in both networks. Train each network for 5 epochs.

a) **(15 marks)** Report the training and testing accuracy for all three networks and comment on the performance of the MLP vs CNNs.
b) **(15 marks)** Plot the training and validation curves for the two CNNs and comment on the output. How does the training time compare for each of the CNNs? How does the different architectures influence these results? What do you expect the accuracies to be if the networks were trained for more epochs?