Assignment 2 – Machine Learning

Marks = 10 + 3(TA)

In groups of 2, choose one of the following 4 options. The details for each option are listed on the pages that follow.

As usual, exercise and submit creative report and code!

Two milestones before the deadline:

Signup sheet + lock the project: By 12 noon, Nov 11, 2022

Mid-conference: On Nov 17, 2022

Final Submission: Email by 12 midnight, Nov 21, 2022

Option 1: Perceptron Learning simulation, on a linearly separable data,

Option 2: Hinton's classic 1986 Nature paper, presented in an Urdu vlog on YT,

Option 3: ANN/MLP trained through Evolutionary Algorithms, on a lineraly non-separable data,

Option 4: Use sklearn library on a hand-drawn roman numerals (I-X), and train a multi-class neural network for maximum training and validation accuracy.

Help for Option 1 discussion:

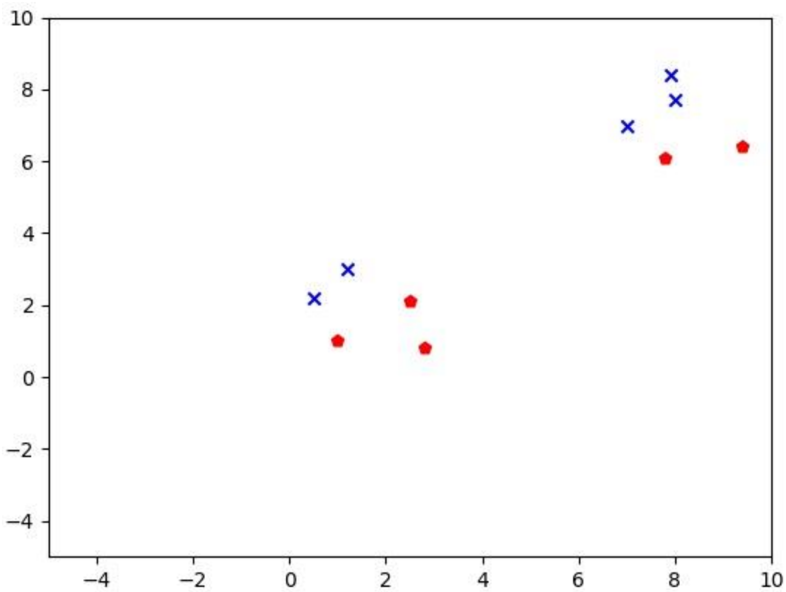
W = [-0.6, 0.75, 0.5] # initial weights

l_r = 0.02 # learning rate

epochs = 30 #total iterations over the training data

trainInput = [[1,1],[9.4,6.4],[2.5,2.1],[8,7.7],[0.5,2.2],[7.9,8.4],[7,7],[2.8,0.8],[1.2,3],[7.8,6.1]]

trainOutput = [1, 1 ,1, -1, -1, -1, -1, 1, -1, 1] #Desired Output



You have to work out (code) for this dataset, the perceptron learning algorithm, i.e. weight updation. Secondly, how do you plot the classifier line in the input space, given the three weights?

Help for Option 2:

Hinton's original backpropagation paper through gradient descent:

https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf

What does the following sentence from Wikipedia mean?
“The derivative of a function of a single variable at a chosen input value, when it exists, is the slope of the tangent line to the graph of the function at that point.”

What does partial derivative mean?

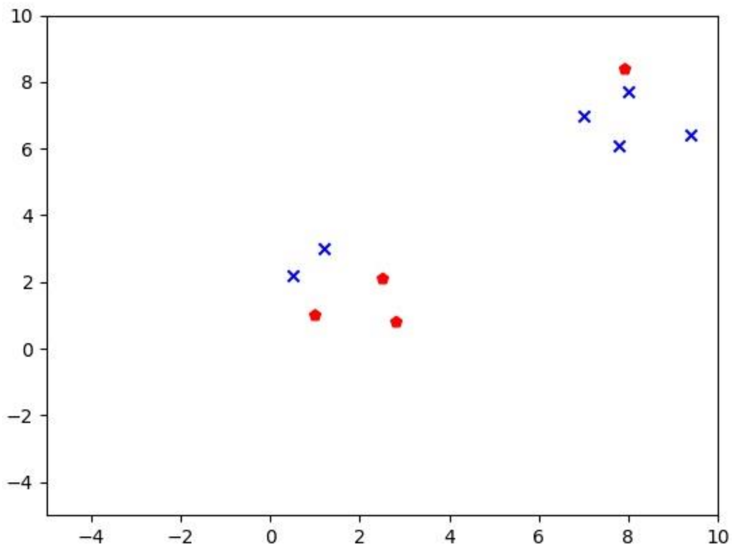
How can we use derivates to design a machine learning algorithm?

How do the weights for hidden and earlier layers get adjusted through gradient descent?

Help for Option 3 discussion:

trainInput = [[1,1],[9.4,6.4],[2.5,2.1],[8,7.7],[0.5,2.2],[7.9,8.4],[7,7],[2.8,0.8],[1.2,3],[7.8,6.1]]

trainOutput = [1, -1, 1, -1, -1, 1, -1, 1, -1, -1]



1) For this linearly non-separable dataset, unlike the dataset for Option 1 groups, you have to work out, manually the minimal architecture of neural network that would definitely be able to achieve 100% accuracy. You'll need a few neurons in the hidden layer (between the input and output).

2) Once the architecture is settled, you'd know how many weights are in the input layer, how many in the hidden layer etc, so that would give you a fixed representation of each individual in the evolutionary algorithm population.

3) Fitness function would integrate neural networks and evolutionary algorithms. How? Work that out.

Help for Option 4 discussion:

Python Library:

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

Dataset:

https://worksheets.codalab.org/bundles/0xcea1d733e1f144d9aba83929af51f191

Besides the code files, submit a detailed report describing how you explored the sklearn library, which functions you eventually used, the experimental settings you applied for each experiment of yours, including your input features, neural network architecture and parametric values, accuracy graphs etc. Finally, your informed analysis about which neural network setting was the best and why. If there are some experiments you could not try, what would they have looked like, if you had more time.

Best Wishes! :)