Department of Computer Engineering
Faculty of Engineering, University of Peradeniya
CO542
Neural Networks and Fuzzy Systems
2021
Lab 03 - Introduction to Perceptron

Objectives

- Implement and train a single layer perceptron using python libraries.
- Recognize the impact of outliers on model training time.
- Apply the knowledge in perceptron for a real-world image classification example and practice simple image processing in Python.

Single Layer Perceptron

This lab is to develop your knowledge on basic neural network techniques. While following this lab you will be asked to design single layer perceptron models for solving simple classification problems using python scikit-learn library.

This lab has 3 exercises,

- In Exercise 1, you have to create a perceptron model to solve canonical logic-gate problems.
- In Exercise 2, you must attempt to use a perceptron model to observe the training time taken for a 2 element input vector with an outlier.
- Finally you can use your knowledge to do the Exercise 3, which is an example for a real world classification problem.

Modeling and Training Perceptrons

Perceptrons can be modeled easily using scikit-learn python machine learning library.

```
# create the model
model = Perceptron()

#train the model
model.fit()

#predict from the model
model.predict()
```

For more refer,

- sklearn.linear_model.Perceptron.
- scikit-learn: machine learning in Python.

Exercise 1

1. Demonstrate the capability of a single-layer perceptron to model the following logic gates: AND , OR , NOT.
2. Generate the output curves/surfaces for these perceptron models as the input/s vary continuously from 0.0 to 1.0 (hint: surface plots in python: surface plots.)

Example : AND Gate

The following steps train an AND gate using the perceptron model.

```
from sklearn.linear_model import Perceptron
import matplotlib.pyplot as plt
import numpy as np
from itertools import product

#Create a variable named data that is a list that contains the four possible inputs to an AND gate.


data = [[0, 0], [0, 1], [1, 0], [1, 1]]
labels = [0, 0, 0, 1]

plt.scatter([point[0] for point in data], [point[1] for point in data], c = labels)
#The third parameter "c = labels" will make the points with label 1 a different color than points with label 0.
plt.show()

#Let's build a perceptron to learn AND.

classifier = Perceptron(max_iter = 40)
classifier.fit(data, labels)
print(classifier.score(data, labels))
```

Exercise 2

1. Demonstrate the behavior of a two input neuron, which is trained to classify 4 input vectors into two categories using single layer perceptron model. (Use smaller magnitudes as the values for the four input vectors).
2. Plot these vectors (Use a scatter plot).
3. Observe the training time.
4. Modify the part Exercise 2.1 such that one input vector is much larger than all the others.
5. Plot these vectors.
6. Observe the training time and compare it with the results you obtained in Exercises 2.3.
7. Classify a new input vector using model.predict() and plot this new point in the previous training set (Note: Use a different color to distinguish it from the training set).
8. Add the training set and the classification line to the plot.
9. Zoom in the plot on an area of interest.

Exercise 3

Construct a single-layer perceptron to identify a subset (of your choice) of the students in the class (not this class!). Sample jpeg image files are in the folder "classpics" in 'feels'.

Install and use the scikit-image package in python to load and manipulate image data. Some useful functions are listed below.

- imread() (Hint: refer as_gray parameter)
- imshow()
- rgb2gray()
- resize()
- rescale()

Submission

You must submit 3 folders (for each exercise) containing,

1. All required .py files (or .ipynb files).
2. A pdf containing all required outputs (images of output curves/surfaces, network training window, performance plot and etc...) and explanations before the deadline (announced in the course page)