ML/AI Practice Problems/Exercises

# First Programs:

1. Write a program that:
   1. Includes comments at the beginning with your name, date, and python version number.
   2. Outputs “Hello World”
2. Write a program that:
   1. Reads first.csv data (see Blackboard) into two lists.
   2. Create a third list by:
      1. for the even indexes (0 is even), multiply the two list elements together.
      2. for the odd indexes, add the two list elements together.
   3. Output:
      1. The minimum, average, and maximum of the odd indexes.
      2. The minimum, average, and maximum of the even indexes.

# Second Program:

Write the following program. Include appropriate comments. You should assume that users will input appropriately.

1. Get a positive integer, m, that is > 20.
2. Create two lists of random integers (ranging from 0 to 100) that are m length: x1 and x2.
3. Create two more lists of random integers (ranging from 0 to 100) that are k=3 length: c1 and c2.
4. Plot (using matplotlib) the c1,c2 pairs with red dots and:
   1. x1,x2 pairs closest to the k=1 c1,c2 pair with blue “x”s,
   2. x1,x2 pairs closest to the k=2 c1,c2 pair with green “x”s, and
   3. x1,x2 pairs closest to the k=3 c1,c2 pair with red “x”s.

# Repository Problems:

1. Multicomputer Use
   1. Create a new project called “gitProjectA” with a single python file called “helloworld.py”. The python file should initially just print “HelloWorld”. Add the Git system to the project in VSCode, create a github repository for it called “git-project-a” and connect the repositories. Push your initial version to github.
   2. On a different computer, clone your “git-project-a” repository to a new working directory for VSCode. Edit your python file by changing the print statement to “Hello New Computer”. Commit and push your changes to your github repository.
   3. On the original computer, pull your “git-project-a” repository to see the new code.
2. Resolving Conflicts
   1. With your “git-project-a” repository that is connected to more than one computer, make changes to the print statement code on each computer (do your changes before you pull/push). One should print “Hi there” and the other should print “Howdy Partner”. Stage and commit your changes – but do not push to github.
   2. Now push the change with “Hi there” to the repository.
   3. Now on the other computer (with “Howdy Partner”), try to push your changes. What happened? This is what happens when you don’t use the latest version of your code when making changes.
   4. To resolve this conflict, pull the repository version and decide which change should accept. Modify your code to print “Hi there Partner”. Then save/stage/commit/push your new version.

# Numeric Problems:

Write the following programs. Include appropriate comments.

1. Write a NumPy program to create a 3x3x3 array with random values.
2. Write a NumPy program to get the n largest values of an array.
3. Write a NumPy program to:
   1. create a 10 x 1 array, t, with random values between 1 and 10.
   2. create a 10 x 10 array, X, with random values between -1 and 1, normally distributed.
   3. Multiply X and t.
   4. Output the result.
4. Write a NumPy program to:
   1. create a 10 x 10 array, X, with random values between 0 and 1.
   2. add a column of “1”s to the X array.
   3. Output the result.
5. Write a NumPy program to:
   1. create a 100 x 100 array, X, with random values between 0 and 255.
   2. create a 1 x 101 vector, t, with random values between 0 and 1.
   3. create a 100 x 1 vector, y, with random values either 0 or 1.
   4. add a column of “1”s to the X array.
   5. multiply X and t appropriately, to get a vector h.
   6. take the sigmoid function of (h – y).
      1. sigmoid(z) = 1/(1+e^(-z))
   7. output the result.

# Linear Regression Problems:

*added 9/20/2019*

Write the following programs. Include appropriate comments.

1. Write a program to perform simple linear regression on a random linear training dataset of x,y data pairs you create. Compute the bias and weight for the linear equation. Print the resulting linear equation, your final cost, and the number of rows of training data you used.
2. Write a program to perform linear regression for multiple features (x’s) for a given y. Also include regression to minimize overfitting. Use the auto dataset on blackboard (y=mpg, with features (x’s): cylinders, displacement, horsepower, weight, model year, and acceleration). Note that you may have to perform screening of the data to eliminate spurious data rows and extraneous data. Print the resulting equation with found weights and bias, your final cost, and the number of rows of training data you used.

# Classification Problems:

*added 10/14/2019*

Write the following programs. Include appropriate comments.

1. Write a program to perform simple classification of handwritten digits (0-9) using the MNIST training dataset (see Blackboard). Print your final cost, and the F1 score.
2. Write a program to perform simple classification of a dataset of your choosing. Print your final cost and the F1 score.

# Unsupervised Learning Problems:

*revised 10/28/2019*

Write the following programs. Include appropriate comments.

1. Develop a K-means clustering algorithm program to solve the unsupervised learning (clustering) problem in two dimensions. Have the program determine how many clusters are “best”.
2. Write a program to perform a K-means algorithm to find clusters for the Power Data (given on Blackboard). You should preprocess the data to average the input values for each hour/month (scenario 1) and for each month/day (scenario 2). Report the “best” cluster centroid values for each scenario.

# CartPole Model:

*added 10/24/2019*

Write the following program. Include appropriate comments.

1. Create and train a Reinforcement Learning Model to successfully control the CartPole simulation.

# SC2 Reference Document

You will need a reference document of information that will help with coding your agent. Therefore, write a document, (be sure to include a reference section that lists where you got all your information), that summarizes/explains what you can do for each unit for each race. Include:

1. Units.
   1. (hint: https://github.com/deepmind/pysc2/blob/master/pysc2/lib/units.py )
   2. You probably want to describe what each unit can do and what its purpose it.
   3. You probably want to organize by race and then general unit type.
2. Actions
   1. (hint: https://github.com/deepmind/pysc2/blob/master/docs/environment.md#list-of-actions )
   2. note that some actions are for workers, some are for the environment (like a camera)
   3. You may want to also list appropriate actions within each unit.
3. Common Code
   1. You probably want to look at a lot of example codes and give a listing of some code lines, keywords, and/or functions that you will likely use in your code. Include lots of comments to help you to remember how to use it.

# Sources of inspiration:

<https://www.w3resource.com/python-exercises/numpy/python-numpy-random.php>