Artificial Intelligence
COMP 5600/ COMP 6600/ COMP 6600 - D01

Instructor: Dr. Shubhra ("Santu") Karmaker
TA 1: Souvika Sarkar
TA 2: Sanjoy Kundu
Department of Computer Science and Software Engineering
Auburn University
Spring, 2024

March 12, 2024

Assignment #4
Machine Learning

Submission Instructions

This assignment is due on **March 26, 2024, at 11:59 pm**. Please submit your solutions via Canvas (https://auburn.instructure.com/). You should submit your assignment as a PDF and any source code files as necessary. Please do not include blurry scanned/photographed equations, as they are difficult for us to grade.

Late Submission Policy

The late submission policy for assignments will be as follows unless otherwise specified:

1. 75% credit within 0-48 hours after the submission deadline.
2. 50% credit within 48-96 hours after the submission deadline.
3. 0% credit after 96 hours after the submission deadline.

Overview:

In this assignment, you will implement and analyze three machine learning algorithms. Each algorithm will focus on a different problem scenario that different learning methods can solve. You may use external libraries as you like, ***but you will get 30 bonus points if you implement all algorithms from scratch using Python***. You can also use other libraries for data loading and pre-processing as necessary.

- **Problem 1 [33 Points]:** See the attached housing data (`Asssignment2_NB_Data.xlsx`). Each tab in the Excel file contains training and test splits. Your goal is to construct a Naïve Bayes classifier for this data.

  1. Compute and show the conditional probability distribution for each feature. Explain how you got these values and show your work. Note: You are expected to do this part of the question by hand. Explain how you got the probability distribution for at least two features in detail.
  2. Using your conditional probability table, write a Python code that will compute the probabilities for each example in the test data. Your program should output the probabilities of each class as well as the final classification based on the MAP rule. Note: You should hard-code the conditional probabilities from the previous step into your code.
  3. Analyze the performance of the code using different metrics (accuracy, precision, recall, etc.) and briefly discuss your insights about the performance of the approach (good or bad) and the merits of using one metric over the other. You can use plots and other mechanisms to support your conclusions.

- **Problem 2 [33 Points]:** In this question, you will be using k-means to perform image compression. Implement a naïve version of the k-means algorithm based on your understanding. Your code must take the number of clusters k as input and perform k-means clustering on the given image (test_image.png). Once the algorithm finishes running, the cluster centroids represent the top-k common colors in the image. Iterate through each pixel in the image and assign the closest color to each pixel. Save and visualize the resulting image. For reading and writing images, you can use OpenCV, which is an open-source computer vision toolkit. The following code will load the image into a NumPy array. You can use this as input to your K-Means algorithm.

```
import cv2
import numpy as np

img = cv2.imread('test_image.png')
height, width, channels = np.shape(img)
for i in range(width):
    for j in range(height):
        pixel = img[j, i]  # Read the pixel at location (i,j)
        img[j, i] = newValue  # Assign a new value to the pixel
```

  Experiment with different values of k and briefly describe your thoughts about which value works best for this problem. You can use plots, error bars, etc. to support your conclusions.

- **Problem 3 [34 Points]:** Suppose that you are conducting a scientific experiment where you are observing the effects of one variable (`x_train.npy` and `x_test.npy`) on the output (`y_train.npy` and `y_test.npy`). On visualizing the relationship between the variables, you see the plot in Figure 1.

  Your goal is to come up with a linear regression model that can take the training data (`x_train.npy` and `y_train.npy`) and model the relationship between the variables x and y. You should implement your own version of linear regression using gradient descent. Things to consider:

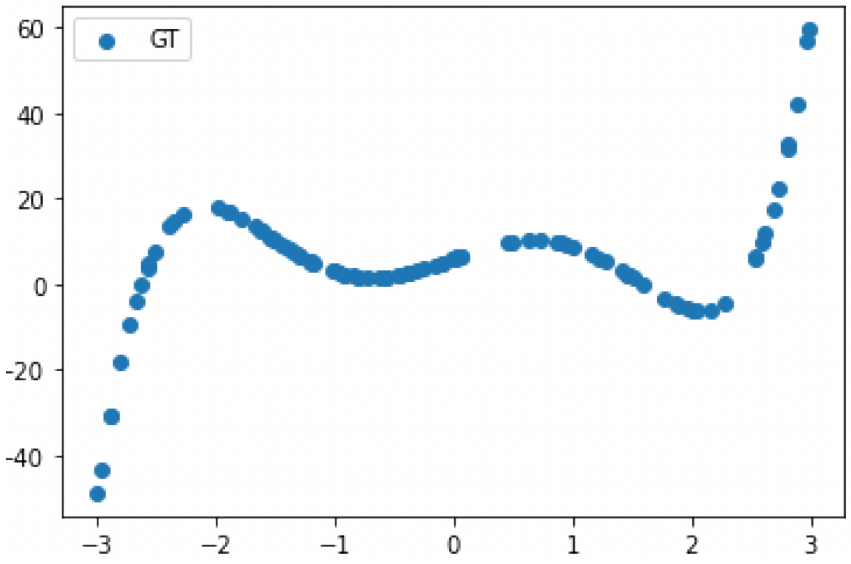

Figure 1: Plot for Problem 3

1. Try to plot this relationship on your own using `matplotlib`. You can also visualize the test data (`x_test.npy` and `y_test.npy`) to see if it gives you any clues about the underlying relationship between the variables.
2. Use your knowledge gleaned from the previous step to answer the following questions:
   a. Is the relationship linear?
   b. Do you need feature engineering to add any non-linearity?
   c. If so, how can you engineer these features?
   d. What are some functions that you can try?
   e. Plot each of them individually to verify!

You will need to write a short report detailing your thought process, the code you wrote in Python to implement the linear regression model and the equation that models the relationship between x and y that you found. You should provide evidence that corroborates your final statement such as plots, prediction errors, etc.

Deliverables: A single IPython Notebook that contains your code and report. You can use the text cells to write your report and embed any plots, illustrations, and/or images that you need to support your claims.