MACM 316 – Assignment 6

Due Date: July 14th, at 11pm.

You must upload both your code and your report. The assignment is due at 11:00pm. I have set the due time in Canvas to 11:05pm and if Canvas indicates that you submitted late, you will be given 0 on the assignment. Your computing report must be exactly 1 page. There will be a penalty given if your report is longer than one page.

- Please read the **Guidelines for Assignments** first.
- Keep in mind that Canvas discussions are open forums.
- Acknowledge any collaborations and assistance from colleagues/TAs/instructor.

A. Computing Assignment – Machine Learning

Required submission: 1 page PDF document and scripts/codes uploaded to Canvas.

The purpose of this assignment is to teach your computer to distinguish between red and blue points in 2D. This is an example of linear discriminant analysis. For further information about this topic, look at the book "Elements of Statistical Learning" by Hastie et. al which is available online for free.

In class we have spent considerable time on polynomial interpolation where the interpolating polynomial exactly equals the underlying data at each of the nodes. In the case where the data is noisy or the system is overdetermined, the method of least squares is used instead. Here is a basic reference on least squares data fitting: *https* : *//en.wikipedia.org/wiki/Least_squares*.

You are recommended to perform the following tasks in the given order and generate the required plots or numbers as you go.

Make sure that you download the associated data file called *dataset.mat* which contains two long matrices *training_set* and *test_set*. Keep these files in the same folder as your script for this assignment. The idea is to train your computer using the training set and then use the test set to see how well it has learned its lesson.

Description of the Data Sets

1. Each dataset has three columns.
2. The training set has 2000 rows (i.e. 2000 red and blue points) and the test set has 400 points.
3. Let $\mathbf{X}$ = training_set(:, [1 2]) (i.e. the first two columns of the training set) and let $\mathbf{y}$ = training_set(:, [3]).
4. Each row of the matrix $\mathbf{X}$ is the location of a point in 2D where the first column is the x-coordinate and the second column is the y-coordinate.
5. Each entry in $\mathbf{y}$ corresponds to a row of $\mathbf{X}$ and is either 0 or 1. If the value is 1 then we say that point is blue and if it is 0 then we say it is red.

Use MATLAB's *load* command to import both the training and test datasets. Next assemble the $\mathbf{X}$ and $\mathbf{y}$ matrices according to the above description and use MATLAB's *plot* command to generates a plot of the *training_set* in 2D. Recall that the $\mathbf{X}$ matrix contains the location of the points while $\mathbf{y}$ contains their color. Run your script to reproduce the following plot.

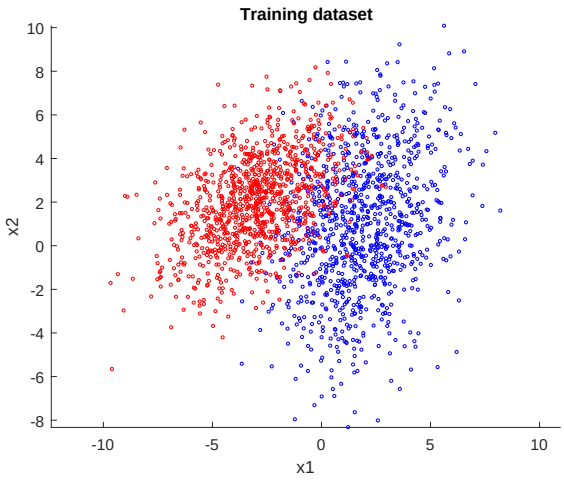

Figure 1: Visualizing the set of blue and red points in the training dataset.

The Training of the computer

1. Now define a matrix $\mathbf{A}$ so that $\mathbf{A} = [\text{ones}(2000, 1), \mathbf{X}]$.
2. Note that $\mathbf{A}$ is a 2000×3 matrix and $\mathbf{y}$ is a 2000×1 vector (check using the *size* function in MATLAB).
3. The objective is to find a 3×1 vector $\boldsymbol{\beta}$, so that $\mathbf{A}\boldsymbol{\beta} = \mathbf{y}$.
4. A unique solution is not possible since this system is overdetermined.
5. Therfore, find an approximate solution $\hat{\boldsymbol{\beta}}$ so that $\text{RSS}(\hat{\boldsymbol{\beta}}) := \|\mathbf{y} - \mathbf{A}\hat{\boldsymbol{\beta}}\|_2^2$ has the minimum value. To do this, compute the vector $\hat{\boldsymbol{\beta}}$ that solves the normal equations

   $$(\mathbf{A}^T\mathbf{A})\hat{\boldsymbol{\beta}} = \mathbf{A}^T\mathbf{y},$$

   and **report the value of $\text{RSS}(\hat{\boldsymbol{\beta}})$ in your report**.
6. The RSS stands for *Residual Sum of Squares*. It gives you an idea of how good $\hat{\boldsymbol{\beta}}$ is at predicting a value in $\mathbf{y}$ given the corresponding row in $\mathbf{A}$.

Next, generate a new plot of the training data (Figure 1) along with a black line given by the following equation

$$\hat{\beta}_1 + \hat{\beta}_2 x_1 + \hat{\beta}_3 x_2 = 1/2, \quad \text{where} \quad \hat{\boldsymbol{\beta}} = [\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3]^T. \tag{0.1}$$

Explain, just by looking at the figure, what this line is saying about the red and blue points. Include the figure and your explanation in your report.

The Testing of how well the computer learned

1. Use $\hat{\boldsymbol{\beta}}$ to predict whether a certain point is red or blue from the test dataset.

2. Let $\mathbf{z} = \text{test\_set}(:, [3])$ which is a vector of size 400×1 and $\mathbf{B} = [\text{ones}(400,1), \text{test\_set}(:,[1\ 2])]$ which is a 400×3 matrix similar to $\mathbf{A}$. Now modify the script to compute $\mathbf{v} = \mathbf{B}\hat{\boldsymbol{\beta}}$ and evaluate the vector $\hat{\mathbf{z}}$ so that

$$\hat{z}_j = \begin{cases} +1 & \text{if } v_j \geq 1/2 \\ 0 & \text{if } v_j < 1/2 \end{cases} \quad \text{for } j = 1, \cdots, 400.$$

3. In MATLAB, output $\hat{\mathbf{z}}$ and $\mathbf{z}$ side by side (do not include in your report). How is $\hat{\mathbf{z}}$ predicting the values in $\mathbf{z}$?

4. Compute the error of the prediction using

$$\text{Err}(\hat{\beta}) = \frac{1}{400} \sum_{j=1}^{400} |z_j - \hat{z}_j|.$$

Report this value in your report. Also, generate a plot of the test dataset where the points are colored according to the prediction $\hat{\mathbf{z}}$ along with the discriminant line defined in equation (0.1) above.