Homework Assignment 3: Logistic Regression

Due Monday, February 6th, 2023 at 11:59pm

Description

In class, we learned logistic regression and how to solve for model parameters using libraries for advanced optimization. In this problem set, you will implement such approaches and evaluate it on data.

What to submit

Download and unzip the template ps3_template.zip. Rename it to ps3_xxxx_LastName_FirstName, and add in your solutions:

1) Ps3_matlab_LastName_FirstName if you are programming in MATLAB OR
2) Ps3_python_LastName_FirstName if you are programming in Python

ps3_xxxx_LastName_FirstName /

- input/ - input data, images, videos or other data supplied with the problem set
- output/ - directory containing output images and other generated files
- ps3.m or ps3.py - your Matlab/Python code for this problem set
- ps3_report.pdf - A PDF file that shows all your output for the problem set, including images labeled appropriately (by filename, e.g. ps0-1-a-1.png) so it is clear which section they are for and the small number of written responses necessary to answer some of the questions (as indicated). Also, for each main section, if it is not obvious how to run your code please provide brief but clear instructions (no need to include your entire code in the report).
- *.m or *.py - Any other supporting files, including Matlab function files, Python modules, etc.

Zip it as `ps3_xxxx_LastName_FirstName.zip`, and submit on canvas.

Guidelines

1. Include all the required images in the report to avoid penalty.
2. Include all the textual responses, outputs and data structure values (if asked) in the report.
3. Make sure you submit the correct (and working) version of the code.

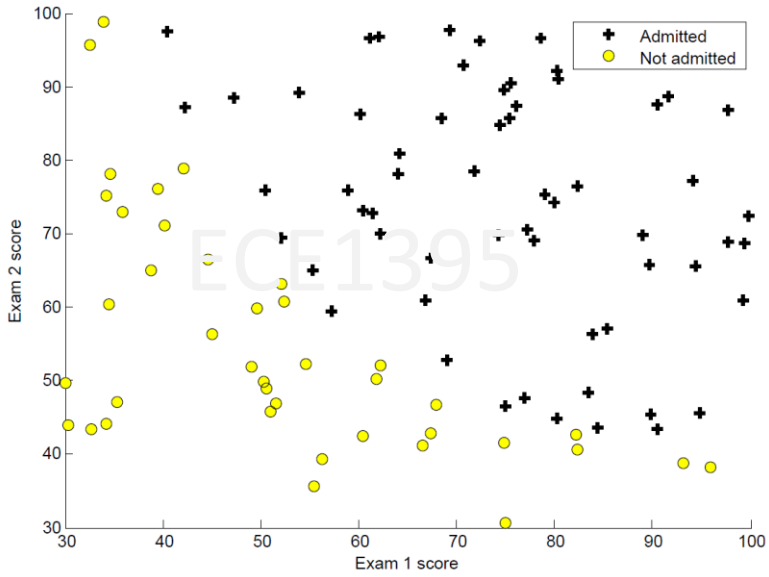
4. Include your name and ID on the report.
5. Comment your code appropriately.
6. Please avoid late submission. Late submission is not acceptable.
7. Plagiarism is prohibited as outlined in the Pitt Guidelines on Academic Integrity.

Questions

1- **Logistic regression**: In this part, you will build a logistic regression model to predict whether a student gets admitted into a university based on their results on two exams. You have historical data from previous applicants that you can use as a training set for logistic regression. For each training example, you have the applicant's scores on two exams and the admissions decision (0 = not admitted, and 1 = admitted). Load the training data from 'hw3_data1.txt' into MATLAB and answer/implement the following question toward building your classifier.

a. Define the feature matrix X, where each row corresponds to one feature example, and the labels vector y. Do not forget to append 1 for each feature vector, which will correspond to the bias (θ_0) that our model learns.
Text output: the size of the feature matrix X and the size of the label vector y.

b. Plot the training date to visualize the problem. Your output should look similar (but might not be exactly the same, e.g., you can use different data markers) to the figure below, where the axes are the two exam scores, and the positive and negative examples are shown with different markers.
Output: Scatter plot of training data as ps3-1-b.png



c. **Randomly** divide the data (from part a) into a training and test sets using approximately 90% for training. After this you should have these variables in your workspace: X_train and y_train as your training dataset, and X_test and y_test as your testing dataset.

d. Write a function, `g = sigmoid(z)` that computes the sigmoid function $g(z) = \frac{1}{1+e^{-z}}$. Test your function by calling `gz = sigmoid(z);` in your main script where `z = [-15:15]`. Plot gz versus z.
Function file: `sigmoid.m` containing function `sigmoid` (identical name)
Output: Plot gz versus z as ps3-1-c.png
Text response: inspect your figure, at what value of this does the output reaches 0.9?

e. Now you will implement the cost function and gradient for logistic regression. Recall that the cost function in logistic regression is

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right],$$

and the gradient of the cost is a vector of the same length as θ where the j^{th} element (for $j = 0, 1, \ldots, n$) is defined as follows

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Note that while this gradient looks identical to the linear regression gradient, the formula is actually different because linear and logistic regression have different definitions of $h_\theta(x)$. For logistic regression, $h_\theta(x) = g(\theta^T x)$.

[**MATLAB Programmers**] Write the function `[J, grad] = costFunction(theta, X_train, y_train)` where `J` contains the value of the cost function computed for a given parameter vector `theta`, and `grad` is the partial derivative of the cost w.r.t each parameter in `theta`.

[**Python Programmers**] Write two functions `costFunction(theta, X_train, y_train)` and `gradFunction(theta, X_train, y_train)` to compute the cost function and the gradient of the cost function, respectively.

[**All students**] Consider the toy dataset (x_1, x_2, y): $(1,0,0), (1,3,1), (3,1,0), (3,4,1)$, test your function(s) when $\theta = [0, 0, 0]^T$.
Function file: `costFunction.m` (and `gradFunction.py`) containing function `costFunction` (identical name).
Text output: the value of the cost J for the toy set when $\theta = [0, 0, 0]^T$.

f. For logistic regression, you want to optimize the cost function $J(\theta)$ with parameters θ. To do so, you are going to use a library for an optimization solver. MATLAB has a built-in fminunc function. fminunc is an optimization solver that finds the minimum of an unconstrained function. In Python, you can use, scipy.optimize.fmin_bfgs (link) for the same purpose. In this part we will use a linear

model , $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$. Concretely, you are going to use fminunc (or fmin_bfgs, if using python) to find the best parameters $\theta = [\theta_0, \theta_1, \theta_2]^T$ for the logistic regression cost function, given a training dataset (of X_train and y_train values). [**MATLAB Programmers**] Use the following options line and **use zeros as initial value of theta**.

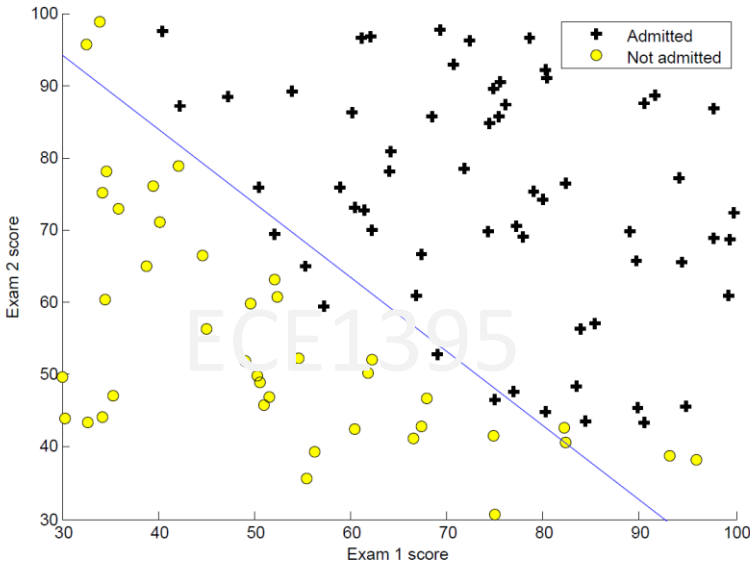
```
%  Set options for fminunc
options = optimset('GradObj', 'on', 'MaxIter', 400);

%  Run fminunc to obtain the optimal theta
%  This function will return theta and the cost
[theta, cost] = ...
    fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);
```

[**Python programmers**] Pass your cost and gradient functions to the fmin_bfgs function, and **use zeros as initial value of theta**. The training data should be passed as a tuple using the args input. Leave all other options at default. If it's taking too long to converge, set the input maxiter to 400.

Hint: Please verify the dimensions and data type of all of your variables before calling the optimization function.

Text output: the optimal parameters θ.
Text output: the value of the cost function at convergence.

g. Once you have the optimal θ, you can plot the decision boundary. With the help of the line function in Matlab/Python (or any other technique of your choice), generate a figure that contains the training samples and the decision boundary. Your figure should be similar (but not identical) to this one.
Output: save the figure as ps3-1-f.png



h. Using the testing data set, compute the accuracy of your logistic regression model. $Accuracy = \frac{Number\ of\ correctly\ classified\ samples}{Total\ number\ of\ testing\ samples}$

i. Use your model, compute the admission probability of a student whose scores are as following: test1 = 80 and test2 = 50.
Text output: the admission probability
Text output: what should be the admission decision? (Admitted or not admitted)

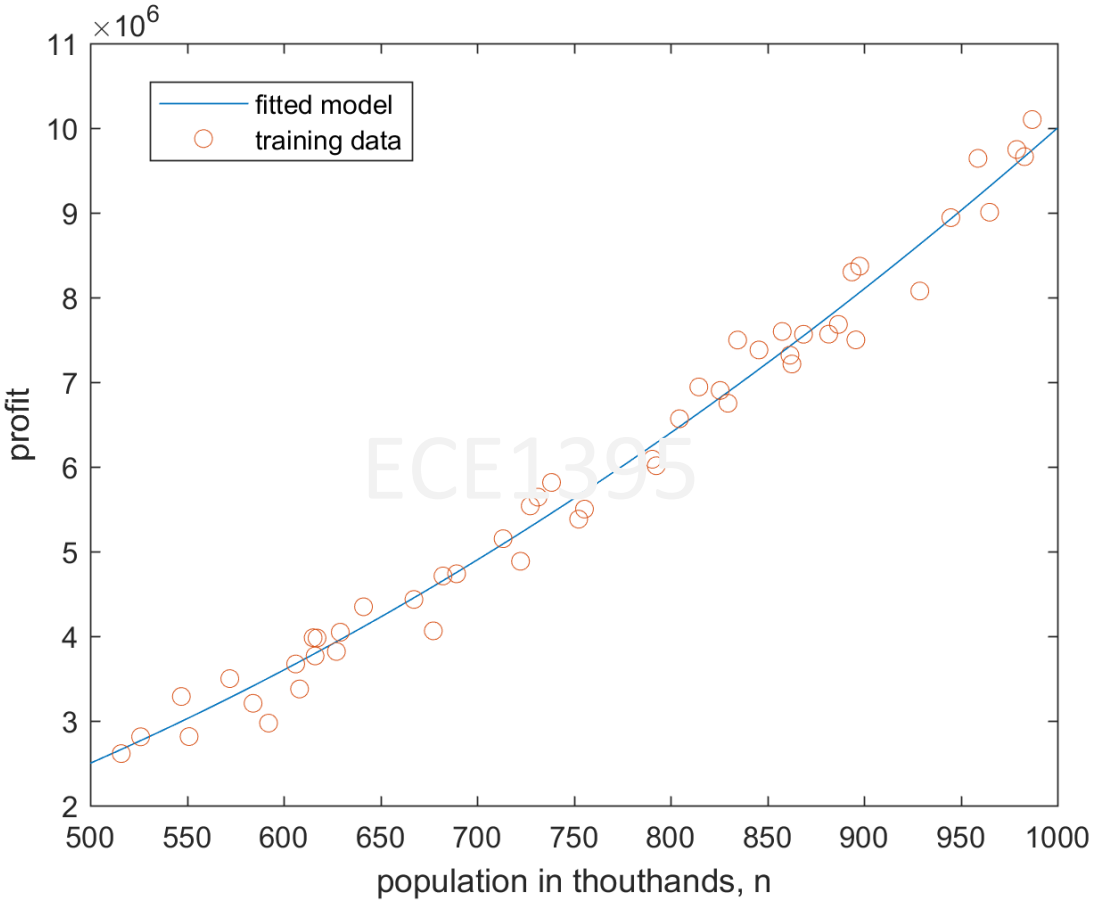
j. **BONUS** – in part e, you were provided with an expression for the cost function and its derivative. Analytically find the derivative of the cost function, and prove that it is equivalent to the provided expression. (This is an optional question, you can submit it for extra bonus points)

2- **Non-linear fitting**: In this problem, you are going to use linear regression to fit a non-linear function. The data in the file 'hw3_data2.csv' contains a training set of a utility company profit in 50 cities. The first column is the city population (n_p) in 1,000 vs the profit in the second column. The profit is assumed to follow a quadratic function in the population size:

$$profit = \theta_0 + \theta_1 n_p + \theta_2 n_p^2$$

a. Formulate a linear regression problem to estimate the profit. Use the normal equation solution of linear regression to find the model parameters $[\theta_0, \theta_1, \theta_2]$.
Text output: the learned model parameters

b. Plot your data and your model on one figure. Your figure should look similar, but not identical to the figure below.
Output: save the figure as ps3-2-b.png


×10^6
11
10
9
8
7
6
5
4
3
2
profit
fitted model
training data
500
550
600
650
700
750
800
850
900
950
1000
population in thouthands, n
ECE1395