Homework Assignment 2: Linear Regression

Due Sunday, September 17th, 2023 at 11:59pm

Description

In class, we discussed linear regression and how to solve for model parameters using gradient descent and normal equation. In this problem set, you will implement such approaches and evaluate it on data.

What to submit

Create your working folder ps2_xxxx_LastName_FirstName.

1) ps2_matlab_LastName_FirstName if you are programming in MATLAB OR
2) ps2_python_LastName_FirstName if you are programming in Python

Your folder should be structured as follows:

ps2_xxxx_LastName_FirstName/

- input/ - input data, images, videos or other data supplied with the problem set . **Please save the datasets in the input directory**
- output/ - directory containing output images and other generated files
- ps2.m or ps2.py - your Matlab/python code for this problem set
- ps2_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 `ps2_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.
7. Plagiarism is prohibited as outlined in the syllabus and in Pitt Guidelines on Academic Integrity.

Questions

1- **Cost function**: As you perform gradient descent to learn minimize the cost function $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$ $(you\ can\ also\ use\ the\ vectorized\ form\ defined\ in\ the\ class)$, it is helpful to monitor the convergence by computing the cost. Write a function, `function J = computeCost(X, y, theta)` that computes the cost given an estimate of the parameter vector θ. As you are doing this, remember that the variables X and y are not scalar values, but matrices whose rows represent the examples from the training set. Your function should be robust to any number of features. You can assume that the bias feature is already added to the data.

Consider this toy data set $\left(x_1^{(i)}, y^{(i)}\right)$: $(0,4), (2,8), (3,10), (4,12)$. Test your function for two different estimates of theta: (i) `theta = [0 0.5]'`, and (ii) `theta = [1 1]'`. Manually compute the cost and compare to your function output. (Hint: Do not forget to add the feature x_0 before calling your function).

Text Output: the cost for your the two test cases

Function file: computeCost.m containing the function `computeCost` (identical names) or an equivalent *.py file.

2- **Gradient descent**: write a function, `[theta, cost] = gradientDescent(X_train, y_train, alpha, iters)` that computes the gradient descent solution to linear regression.

inputs:

- `X_train` is an $m \times (n+1)$ feature matrix with m samples and n feature dimensions (don't forget to add the feature x_0 before calling your function). m is the number of samples in the training set.
- `y_train` is an mx1 vector containing the output for the training set. The i-th sample in `y_train` should correspond to the i-th row in `X_train`
- `alpha`, the learning rate to use in the weight update.
- `iters`, the number of iterations to run gradient descent for.

outputs:

- `theta` is a $(n+1) \times 1$ vector of weights (one per feature dimension).
- `cost` is a `iters`×1 vector of cost values (one per each iteration).

Your function should use a RANDOM initialization for theta, and then update theta `iters` times using the gradient descent algorithm we studied in the class. Of course, we could modify the function to exit if

the solution converge before reaching the maximum number of iterations (`iters`), but we are not implementing this option in your function

Test your function using the toy dataset in question 1; use alpha = 0.001, and iter = 15.

Text Output: your estimate of theta and the associated cost after 15 iterations.

Function file: gradientDescent.m containing the function `gradientDescent` (identical names) or an equivalent *.py file.

3- **Normal equation**: write a function, `[theta] = normalEqn(X_train, y_train)` that computes the closed-form solution to linear regression using normal equation. The inputs and outputs are as defined in question 2.

Test your function using the toy dataset in question 1.

Text Output: your estimate of theta. Is there a significant difference between your estimates in 2 and 3? If yes, why do you see that difference? What do you need to do such that the two approaches give almost the same result?

Function file: normalEqn.m containing the function `normalEqn` (identical names) or an equivalent *.py file.

4- **Linear regression with one variable**: The data in the file 'hw2_data1.csv' contains the prices of automobiles (**in $1,000s**) vs the horse power of each car (**in 100s hp**). We want to build an app to get an estimate of the expected car price (based on the car's horse power) to give our customer an idea about the price before heading a dealership.

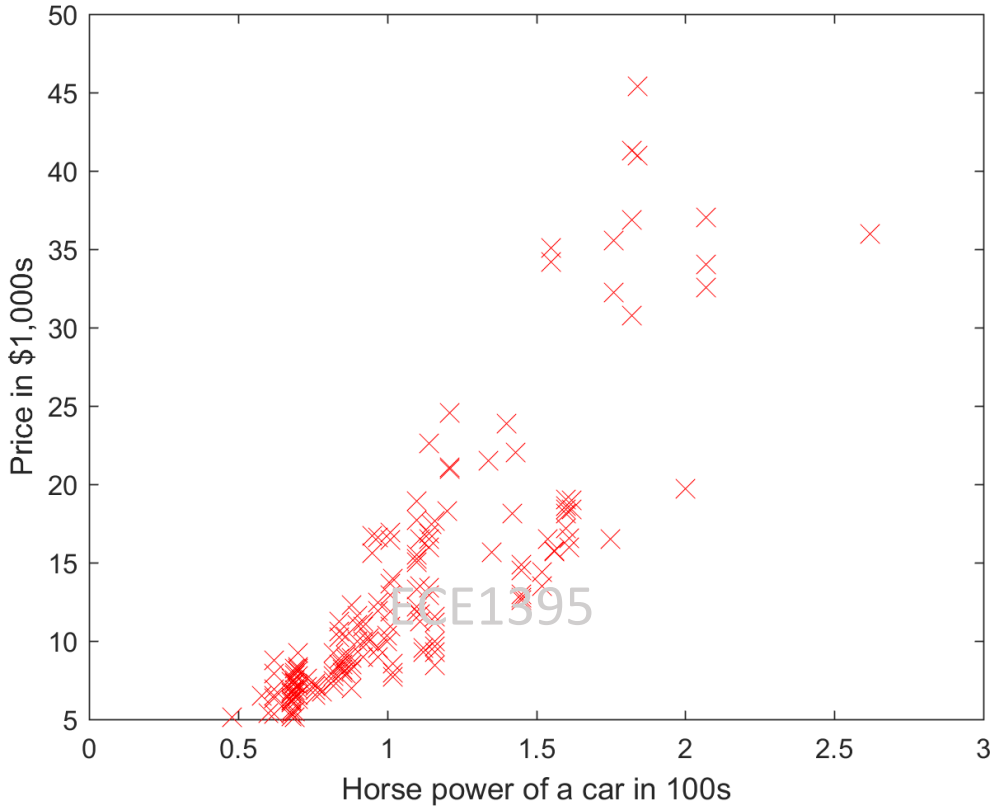
a. Load this data in MATLAB/Python. The first column is the horse power of an automobile, and the second column is the price of that car.

b. Plot the date to visualize the problem. Your output should look like the figure below.
Output: store the scatter plot of the data as ps2-4-b.png

c. We store each example as a row in the feature matrix X. To take into account the intercept term (θ_0), we add an additional first column to X and set it to all ones. This allows us to treat x_0 as simply another `feature'. Define X and according to the description in this part and part a.
Text output: the size of the feature matrix X and the size of the label vector y.

d. **Randomly** divide the data into a training and test set using approximately 90% for training. After this step, you should have these variables in your workspace: X_train and the corresponding

y_train as your training dataset, and X_test and the corresponding y_test as your testing dataset.



e. Use your training set, a learning rate of 0.3, and 500 iterations to compute the gradient descent solution of the model parameters θ. Plot the vector cost that shows the cost function for each iteration. In addition, now since you have the model parameters, you can plot your model (i.e., the line corresponding to your hypothesis). On the figure you generated on part b, plot the line corresponding to your hypothesis, i.e., $\theta_0 + \theta_1 x$.

Output: a plot of cost vs iteration# saved as ps2-4-e-1.png
Output: a figure that shows the data points along with the line representing your plot saved as ps2-4-e-2.png. Please don't forget to add a figure legend/key.
Text output: the computed model parameters θ.

f. Use the obtained model parameters from e to make predictions on profits using your testing set X_test, (i.e., $y_{pred} = h(X_{test})$). Compute the average mean squared error (cost) between the predicted vector y_pred and the ground-truth vector y_test.
Text output: your prediction error.

g. Use the normalEqn function and your training dataset to learn the model parameters θ, make predictions on profits using your testing set X_test. Compute the average mean squared error (cost) between the predicted vector y_pred and the ground-truth vector y_test.
Text output: your prediction error, compare these predictions to the one you obtained in f. Any comments?

h. In this part we want to study the effect of the learning rate. Use 300 iterations, solve for theta using the following values for the learning rate $\alpha = [0.001\ 0.003\ 0.03\ 3]$. This means that you have to run your function 4 times. In each time you would get a different theta and cost vectors. Plot the cost vs iteration# for each alpha on a separate figure. Use legend to identify different lines.
Output: Four figures showing the progression of cost vs iteration# for 4 different values of alpha, ps2-4-h-1.png through ps2-4-h-4.png
Text output: comment on the figures.

5- **Linear regression with multiple variables**: The data in the file 'hw2_data2.txt' contains a training set of housing prices in one city. The first column is the size of the house (in square feet), the second column is the number of bedrooms, and the third column is the price of the house.

a. Load the data into your workspace. Then standardize the data, by computing the mean and standard deviation for each feature dimension, then subtracting the mean and dividing by the stdev for each feature and each sample. Append a 1 for each feature vector, which will correspond to the bias (θ_0) that our model learns.
Text output:
   - mean and standard deviation of each vector
   - the size of the feature matrix X and the size of the label vector y.

b. Use a learning rate of 0.01 and 750 iterations, compute the gradient descent solution of the model parameters θ. Plot the vector cost that shows the cost function for each iteration.
Output: a plot of cost vs iteration# saved as ps2-5-b.png
Text output: the computed model parameters θ.

c. Predict the price of houses with 1080 square feet and 2 bedrooms. Note that you need to normalize this input feature vector using the mean and standard deviation from a.
Text output: your predictions