CS1675: Homework 3

Due: 9/27/2018, 11:59pm

This assignment is worth 50 points.

In this assignment, you will solve a regression problem in two ways: using the closed-form least-squares solution, and using gradient descent.

Part I: Linear regression using closed-form solution (10 points)

[5 pts] Write a function `[w] = lr_solve_closed(X_train, y_train)` that computes the closed-form least-squares solution to linear regression, using the Moore-Penrose inverse, as derived in class. Use the Matlab function `pinv`. The body of this function only requires one line of code.

Inputs:

- `X_train` is an NxD feature matrix with N samples and D feature dimensions. N is the number of samples in the training set.
- `y_train` is an Nx1 vector containing the labels for the training set. The i-th sample in `y_train` should correspond to the i-th row in `X_train`.

Outputs:

- `w` is a Dx1 vector of weights (one per feature dimension).

[5 pts] Also write a function `[y_pred] = lr_predict(X_test, w)` that uses the weights computed above, to predict a label for a new test sample.

Inputs:

- `X_test` is an MxD feature matrix with M samples and D feature dimensions. M is the number of samples in the test set.
- `w` is a Dx1 vector of weights.

Outputs:

- `y_pred` is the predicted Mx1 vector of labels for the test set.

Part II: Linear regression using gradient descent (20 points)

Now implement the gradient descent solution, in a function `[w] = lr_solve_gd(X_train, y_train, iters, eta)`.

Inputs: same as for `lr_solve_closed`, plus:

- `iters`, the number of iterations to run gradient descent for, and
- `eta`, the learning rate to use in the weight update.

Outputs: same as for `lr_solve_closed`.

Instructions:

1. [5 pts] First, initialize the weights in some way (use either random values or all zeros).
2. [10 pts] Second, repeat the following `iters` times. In each iteration, first compute the loss function gradient using all training data points. To do this, you need to use `lr_predict.m`.
3. [5 pts] Then, adjust the weights in the direction opposite to the gradient.

Part III: Testing on the Wine Quality dataset (20 points)

You will use the Wine Quality dataset. Use only the red wine data. The goal is to find the quality score of some wine based on its attributes. Include your code in a script `regression.m`. In a file `report.pdf` or `report.docx`, report the L2 error (described below) for the closed-form and gradient descent solutions.

1. [10 pts] First, download the `winequality-red.csv` file, load it in Matlab (e.g. using `dlmread`) and divide the data into a training and test set using approximately 50% for training. Standardize the data, by computing the mean and standard deviation for each feature dimension using the train set only, 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 that our model learns.
2. [5 pts] Find the direct closed-form solution and evaluate the accuracy on the test set, by computing the L2 distance between the predicted vector `y_pred` and the ground-truth vector `y_test`. Print the L2 error in your script, with an appropriate description for what is being printed; use `fprintf`. Include it in your report.
3. [5 pts] Now compute and evaluate the gradient descent solution. Use 50 iterations, and experiment with the following values for the learning rate: `10.^(-6:-1)`. Evaluate the L2 distance between predicted and ground-truth test labels as above. Print the errors for each learning rate and include them in your report.

Submission: Please include the following files:

- `lr_solve_closed.m`
- `lr_predict.m`
- `lr_solve_gd.m`
- `regression.m`
- `report.pdf/docx`