[25 points]

In this coding assignment you are to implement logistic regression classifier.

1. Read “corrupted_2class_iris_dataset.dat” and randomly shuffle the data.
   This is a subset of the corrupted iris dataset used in the previous assignment. It contains only the setosa and versicolor species (called class 1 and class 0, respectively).
2. Apply 10-fold cross validation for training and testing.
3. Use batch gradient descent to find the optimal weight vector.
   Experiment with different learning rates.
   Run it for 1500 iterations. You may change the number of iterations as you see fit.
4. Compute the cost function.
5. Classify test data.
6. Show the classification accuracy per iteration.
7. Show the average classification accuracy after the 10-fold CV is completed.
8. Plot the cost function as a function of training iterations.

Hints:

```
clear all; close all; clc;
data = dlmread('corrupted_2class_iris_dataset.dat');

N = 100;  % total number of samples
NC = 50;  % size of each class
K = 10;  % K-fold
d = 4;   % number of features
nu = 0.01;   % learning rate

% Randomly shuffle data
%seed = 150; rand('seed', seed);
index = randperm(N);
data_shuffled = data(index,:);


% 10-fold cross validation
for k=1:K

  % Separate training data and test data
  % 90% of the data for training, 10% for testing

  % TRAINING
  % Size X = 90x5, the first column = 1
  % Size y = 90x1 (class 1 and class 0 labels)
  % Size w = 5x1, initialized randomly, w1 is the bias

  % Apply Gradient Descent and run for 1500 iterations


  % TESTING

  % Compute sigm for each test data and assign label
  % Check against true response

end

% Evaluate classification accuracy
%   Accuracy per iteration = no of correct classification / 10
%   Average accuracy for all 10-fold CV
%   fprintf('Accuracy = %5.4f\n', ...) generates nice format

% Plot cost function vs training iterations
plot (J);    % Size J = 1500x1
xlabel('Training iterations');
ylabel('Cost function J');
```

Expected output:

```
Classification accuracy
ans =
    0.9000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    0.9000
    1.0000
    0.9000
    0.9000

Average accuracy = 0.9600
```