CS 434: Implementation assignment 1

Due Friday April 12th 11:59PM, 2019

General instructions

1. The assignment should be implemented in Python 3. You should make sure that your code can be run on the flip server.
2. You can work in team of up to 3 people. Each team will only need to submit one copy of the source code and report.
3. You need to submit your source code (self contained, well documented and with clear instruction for how to run) and a report via TEACH. In your submission, please clearly indicate your team members' information.
4. Be sure to answer all the questions in your report. Your report should be typed, submitted in the pdf format. You will be graded based on both your code as well as the report. In particular, the clarity and quality of the report will be worth 10 points. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.

1 Linear regression

Data You will use the Boston Housing dataset of the housing prices in Boston suburbs. The goal is to predict the median value of housing of an area (in thousands) based on 13 attributes describing the area (e.g., crime rate, accessibility etc). The file housing_desc.txt describes the data. Data is divided into two sets: (1) a training set housing_train.txt for learning, and (2) a testing set housing_test.txt for testing. Your task is to implement linear regression and explore some variations with it on this data.

1. (10 pts) Load the training data into the corresponding X and Y matrices, where X stores the features and Y stores the desired outputs. The rows of X and Y correspond to the examples and the columns of X correspond to the features. Introduce the dummy variable to X by adding an extra column of ones to X (You can make this extra column to be the first column. Changing the position of the added column will only change the order of the learned weight and does not matter in practice. Compute the optimal weight vector $\mathbf{w}$ using $\mathbf{w} = (X^TX)^{-1}X^TY$. Feel free to use existing numerical packages (e.g., numpy) to perform the computation. Report the learned weight vector.
2. (10 pts) Apply the learned weight vector to the training data and testing data respectively and compute for each case the average squared error(ASE), defined by $1/n\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, which is the sum of squared error normalized by the total number of examples in the data. Report the training and testing ASEs respectively. Which one is larger? Is it consistent with your expectation?

   Write your code so that you get the results for questions 1 and 2 using the following command:

   python q1_2.py housing_train.txt housing_test.txt

   The output should include:

   - the learned weight vector
   - ASE over the training data
   - ASE over the testing data

3. (10 pts) Remove the dummy variable (the column of ones) from X, repeat 1 and 2. How does this change influence the ASE on the training and testing data? Provide an explanation for this influence.

   Write your code so that you get the results for question 3 using the following command:

   python q1_3.py housing_train.txt housing_test.txt

   The output should include:

   - the learned weight vector
   - ASE over the training data
   - ASE over the testing data

4. (20 pts) Modify the data by adding additional random features. You will do this to both training and testing data. In particular, for each instance, generate d (consider $d = 2, 4, 6, ...10$, feel free to explore more values) random features, by sampling from a standard normal distribution. For each d value, apply linear regression to find the optimal weight vector and compute its resulting training and testing ASEs. Plot the training and testing ASEs as a function of d. What trends do you observe for training data and test data respectively? Do more features lead to better prediction performance at testing stage? Provide an explanation to your observations.

   Write your code so that you get the results for question 4 using the following command:

   python q1_4.py housing_train.txt housing_test.txt

   The output should include:

   - plot of the training ASE (y-axis) as a function of d (x-axis)
   - plot of the testing ASE (y-axis) as a function of d (x-axis)

2 Logistic regression with regularization

Data. For this part you will work with the USPS handwritten digit dataset and implement the logistic regression classifier to differentiate digit 4 from digit 9. Each example is an image of digit 4 or 9, with 16 by 16 pixels. Treating the gray-scale value of each pixel as a feature (between 0 and 255), each example has $16^2 = 256$ features. For each class, we have 700 training samples and 400 testing samples. You can view these images collectively at `http://www.cs.nyu.edu/~roweis/data/usps_4.jpg`, and`http://www.cs.nyu.edu/~roweis/data/usps_9.jpg`

The data is in the csv format and each row corresponds to a hand-written digit (the first 256 columns) and its label (last column, 0 for digit 4 and 1 for digit 9).

1. (20 pts) Implement the batch gradient descent algorithm to train a binary logistic regression classifier. The behavior of Gradient descent can be strongly influenced by the learning rate. Experiment with different learning rates, report your observation on the convergence behavior of the gradient descent algorithm. For your implementation, you will need to decide a stopping condition. You might use a fixed number of iterations, the change of the objective value (when it ceases to be significant) or the norm of the gradient (when it is smaller than a small threshold). Note, if you observe an overflow, then your learning rate is too big, so you need to try smaller (e.g., divide by 2 or 10) learning rates. Once you identify a suitable learning rate, rerun the training of the model from the beginning. For each gradient descent iteration, plot the training accuracy and the testing accuracy of your model as a function of the number of gradient descent iterations. What trend do you observe? Write your code so that you get the results for question 1 using the following command:

   python q2_1.py usps_train.txt usps_test.txt learningrate

   The output should include:

   - plot of the learning curve: training accuracy (y-axis) as a function of the number of gradient descent iterations (x-axis)

- plot of the learning curve: testing accuracy (y-axis) as a function of the number of gradient descent iterations (x-axis)

2. (10 pts) Logistic regression is typically used with regularization. Here we will explore L_2 regularization, which adds to the logistic regression objective an additional regularization term of the squared Euclidean norm of the weight vector.

$$L(\mathbf{w}) = \sum_{i=1}^{n} l(g(\mathbf{w}^T\mathbf{x}^i), y^i) + \frac{1}{2}\lambda|\mathbf{w}|^2$$

   where the loss function l is the same as introduced in class. Find the gradient for this objective function and modify the batch gradient descent algorithm with this new gradient. Provide the pseudo code for your modified algorithm.

3. (25 pts) Implement your derived algorithm, and experiment with different λ values (e.g., $10^{-3}, 10^{-2}, ..., 10^{3}$). Report the training and testing accuracies (i.e., the percentage of correct predictions) achieved by the weight vectors learned with different λ values. Discuss your results in terms of the relationship between training/testing performance and the λ values. Write your code so that you get the results for question 3 using the following command:

   python q2_3.py usps_train.txt usps_test.txt lambdas

   where *lambdas* contains the list of λ values to be tested. The output should include:

   - plot of the training accuracy (y-axis) as a function of the λ value (x-axis)
   - plot of the testing accuracy (y-axis) as a function of the λ value (x-axis)

Remark 1 *For logistic regression, it would be a good idea to normalize the features to the range of* $[0, 1]$*. This will makes it easier to find a proper learning rate. You can find information about feature normalization at* `https://en.wikipedia.org/wiki/Feature_scaling`*)*