Homework on Linear and Logistic Regressions

In this homework you will implement *Linear Regression* and *Logistic Regression* algorithms as you have learned in class. The exercise is based on the programming exercises for the Stanford Unsupervised Feature Learning and Deep Learning Tutorial. The original starter code for the exercises can be found in the GitHub Repo. in this link under the ex1/ directory.

Guidelines

- The solution for this homework is to be posted as a .pdf file.
- The implementation will be done with MATLAB/Octave.
- Extract `Combined_HW_Linear_and_Logistic_Reg.zip`. The zip contains exercises 1 and 2 on Linear and Logistic Regressions, as well as an auxiliary folder called `common`.

1 Linear Regression

In this exercise we wish to address a regression problem of predicting the price of a house based on given features describing the house (such as its size, number of bedrooms, etc.).

- Open `HWonLinearRegression` folder which contains 3 files:
  1. `ex1a_linreg.m` - the main script. It contains the makings of a simple linear regression experiment. You don't need to edit this file as it already performs the setup steps for you, but make sure you understand it.
  2. `housing.data` - the prepared dataset from which the data will be loaded in the main script.
  3. `linear_regression.m` - here you will implement your code.

  Since $x_0^{(i)} = 1$ is defined for convenience as it was explained in class, an extra '1' feature is added to the dataset so that θ_0 will act as an intercept term in the linear function. The examples in the dataset are randomly shuffled and the data is then split into a training and testing set. There are 14 house features per example that are used as input to the learning algorithm and are stored in the variables `train.X` and `test.X`. The target value to be predicted is the estimated house price for each example. The prices are stored in `train.y` and `test.y`, respectively, for the training and testing examples.

  Our goal is to use the training set to find the best choice of θ for predicting the house prices and then check its performance on the testing set.

- Open `linear_regression.m`. This function receives the training data `X`, the training target values (house prices) y, and the current features parameters θ. Here you should implement the code for calculating the objective function $C(\theta)$ and its gradient with respect to the features parameters $\nabla_\theta C(\theta)$ for the linear regression. Store the computed values in the variables f and g respectively. Make sure you write your code efficiently without looping, i.e. use efficient matrix-wise calculations.

  The main script calls the `minFunc` optimization package with $C(\theta)$ and $\nabla_\theta C(\theta)$. `minFunc` will attempt to find the best choice of θ by minimizing $C(\theta)$ you implemented in `linear_regression.m` based on known algorithms such as those we studied in class.

  After `minFunc` completes (i.e., after training is finished), the training and testing error is printed out. Optionally, it will plot a quick visualization of the predicted and actual prices for the examples in the

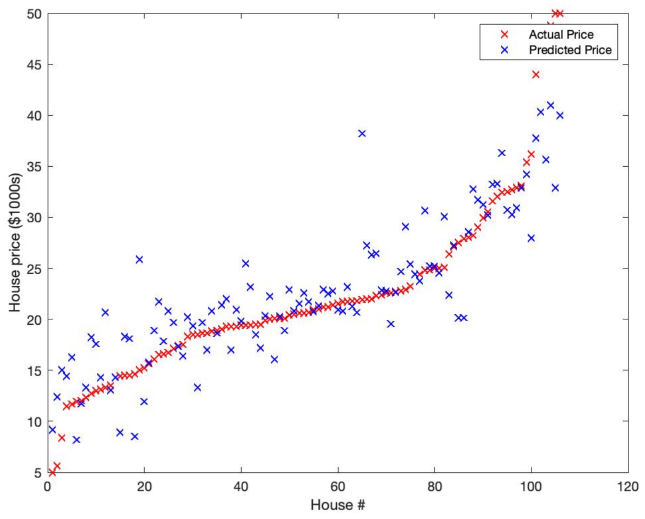

Figure 1: Predicted house price.

test set. Once you complete the exercise successfully, the resulting plot should look something like in Fig. 1. Typical values for the RMS training and testing error are between 4.5 and 5.

- Make a .pdf file and add your `linear_regression.m` code and the output graph .

2 Logistic Regression

In this exercise we wish to classify images of digits from the MNIST dataset (discussed in homework 1 on KNN) as either "0" or "1".

- Open `HWonLogisticRegression` folder which contains the following:

  1. `ex1b_logreg.m` - the main script. You don't need to edit this file as it already performs the setup steps for you, but make sure you understand it.
  2. `MNIST` - the MNIST database and auxiliary functions for loading. Each of the digits is is represented by a 28x28 grid of pixel intensities, which we will reformat as a vector $x^{(i)}$ with 28x28 = 784 elements. The label is binary, so $y^{(i)} \in \{0, 1\}$.
  3. `ex1_load_mnist.m` - a function called by the main script to load the MNIST training and testing data. Even though the MNIST dataset contains 10 different digits (0-9), in this exercise we will only load the 0 and 1 digits. In addition to loading the pixel values into a matrix `X` (so that that j'th pixel of the i'th example is $X_{ji} = x_j^{(i)}$) and the labels into a row-vector `y`, `ex1_load_mnist.m` will also perform some simple normalizations of the pixel intensities so that they tend to have zero mean and unit variance.
  4. `logistic_regression.m` - here you will implement your code.
  5. `sigmoid.m` - calculates the sigmoid function. You may call it from `logistic_regression.m`.
  6. `binary_classifier_accuracy.m` - a function called from the main script to calculate the accuracy of the implemented logistic regression algorithm.

The main script `ex1b_logreg.m` also performs the following tasks for you:

1. The code will append a row of 1's so that θ_0 will act as an intercept term.

2. The code calls minFunc with the `logistic_regression.m` file as objective function. Your job will be to fill in `logistic_regression.m` to return the objective function value and its gradient.
3. After minFunc completes, the classification accuracy on the training set and test set will be printed out.

- Open `logistic_regression.m`. Here you should implement the code for calculating the objective function $C(\theta) \triangleq -l(\theta)$ studied in class and its gradient $\nabla_\theta C(\theta)$ for the logistic regression. Store the computed values in the variables f and g respectively. Make sure you write your code efficiently without looping, i.e. use efficient matrix-wise calculations. Once you have completed these tasks, you will be able to run the `ex1b_logreg.m` script to train the classifier and test it.

  If your code is functioning correctly, you should find that your classifier is able to achieve 100% accuracy on both the training and testing sets! It turns out that this is a relatively easy classification problem because 0 and 1 digits tend to look very different. In future exercises it will be much more difficult to get perfect results like this.

- Add to your .pdf file your `logistic_regression.m` code and submit the file in Moodle.