Linear Regression with Multiple Variables using Gradient Descent

Purpose: Gain insight into (a) basic machine learning concepts, and (b) how a simple regression problem can be solved using linear regression.

In this project we are required to implement a linear regression model in order to fit a plane using gradient descent. The dataset is stored in a comma-separated values (CSV) file and includes N examples (x_i, y_i). The data has already been split up into a training and test set. Each input vector $x_i = (x_{i1}, x_{i2}, \ldots x_{ip})^T$ is defined in terms of two features ($p = 2$) and will be used to predict the associated real-valued output y_i. Assuming the relationship between the input vector and output variable is linear, we can find a (hyper) plane that fits the data reasonably well using a linear regression model. This model, or hypothesis space, can be seen in Equation 1, where W is a weight vector and b is the bias term[1].

$$f(x_i) = f_{W,b}(x_i) = b + \sum_{j=1}^{p} W_j x_{ij} \qquad (1)$$

Now that we have a model we need something that allows us to evaluate the quality of the current hypothesis. The mean squared error (MSE) function is one such *loss*, or error, function that measures the average squared differences between model predictions $f(x_i)$ and target values y_i over all observations. The loss function can be seen in Equation 2, where n is the number of examples included in the computation.

$$L(W, b) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 \qquad (2)$$

Ideally, we want to pick the hypothesis that yield the lowest loss over the examples.
The most popular way of estimating this is called *least squares*. In this assignment you will perform least squares using an iterative optimization algorithm called gradient descent. In order to use gradient descent we need to find out how the loss function changes when W or b are allowed to vary. The results of these partial derivatives can be seen in Equation 3.

$$\frac{\partial L}{\partial W} = \frac{2}{n} \sum_{i=1}^{n} (f(x_i) - y_i) x_i \qquad \frac{\partial L}{\partial b} = \frac{2}{n} \sum_{i=1}^{n} (f(x_i) - y_i) \qquad (3)$$

With these partial derivatives we can specify the update rules used to alter the trainable parameters. This can be seen in Equation 4, where _ is the learning rate (decides how much the parameters should change). *The rules are repeated until convergence or until the maximum number of iterations has been reached.*

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W}$$
$$b \leftarrow b - \alpha \frac{\partial L}{\partial b} \qquad (4)$$

Implementing linear regression and procedures for optimizing it using gradient descent.
Training the model on the training dataset.

[1]There are several alternative ways to express the linear regression model. Another popular approach employed by Andrew Ng in CS 229 is to incorporate the bias, or intercept term, into the weight vector W, where this new vector is called Θ. In other words, $\Theta = (b, w_1, w_2, \ldots w_p)^T$. Consequently, an extra dimension of all ones must be prepended to x_i.