HW #1

DUE SEP 25 (ONLINE SUBMISSION VIA GRADESCOPE)

(1) For this problem, please use the dataset in the link below. Please use the first column as the sales data. Others are predictors. Use the first twenty entries to create a regression model using the code snippet included in the class discussion. Then use the remaining points to judge the performance of the predictor. The code snippet also an available resource with the book (Hands on ML, HML Book)
`https://college.cengage.com/mathematics/brase/understandable_statistics/7e/students/datasets/mlr/excel/mlr05.xls`

(2) Consider the perceptron in two dimensions: $h(x) = sign(w^T x)$ where $w = [w_0, w_1, w_2]^T$ and $x = [1, x_1, x_2]^T$. Technically, x has three coordinates, but we call this perceptron two-dimensional because the first coordinate is fixed at 1.
   (a) show that the regions on the plane where $h(x) = +1$ and $h(x) = -1$ are seperated by a line. If we express this line by the equation $x_2 = ax_1 + b$, what are the slope a and intercept b in terms of w_0, w_1, w_2?
   (b) Draw a picture for the cases $w = [1, 2, 3]^T$ and $w = -[1, 2, 3]^T$.

   In more that two dimensions, the $+1$ and -1 regions are separated by a hyper-plane, the generalization of a line.

(3) Consider a linearly separable data set $(x_1, y_1), \ldots, (x_m, y_m)$. Let $B = min\{||w|| : \forall i \in [m], y_i(w^T x_i) \geq 1\}$, and let $R = max_i ||x_i||$. Show that the Perceptron algorithm stops after at most $(RB)^2$ iterations, and after it stops all points are properly classified.

(4) The perceptron learning algorithm works like this: In each iteration t, pick a random $x(t), y(t)$ and compute the signal $s(t) = w^T(t)x(t)$. If $y(t).s(t) \leq 0$, update w by

$$w(t+1) \leftarrow w(t) + y(t).x(t);$$

   One may argue that this algorithm does not take the 'closeness' between $s(t)$ and $y(t)$ into consideration. Let's look at another perceptron learning algorithm: In each iteration, pick a random $(x(t), y(t))$ and compute $s(t)$. If $y(t).s(t) \leq 1$, update w by

$$w(t+1) \leftarrow w(t) + \eta.(y(t) - s(t)).x(t);$$

Where η is a constant. That is, if $s(t)$ agrees with $y(t)$ well (their product is > 1), the algorithm does nothing. One the other hand, if $s(t)$ is further from $y(t)$, the algorithm changes $w(t)$ more. In this problem, you are asked to implement this algorithm and study its performance.

(a) Genrate a training data set of size 100 similar to that used in Exercise 1.4. Generate a test data set of size 10,000 from the same process. To get g, run the algorithm above with $\eta = 100$ on the training data set, until a maximum of 1,000 updates has been reached. Plot the training data set, the target function f, and the final hypothesis g on the same figure. Report the error on the test set.
(b) Use the data set in (a) and redo everything with $\eta = 1$.
(c) Use the data set in (a) and redo everything with $\eta = 0.01$.
(d) Use the data set in (a) and redo everything with $\eta = 0.0001$.
(e) Compare the results that you get from (a) to (d).

The algorithm above is a variant of the so-called Adaline (*Adaptive Linear Neuron*) algorithm for perceptron learning.