Linear regression implementation

a) Read data:

train.txt, test.txt → Each column is a feature except last column, which is label

X → load feature columns in ~~matx~~ matrix X

t → load label column in t

In case of simple LR: $X = \begin{bmatrix} 3032 \\ 2078 \\ 2400 \\ \vdots \end{bmatrix}$, $t = \begin{bmatrix} 525k \\ 230k \\ 87k \\ \vdots \end{bmatrix}$

b) Standardize features:

mean → find mean of each feature (column) and store it in mean

std → find standard deviation of each feature (column) and store it in std

Standardize features in X by subtracting their mean and dividing by their standard deviation: $s = (X - mean)/std$

For simple LR: $X = \begin{bmatrix} 3032 \\ 2078 \\ 2400 \\ \vdots \end{bmatrix}$

lets say mean of 1st feature/column in X, mean = [2528]

" " standard deviation of 1st feature/column of X, std = [238]

Then, $s = (X - mean)/std = \begin{bmatrix} \frac{3032-2528}{238} \\ \frac{2078-2528}{238} \\ \frac{2400-2528}{238} \\ \vdots \end{bmatrix}$

c) Implement GD in 'train' function to compute w.

For ~~mt~~ simple LR: $w = [w_0 \;\; w_1]$.

Each epoch ~~update~~ of GD: $w = w - eta.grad$

where ~~grad~~ $grad = \left[\frac{\partial}{\partial w_0} J(w) \;\; \frac{\partial}{\partial w_1} J(w) \right]$ computed using 'compute_gradient'

Run GD for 200 epochs ~~we~~ with $eta = 0.1$

d) Use 'compute_gradient' function to compute derivatives of cost function ~~grad~~

$grad = \left[\frac{\partial}{\partial w_0} J(w) \;\; \frac{\partial}{\partial w_1} J(w) \right]$., ~~where grad co~~

For simple LR:

⇒ X should have bias features added as first column:

$$X = \begin{bmatrix} 1 & 3032 \\ 1 & 2078 \\ 1 & 2400 \\ \vdots & \vdots \end{bmatrix} \rightarrow \underline{50 \times 2}$$

~~⇒ $w^T X$ should be = $[w_0 \;\; w_1]$ *~~ $w^T.X$

$$\Rightarrow w^T.X \text{ should be} = [w_0 \;\; w_1] * \underbrace{\begin{bmatrix} 1 & 3032 \\ 1 & 2078 \\ \vdots & \vdots \end{bmatrix}}_{50 \times 2} = \underbrace{\begin{bmatrix} w_0 & w_1.3032 \\ w_0 & w_1.2078 \\ \vdots & \vdots \end{bmatrix}}_{50 \times 2}$$

⇒ Given $w^T.X$, obtain $[L - t]$ s.t.

$$L = \underbrace{\begin{bmatrix} w_0 + w_1.3032 \\ w_0 + w_1.2078 \\ w_0 + w_1.2400 \\ \vdots \end{bmatrix}}_{50 \times 1} \text{ and labels } t = \underbrace{\begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \end{bmatrix}}_{50 \times 1}$$

$$\therefore [L - t] = \begin{bmatrix} w_0 + w_1.3032 - t_1 \\ w_0 + w_1.2078 - t_2 \\ \vdots \end{bmatrix} \rightarrow \underline{50 \times 1}$$

CamScanner

$\Rightarrow$ Given ~~grad~~ $\frac{\partial}{\partial w_j} J(w) = \frac{1}{N} \sum_{i=1}^{N} (h_w(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$ for $j = 0, 1$

$\Rightarrow$ In vectorized form, grad $= [\frac{\partial}{\partial w_0} J(w) \ \ \frac{\partial}{\partial w_1} J(w)] = \frac{1}{N}(X.T \cdot [L-t]$

$\Rightarrow$ Here, $X.T = \begin{bmatrix} 1 & 1 & 1 & \dots \\ 3032 & 2078 & 2400 & \dots \end{bmatrix}$

$\underline{2 \times 50}$

$$\Rightarrow X.T \cdot [L-t] = \begin{bmatrix} 1 & 1 & \dots \\ 3032 & 2078 & \dots \end{bmatrix} \cdot \begin{bmatrix} w_0 + w_1 . 3032 - t_1 \\ w_0 + w_1 . 2078 - t_2 \\ \vdots \end{bmatrix}$$

$\underline{2 \times 50} \cdot \underline{50 \times 1}$

$$= \begin{bmatrix} 1(w_0 + w_1 . 3032 - t_1) + 1(w_0 + w_1 . 2078 - t_2) + \dots \\ 3032(w_0 + w_1 . 3032 - t_1) + 2078(w_0 + w_1 . 2078 - t_2) + \dots \end{bmatrix}$$

$\underline{2 \times 1}$

$\Rightarrow N$ = number of training examples

$\therefore$ Obtain and return vectorized form of grad in 'compute_gradient' function.

e) Compute cost function $J(w)$ in 'compute_cost' function.

$$\Rightarrow J(w) = \frac{1}{2N} \cdot \sum_{i=1}^{N} (h_w(x^{(i)}) - y^{(i)})^2$$

For simple LR:

$\Rightarrow$ ~~Find~~ obtain $[L-t] = \begin{bmatrix} w_0 + w_1 . 3032 - t_1 \\ w_0 + w_1 . 2078 - t_2 \\ \vdots \end{bmatrix} \rightarrow \underline{50 \times 1}$

~~$\Rightarrow$ Then find s = sum of all values in $[L-t]$. Find s^2.~~

~~$\therefore J(w)$ is given by $= \frac{1}{2N}$.~~

⇒ Then obtain s^2 = sum of squared values in [L-t]

⇒ $s^2 = (w_0 + w_1 . 3032 - t_1)^2 + (w_0 + w_1 . 2078 - t_2)^2 + \ldots$

⇒ Given s^2, obtain $J(w) = \frac{1}{2N} \cdot s^2$

⇒ Return $J(w)$

e. Use the 'compute_rmse' function to compute and return root mean square error of dataset

⇒ Compute and return RMSE as square root of $J(w)$ computed above.

f. In the main method, standardize the training and test data, Xtrain and Xtest. Then add a column of ones as the first column of training and test data. The column of ones are the bias features.

For simple LR: $Xtrain = \begin{bmatrix} 3032 \\ 2074 \\ \vdots \end{bmatrix}$, after adding bias features, $Xtrain = \begin{bmatrix} 1 & 3032 \\ 1 & 2074 \\ \vdots & \vdots \end{bmatrix}$

3. Implement the steps above for the multiple LR problem where there ~~am~~ are 3 data features corresponding to the first 3 columns and the labels " " " last column in the dataset.

4. BONUS [SGD]. Implement SGD ~~am~~ using the 'train_sgd' function to solve the simple and multiple LR problems. In each epoch of SGD:

⇒ w = w - eta.grad is computed for a single training example

⇒ Iterate ~~above~~ over ~~all~~ all training examples.