CSC411: Assignment 1

Due on Thursday, October 5th, 2017

Student Name: **Gokul K. Kaushik**

Student Number: **999878191**

Table of Contents

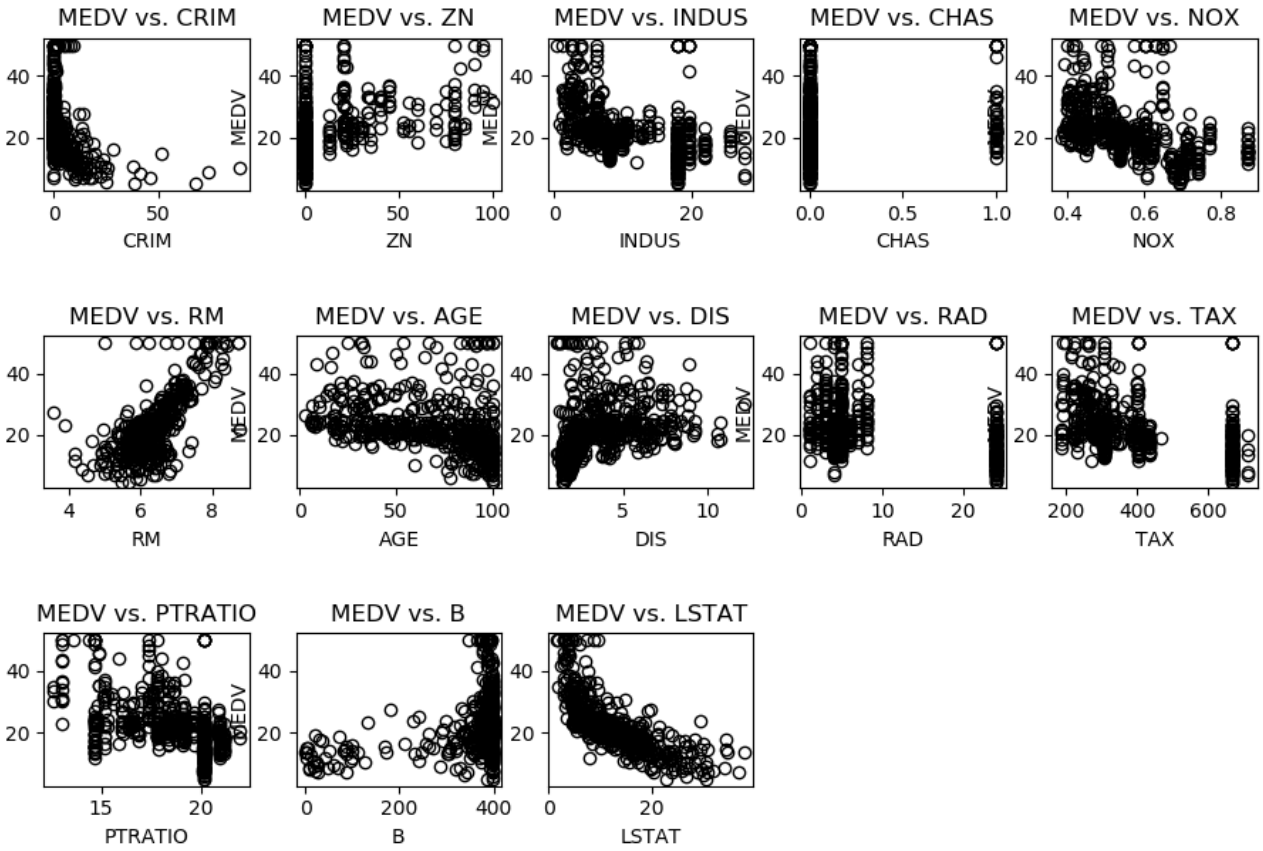
(Part 1) Learning Basics of Regression in Python

Describe and Summarize the data

- 506 data samples
- 13 input (explanatory features) named:
  1. CRIM (range: 0.00632 - 88.9762)
  2. ZN (range: 0.0 - 100.0)
  3. INDUS (range: 0.46 - 27.74)
  4. CHAS (range: 0.0 - 1.0) which is binary as it only takes a value between 0.0 and 1.0 and nothing in between.
  5. NOX (range: 0.385 - 0.871)
  6. RM (range: 3.561 - 8.78)
  7. AGE (range: 2.9 - 100.0)
  8. DIS (range: 1.1296 - 12.1265)
  9. RAD (range: 1.0 - 24.0)
  10. TAX (range: 187.0 - 711.0)
  11. PTRATIO (range: 12.6 - 22.0)
  12. B (range: 0.32 - 396.9)
  13. LSTAT (range: 1.73 - 37.97)
- Features consists of real and positive numbers
- Target name is MEDV, the median value of a house and it consists of real numbers (range: 5.0 - 50.0)

Visualization

This is a plot of every feature against the target where the feature and targets are on the x and y-axes of the Cartesian plane respectively.



Feature Weights

#	Feature Name	Value	Description
0	Bias	34.72315854	**Not a feature - Bias Value**
1	CRIM	-0.1001396	per capita crime rate by town
2	ZN	0.0411024	proportion of residential land zoned for lots over 25,000 sq.ft.
3	INDUS	0.02359639	proportion of non-retail business acres per town
4	CHAS	2.74448198	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5	NOX	-17.49103746	nitric oxides concentration (parts per 10 million)
6	RM	4.16892358	average number of rooms per dwelling
7	AGE	-0.01047614	proportion of owner-occupied units built prior to 1940
8	DIS	-1.4367399	weighted distances to five Boston employment centres
9	RAD	0.29696781	index of accessibility to radial highways
10	TAX	-0.01349442	full-value property-tax rate per $10,000
11	PTRATIO	-0.93914748	pupil-teacher ratio by town
12	B	0.00733658	1000(Bk - 0.63)ˆ2 where Bk is the proportion of blacks by town
13	LSTAT	-0.45653296	% lower status of the population

INDUS, as the table states, is the proportion of non-retail business acres per town. The weight associated with it is positive implying a positive correlation with the target (i.e. an increase or decrease in INDUS will result in an increase or decrease in the housing price respectively).

However, 0.023 is a minuscule in magnitude and this means that a large change in INDUS will result in a small change in the housing price. Therefore, INDUS is a statistically insignificant explanatory variable.

This makes sense as the non-retail business acres per town would is a secondary feature relative to: pollution, number of rooms, population status, and the Charles River.

Mean Squared Error Value

The MSE obtained from the test set was **23.423**.

Two Other Loss Functions

Two other loss functions that were used are:

1. **The Absolute Loss function**, L_1:

$$L_1 = \sum_{i=1}^{N} |y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}|$$

The loss function gave us an error of **3.722**. It was chosen as the L_2 loss function heavily penalizes outliers, whereas, the L_1 penalizes all samples equally. The huge difference between L_2's 23.423 and L_1's 3.722 (even squared is below 16) shows that there are a number of outlier points that L_2 has penalized. This outlier set could be anomalies that the loss function should not optimize for and L_1 tells us this.

2. **The Huber Loss function** L_δ:

$$L_\delta = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

This function gave us an error of **4.204**. It was chosen as it gives us the best of both L_1 and L_2. It heavily penalizes nearby points (using the square of the distance) and ignores distant outliers (by treating them as an L_1 loss). The point at which it switches between either case is determined by δ, a user defined parameter. Using the standard value of $\delta = 1.35$, the error was obtained. It was closer to the L_1 loss rather than the L_2. This further justifies the existence of many outliers and shows there are far more consistently close samples.

Most Significant Features

The most significant features from the table by magnitude and logic are **in no order**:

- **CRIM (-0.10)**: The per capita **crime rate** has a negative correlation with the price. Crime is a deterrent to raising housing prices.
- **CHAS (2.74)**: Houses along the **Charles River** are more expensive, which is plausible as they might be prime real-estate due to their view and access to the river.
- **NOX (-17)**: Housing prices should be **lower in polluted areas** and this remains true with the negative correlation in the MEDV vs NOX graph. However, the large weight value for pollution is probably due to its small magnitude as a feature (0.4- 0.9).
- **RM (4.16)**: Houses with **more rooms should be more expensive** and that remains the case. The number of rooms and the size of the house is generally what is listed in the housing flyers (important) and the weight confirms this.
- **LSTAT (-.45)**: Houses in neighbourhoods with **lower status would be cheaper**. The graph also shows a linear negative correlation between MEDV vs LSTAT. The low weight value is could be because of its high mean sample value.
- **TAX (-.013)**: The higher the property tax, the less incentive for buyers to buy. Taxes has a large sample values (from 187 to 711). The weight associated with it might not accurately represent its importance.

Note: The feature weights table's weights were taken as is based on the instructions of the assignment. Features have different value ranges. Therefore, features with low value ranges, and are in reality good predictors, could have an exponentially higher weight to compensate for the low magnitude range. In order to have a less biased view, **feature scaling should be performed, after which, weights calculated**.

(Part 2) Locally Reweighted Regression

(1) Solution to the Weighted Least Square Problem

Given

$$w^* = \arg\min_w \frac{1}{2}\sum_{i=1}^{N} a^i (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

show

$$w^* = (\mathbf{X}^T \mathbf{A}\mathbf{X} + \lambda \mathbf{I}^{-1})\mathbf{X}^T \mathbf{A}\mathbf{Y}$$

The solution is:
Take the equation of the Loss function $L(w)$ and minimize with respect to w^*.
Therefore, the Loss Function is:

$$L(w) = \frac{1}{2}\sum_{i=1}^{N} a^i (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

Replace the Sum Components with Vectors $a^{(i)}, y^{(i)}, \mathbf{w}^T, \mathbf{x}^{(i)}$ etc. as a column vector **or** a matrix with the row length N depending on what they initially were.

As $a^{(i)}$ and $y^{(i)}$ were previously scalar values, they become column vectors with row length N. They changed from $a^{(i)}$ to $\mathbf{a}$ and $y^{(i)}$ to $\mathbf{y}$ to indicate column vectors.

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Similarly, $\mathbf{w}^T$ and $\mathbf{x}^{(i)}$ were previously a row and column vector respectively:

$$\mathbf{w}^T = \begin{bmatrix} w_0 & w_1 & \dots & w_D \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_D^i \end{bmatrix}$$

where D is the number of input features and N is the number of data points/samples. Note that $\mathbf{w}^T$ is independent of (i), the current data sample being iterated over.

Therefore, $\mathbf{x}$ can be replaced with the following Matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_D^1 \\ 1 & x_1^2 & x_2^2 & & x_D^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^N & x_2^N & \dots & x_D^N \end{bmatrix}$$

resulting in $\mathbf{X}$ being a $(N \times (D+1))$ dimensional matrix.
We replace the existing variables with vectors and matrices, remove the summation. The equation can be rewritten as:

$$L(w) = \frac{1}{2}\mathbf{a}(\mathbf{y} - \mathbf{X}\mathbf{w})^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

The vector **a** has been changed to the matrix **A** as shown in the instructions in order to maintain the dimensionality. This can be expanded to:

$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T\mathbf{A}(\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}(\mathbf{w}^T\mathbf{w})$$

$$= \frac{1}{2}(\mathbf{y}^T\mathbf{A} - \mathbf{w}^T\mathbf{X}^T\mathbf{A})(\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}(\mathbf{w}^T\mathbf{w})$$

$$= \frac{1}{2}(\mathbf{y}^T\mathbf{A}\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{A}\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w}) + \frac{\lambda}{2}(\mathbf{w}^T\mathbf{w})$$

Now, taking the gradient of $L(\mathbf{w})$ and setting the value to 0 (i.e. $\nabla L(\mathbf{w}) = 0$) in order to obtain the $\arg\min_w$:

$$\nabla L(\mathbf{w}) = \frac{1}{2}(0 - 2\mathbf{X}^T\mathbf{A}\mathbf{y} + 2\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w}) + \lambda\mathbf{w} = 0$$

$$-\mathbf{X}^T\mathbf{A}\mathbf{y} + \mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = 0$$

$$+\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{A}\mathbf{y}$$

Separating the equation for **w**:

$$\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{A}\mathbf{y}$$

$$(\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{A}\mathbf{y}$$

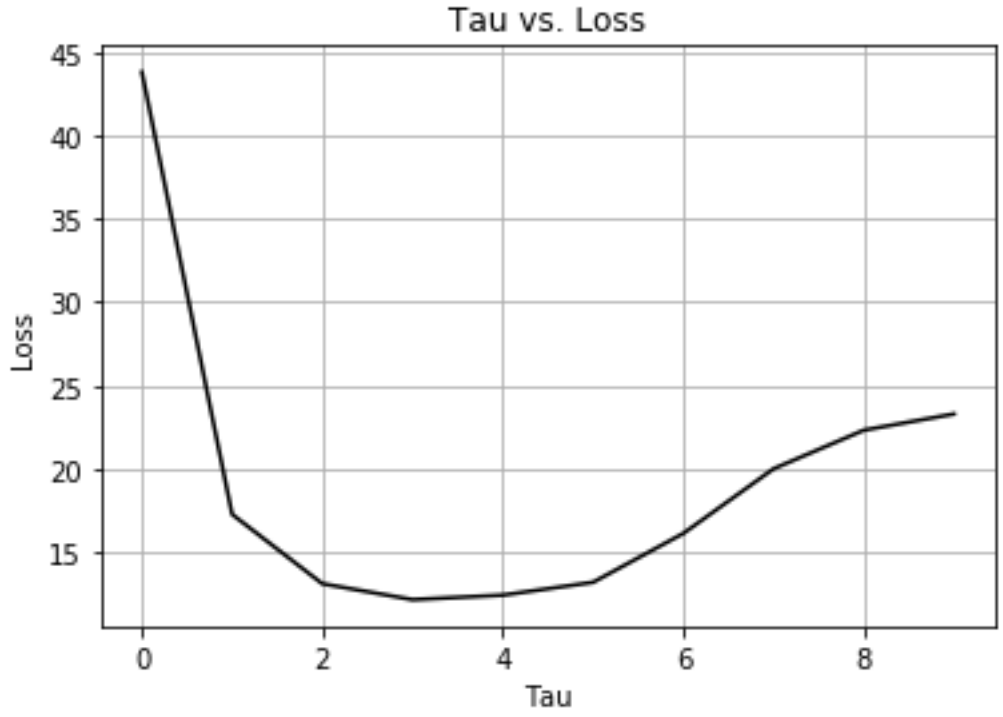
Leading to the final proof by taking the $\arg\min_w = \mathbf{w}^*$:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{A}\mathbf{y}$$

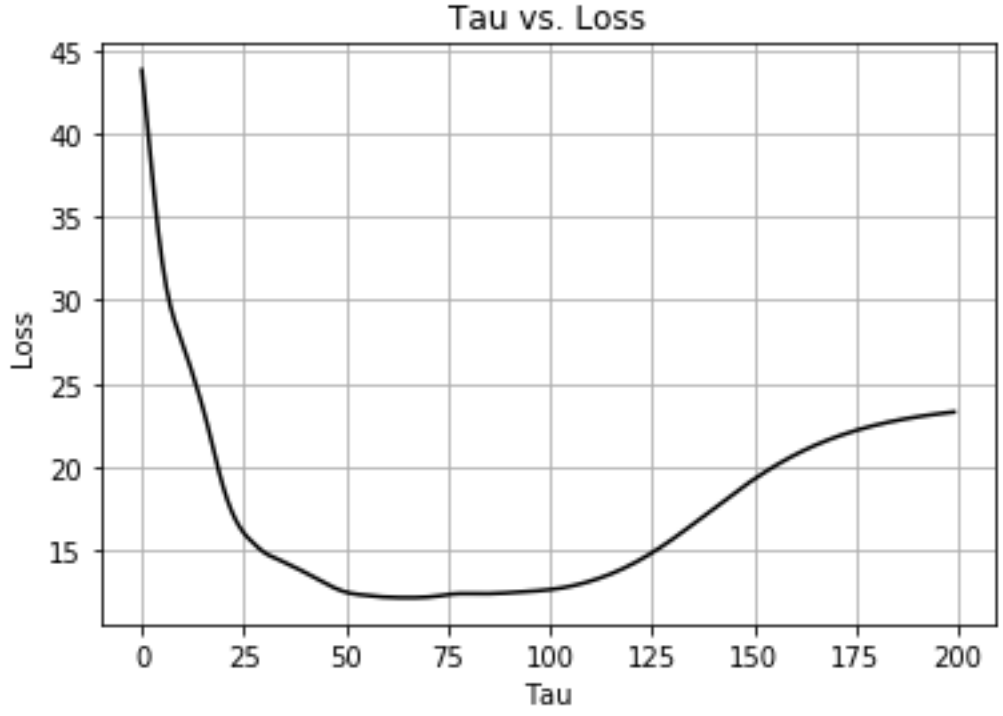
Thus proved.

(3) k-fold Cross Validation Plots

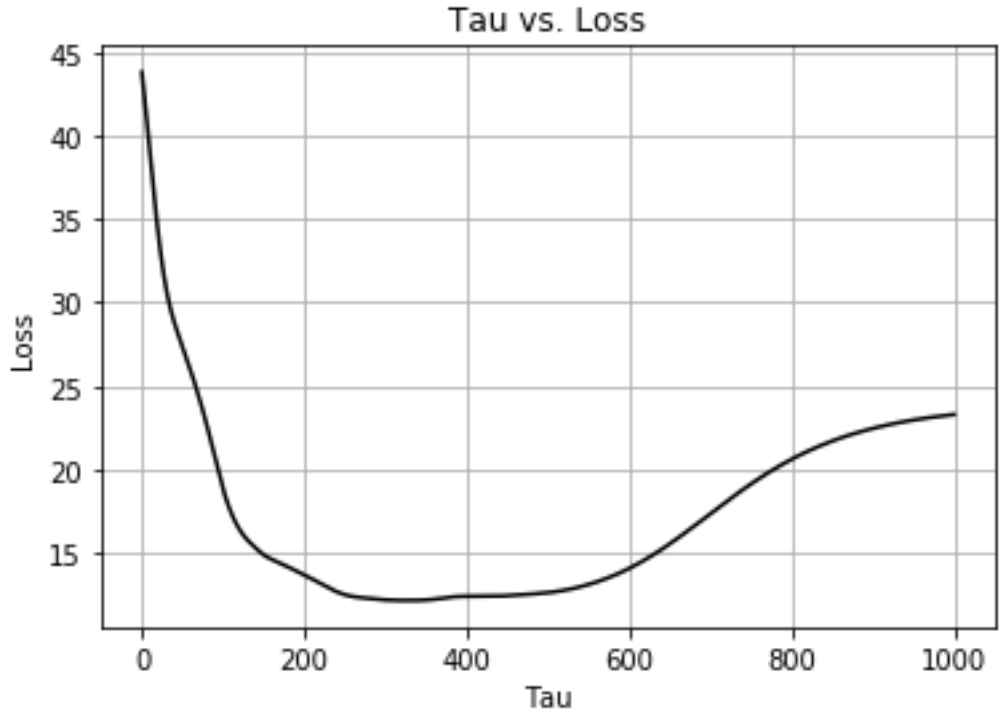
Using the k-fold Cross Validation technique, I have plotted the loss values versus τ for the following values 10, 200 and 1,000. The plots are:
For $\tau = 10$:



For $\tau = 200$:



For $\tau = 1000$:



(4) Algorithm Behaviour for $\tau \to 0$ and $\tau \to \infty$

From the graphs in the previous section, we note that as τ **approaches 0**, it **increases drastically** in value and **approaches a value of 44**.

Similarly, from the graphs in the previous section, we note that as τ **approaches** ∞, the **value stabilizes** to **approximately 24**.

(Part 3) Mini-batch SGD Gradient Estimator

(1) Mini-Batch Proof

We need to prove that:

$$\mathbb{E}_I[\frac{1}{m}\sum_{i \epsilon I} a_i] = \frac{1}{n}\sum_{i=I}^{n} a_i$$

The expected value of a given variable can be expanded to:

$$\mathbb{E}[X] = \sum_{x}^{S} p(x)f(x)$$

where $p(x)$ is the probability function of x and $f(x)$ is the given value of x over the entire sample space S.

Applying this identity on the L.H.S of the first equation, we get:

$$\mathbb{E}_I[\frac{1}{m}\sum_{i\epsilon I} a_i] = \sum_I p(I)(\frac{1}{m}\sum_{i\epsilon I} a_i)$$
$$= \frac{1}{m}\sum_I p(I)(\sum_{i\epsilon I} a_i)$$

As $p(I)$ is uniformly distributed equally and we take $\binom{n}{m}$ such batches, the equation can be rewritten as:

$$= \frac{1}{m \times \binom{n}{m}} \sum_{j=1}^{\binom{n}{m}} \sum_{i\epsilon I_j} a_i$$

Expanding the $\binom{n}{m}$, we get:

$$= \frac{m!(n-m)!}{m \times n!} \sum_{j=1}^{\binom{n}{m}} \sum_{i\epsilon I_j} a_i$$

This simplifies to:

$$= \frac{(m-1)!(n-m)!}{n!} \sum_{j=1}^{\binom{n}{m}} \sum_{i\epsilon I_j} a_i$$

The $\sum\sum$ represents all sum of all entries in a min-batch, for all min-batches. Therefore it can be written as the sum of all mini-batches:

$$= \frac{(m-1)!(n-m)!}{n!} \sum_{i=1}^{n} \binom{n-1}{m-1} a_i$$

The binomial constants can be taken out and expanded to:

$$= \frac{(m-1)!(n-m)!}{n!} \times \frac{(n-1)!}{(m-1)!(n-m)!} \sum_{i=1}^{n} a_i$$

This results in the final answer of:

$$\mathbb{E}_I[\frac{1}{m}\sum_{i\epsilon I} a_i] = \frac{1}{n}\sum_{i=I}^{n} a_i$$

Thus proved.

(2) Show $\mathbb{E}_I[\nabla L_I(x, y, \theta)] = \nabla L(x, y, \theta)$

The solution is:
From the instructions, take:

$$L_I(x, y, \theta) = \frac{1}{m}\sum_{i\epsilon I} l(\mathbf{x}^{(i)}, y^{(i)}, \theta)$$

Apply the gradient on both sides:

$$\nabla L_I(x, y, \theta) = \frac{1}{m}\sum_{i\epsilon I} \nabla l(\mathbf{x}^{(i)}, y^{(i)}, \theta)$$

Apply the Expected value on both sides:

$$\mathbb{E}_I[\nabla L_I(x, y, \theta)] = \frac{1}{m} \sum_{i \epsilon I} \mathbb{E}_I[\nabla l(\mathbf{x}^{(i)}, y^{(i)}, \theta)]$$

Both $\mathbb{E}_I$ and ∇ are linear operators that can be exchanged **if** the function is sufficiently smooth and bounded (which the loss function is). Therefore, over a uniform dataset, U, we can rewrite:
The identity:

$$\mathbb{E}_{n \sim U}[\nabla L_n(w)] = \nabla \ \mathbb{E}_{n \sim U}[L_n(w)]$$

First, we take the the expected value and the gradient outside the summation:

$$\mathbb{E}_I[\nabla L_I(x, y, \theta)] = \mathbb{E}_I[\nabla(\frac{1}{m} \sum_{i \epsilon I} l(\mathbf{x}^{(i)}, y^{(i)}, \theta)])$$

Applying the identity, we can switch the ∇ and the $\mathbb{E}_I$:

$$\mathbb{E}_I[\nabla L_I(x, y, \theta)] = \nabla(\mathbb{E}_I[\frac{1}{m} \sum_{i \epsilon I} l(\mathbf{x}^{(i)}, y^{(i)}, \theta)])$$

From the previous question, we get the following statement (which we had to prove):

$$\mathbb{E}_I[\frac{1}{m} \sum_{i \epsilon I} a_i] = \frac{1}{n} \sum_{i=1}^{n} a_i$$

Applying the statement to the current equation and substituting the purple part, we get this:

$$\mathbb{E}_I[\nabla L_I(x, y, \theta)] = \nabla(\frac{1}{n} \sum_{i=1}^{n} l(\mathbf{x}^{(i)}, y^{(i)}, \theta)])$$

Which is basically:

$$\mathbb{E}_I[\nabla L_I(x, y, \theta)] = \nabla(L(\mathbf{x}, y, \theta))$$

Thus proved.

(3) Importance of the Previous Result

In simple English, the previous part asked us to show that the gradient of a mini-batch is a good estimator (equal to) for the gradient of the entire Loss function.

It is important as it **mathematically justifies sampling and the use of the mini-batch's gradient as an estimator of the gradient of the entire data set**.

(4a) ∇L for Cost Function

Given that:

$$L(x, y, \theta) = \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{x}^{(i)}, y^{(i)}, \theta))$$

and the loss function $l(\mathbf{x}^{(i)}, y^{(i)}, \theta)$ is:

$$l(\mathbf{x}^{(i)}, y^{(i)}, \theta)) = (y - w^T\mathbf{x})^2$$

Therefore,

$$L(x, y, \theta) = \frac{1}{n}\sum_{i=1}^{n}(y - w^T\mathbf{x})^2$$

Transforming the equations into vectors and matrices in the same manner as was done in class and in the Solution to the Weighted Least Square Problem earlier.

Therefore, the equation can be represented as:

$$L(x, y, w) = \frac{1}{n}||\mathbf{y} - \mathbf{X}\mathbf{w}||^2$$

Applying the exact same steps as performed in the Solution to the Weighted Least Square Problem, we get:

$$L(x, y, w) = \frac{1}{n}(\mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w})$$

This differentiates to (w.r.t w):

$$\nabla(L(\mathbf{w})) = \frac{2}{n}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y})$$

Thus solved.

(5) True Gradient (∇L)vs. Batch Gradient Comparisons

The weights obtained from the true gradient, ∇L have been compared with the weights obtained by performing batch gradients using the following metrics:

- **Squared Distance Metric** gives a value of 825.083. The larger the values, the larger the difference in their magnitudes.
- **Cosine Similarity** gives a value of 0.999999942522, where 1 is for identical vectors.

It would be more meaningful to take the cosine similarity metric in this case. Differences in the magnitudes between the batch gradient and true gradient weights can be high. However, the weight signifies the importance of a parameter relative to other parameters. The cosine similarity check compares that and therefore, it is more meaningful.

(6) Plot $\log \sigma_j$ against $\log m$

The single parameter chosen, w_j was the RM (average number of rooms per dwelling) for $j = 5$ and the graph was plotted for m in the range of [0,400] and $K = 500$: