Report on HW1

ZHANG Qianhao / 1004654377

Part 1

> Tabulate each feature along with its associated weight and present them in a table.
> Explain what the sign of the weight means in the third row ('INDUS') of this table. Does the sign match what you expected? Why?

feature	weight
CRIM	-1.35894776e-01
ZN	6.62353415e-02
INDUS	-2.00219590e-02
CHAS	3.64229004e+00
NOX	-2.53354891e+01
RM	2.49604312e+00
AGE	8.30423273e-03
DIS	-2.11422052e+00
RAD	4.15121115e-01
TAX	-9.85323871e-03
PTRATIO	-8.26728203e-01
B	1.24525892e-02
LSTAT	-7.26666188e-01

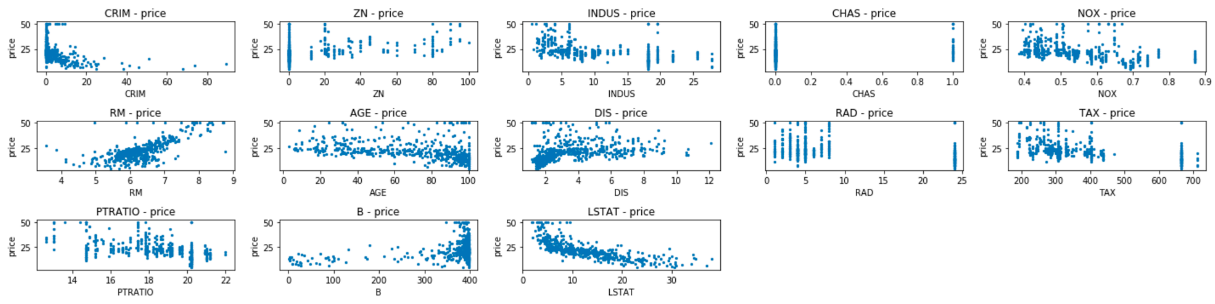
The sign negative sign of 'INDUS' is expected because it can be observed from the visualization that higher prices are more concentrated in the zone where INDUS is low, suggesting that it has negative influence upon the price.

> Suggest and calculate two more error measurement metrics; justify your choice.

Mean absolute error: to remain robust in the case of outliers. (L2 loss is not very stable if there are outliers) R squared loss: if confident that the model should be linear, R2 is a very intuitive scale of how well the model fits the data.

> Feature Selection: Based on your results, what are the most significant features that best predict the price? Justify your answer.

`RM` and `LSTAT`.



Based on the visualization, `features[5] RM` and `features[12] LSTAT` shows the most linearity againg the price. After the computation of f-scores and p-values (making use of `sklearn.feature_selection.f_regression`) for all the 13 features, the features with highest f-scores and lowest p-values are also `RM` and `LSTAT`.

```
MSE: 17.8281534772
MAE: 3.33427059657
R2: 0.62943646747
f_score: [  88.15124178   75.2576423   153.95488314   15.97151242
112.59148028
  471.84673988   83.47745922   33.57957033   85.91427767  141.76135658
  175.10554288   63.05422911  601.61787111]
p_value: [  2.08355011e-19   5.71358415e-17   4.90025998e-31   7.39062317e-
05
   7.06504159e-24   2.48722887e-74   1.56998221e-18   1.20661173e-08
   5.46593257e-19   5.63773363e-29   1.60950948e-34   1.31811273e-14
   5.08110339e-88]
```

- The other steps are included in the code. (see the comments in main function of q1.py)

Part 2

1. Given $\{(\mathbf{x}^{(1)}, y^{(1)}), .., (\mathbf{x}^{(N)}, y^{(N)})\}$ and positive weights $a^{(1)}, ..., a^{(N)}$ show that the solution to the *weighted* least square problem

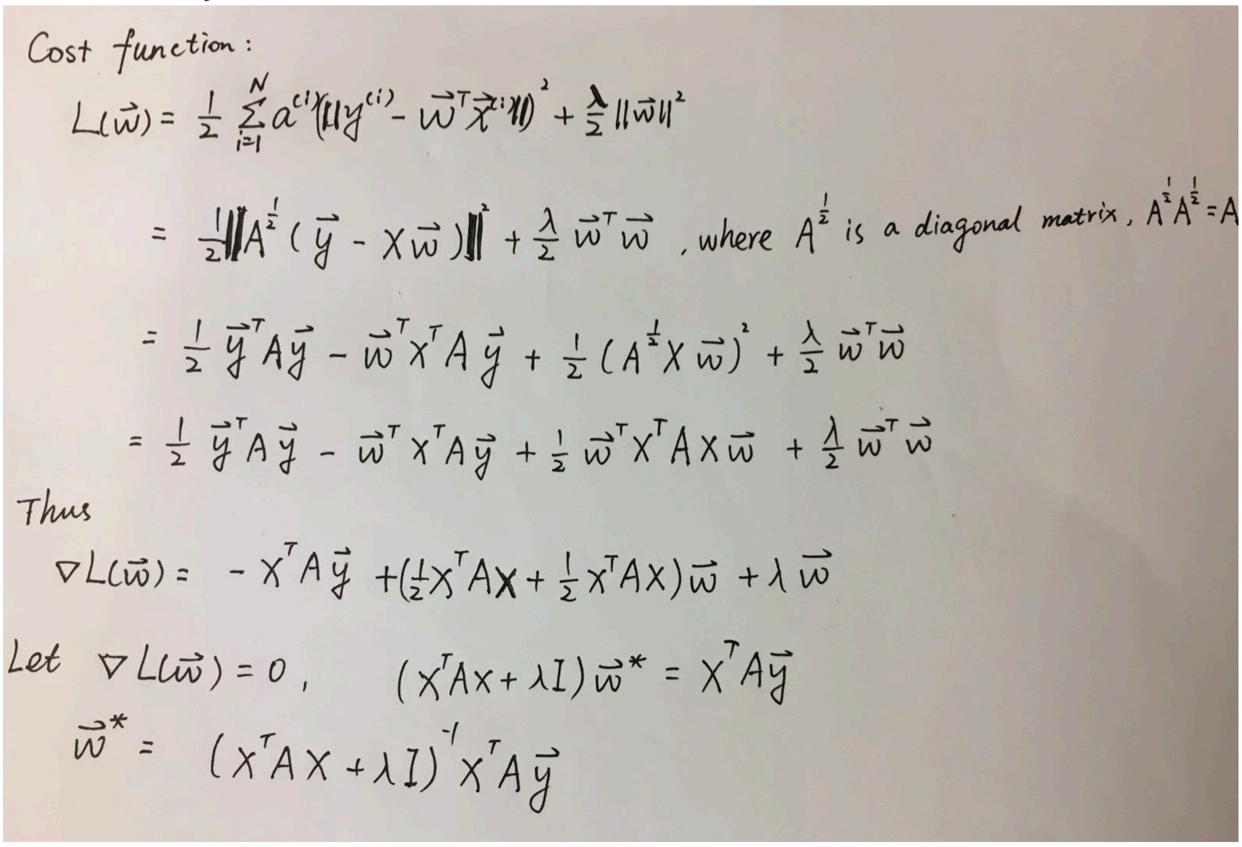
$$\mathbf{w}^* = \arg\min \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{1}$$

is given by the formula

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{A}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{A}\mathbf{y} \tag{2}$$

where $\mathbf{X}$ is the design matrix (defined in class) and $\mathbf{A}$ is a diagonal matrix where $\mathbf{A}_{ii} = a^{(i)}$

Note that A is symmetric, $A^T = A$



Cost function:

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

$$= \frac{1}{2}||A^{\frac{1}{2}}(\vec{y} - X\vec{w})||^2 + \frac{\lambda}{2}\vec{w}^T\vec{w}$$, where $A^{\frac{1}{2}}$ is a diagonal matrix, $A^{\frac{1}{2}}A^{\frac{1}{2}} = A$

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

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

Thus

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

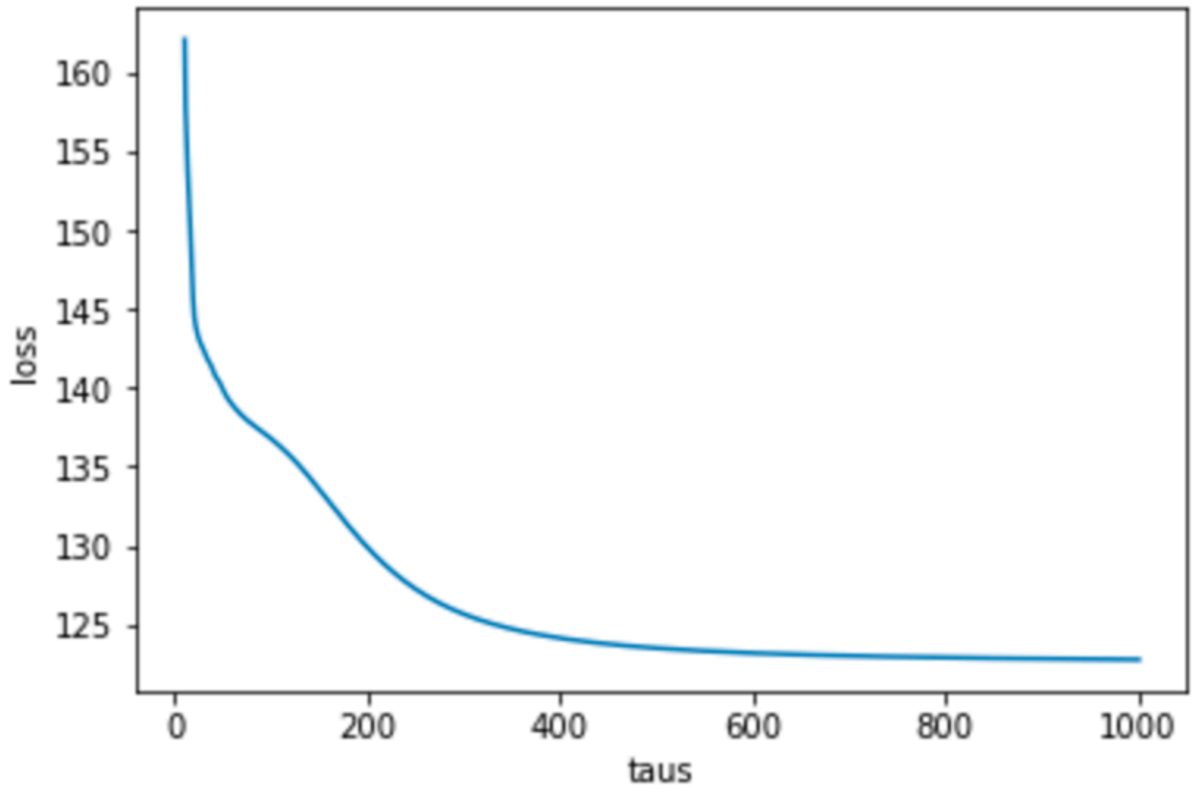
Let $\nabla L(\vec{w}) = 0$, $(X^T A X + \lambda I)\vec{w}^* = X^T A\vec{y}$

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

2. Locally reweighted least squares combines ideas from k-NN and linear regression. For each new test example $\mathbf{x}$ we compute distance-based weights for each training example $a^{(i)} = \frac{\exp(-||\mathbf{x}-\mathbf{x}^{(i)}||^2/2\tau^2)}{\sum_j \exp(-||\mathbf{x}-\mathbf{x}^{(j)}||^2/2\tau^2)}$, computes $\mathbf{w}^* = \arg\min \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||^2$ and predicts $\hat{y} = \mathbf{x}^T\mathbf{w}^*$. Complete the implementation of locally reweighted least squares by providing the missing parts for q2.py.

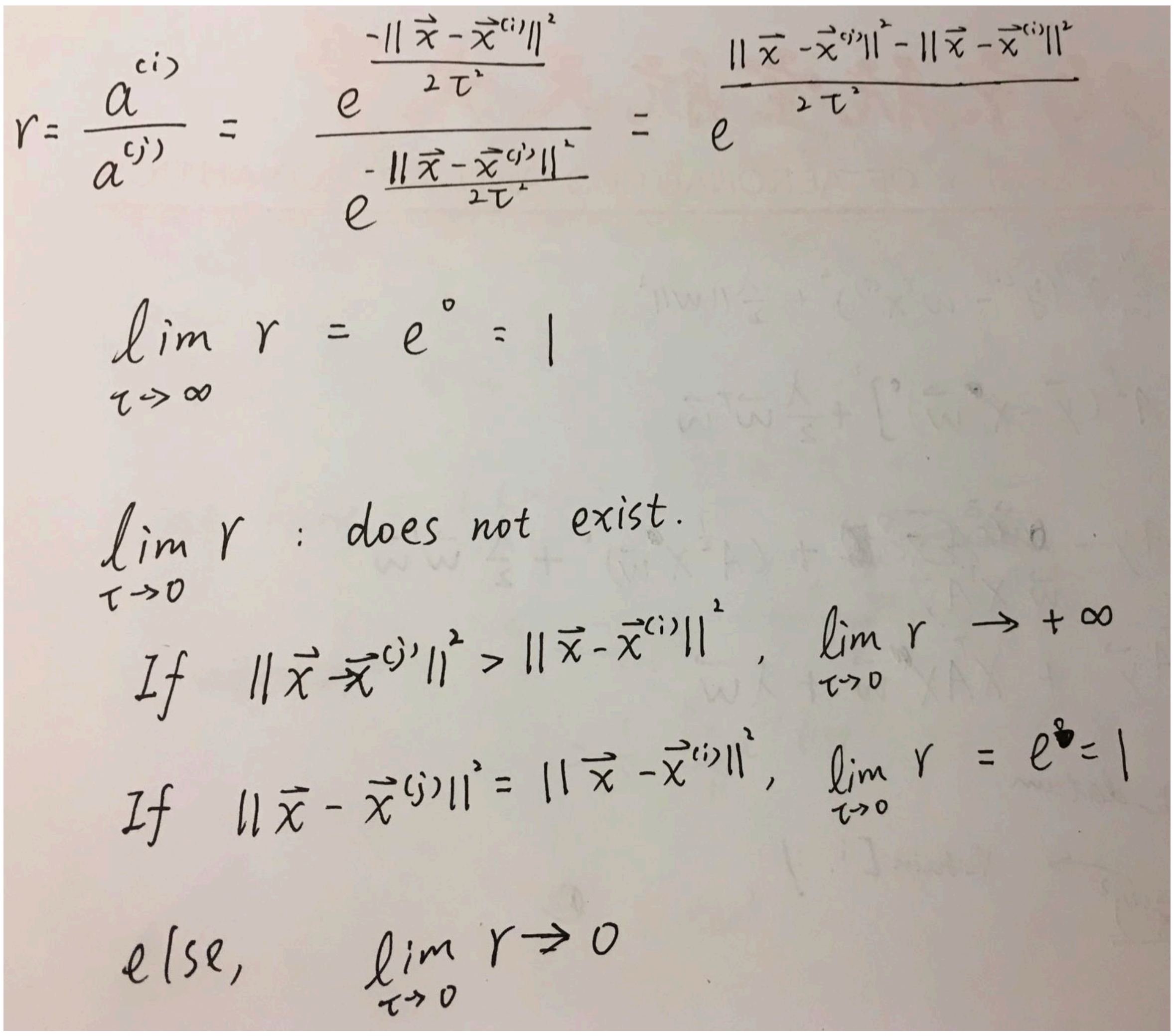
- As it is in the code.
  min loss = 122.74883420654332

3. Use k-fold cross-validation to compute the average loss for different values of τ in the range [10,1000] when performing regression on the Boston Houses dataset. Plot these loss values for each choice of τ.



- As seen. The y-axis is the value of loss function. The x-axis is the value of taus.

4. How does this algorithm behave when $\tau \to \infty$? When $\tau \to 0$?



$$r = \frac{a^{(i)}}{a^{(j)}} = \frac{e^{\frac{-\|\vec{x}-\vec{x}^{(i)}\|^2}{2\tau^2}}}{e^{\frac{-\|\vec{x}-\vec{x}^{(j)}\|^2}{2\tau^2}}} = e^{\frac{\|\vec{x}-\vec{x}^{(j)}\|^2 - \|\vec{x}-\vec{x}^{(i)}\|^2}{2\tau^2}}$$

$$\lim_{\tau \to \infty} r = e^0 = 1$$

$\lim_{\tau \to 0} r$: does not exist.

If $\|\vec{x}-\vec{x}^{(j)}\|^2 > \|\vec{x}-\vec{x}^{(i)}\|^2$, $\lim_{\tau \to 0} r \to +\infty$

If $\|\vec{x}-\vec{x}^{(j)}\|^2 = \|\vec{x}-\vec{x}^{(i)}\|^2$, $\lim_{\tau \to 0} r = e^0 = 1$

else, $\lim_{\tau \to 0} r \to 0$

- As τ → ∞, the ratio of different weights approaches 1. In this case, the algorithm becomes just like unweighted.
  Specifically, because the limit of $a^{(i)}/a^{(j)}$ when τ → ∞ is 1, for any i and j in matrix A, A = I.
  Thus $\vec{w}^*$ = $(X^T X + \lambda I)^{-1} X^T \vec{y}$, which is exactly the optimal solution of an unweighted least square problem.
- As τ → 0, the ratio of different weights approaches infinite or 0 (except that the norms of the 2 test vectors).
  So the optimization of the algorithm will be totally biased to the test case(s) with the largest deviation. The only effective test case(s) will be the one(s) with the largest deviation $|||\vec{x} - \vec{x}^{(i)}||^2$.
  Specifically, w.l.o.g. suppose $\vec{x}^{(i)}$ is the test case with the largest deviation, then A will be a N by N matrix where $A_{i,i}$ = 1 and any other element is 0. So $A * A = A$.
  Thus,
  $\vec{w}^*$ = $(X^T AAX + \lambda I)^{-1} X^T AA\vec{y}$
  $\vec{w}^*$ = $(X'^T X' + \lambda I)^{-1} X'^T \overrightarrow{y'}$, where
  $X' = AX = [0 \ldots \vec{x}^{(i)T} \ldots 0]$
  $\overrightarrow{y'} = A\overrightarrow{y'} = [0 \ldots y^{(i)} \ldots 0]^T$
  So the matrix A in this case is filtering out any test case other than $\vec{x}^{(i)}, y^{(i)}$. The optimization of $\vec{w}$ will be focused only on that case.

Part 3

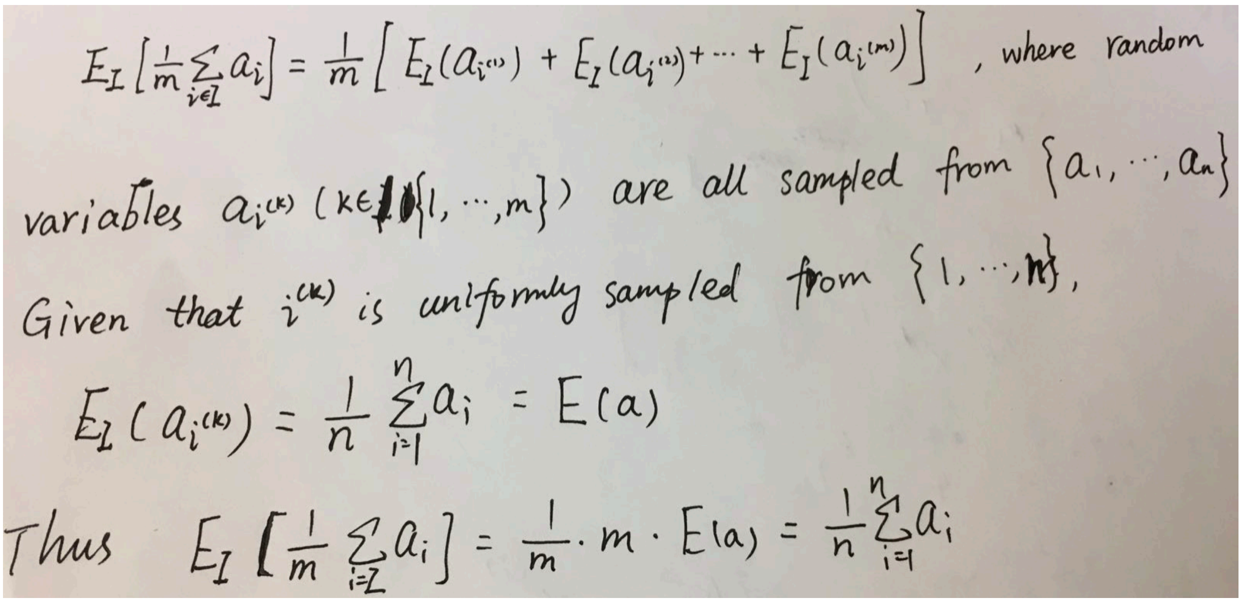
Consider a dataset $\mathcal{D}$ of size n consisting of $(\mathbf{x}, y)$ pairs. Consider also a model $\mathcal{M}$ with parameters θ to be optimized with respect to a loss function $L(\mathbf{x}, y, \theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}^{(i)}, y^{(i)}, \theta)$.

We will aim to optimize L using mini-batches drawn randomly from $\mathcal{D}$ of size m. The indices of these points are contained in the set $\mathcal{I} = \{i_1, \ldots, i_m\}$, where each index is distinct and drawn uniformly without replacement from $\{1, \ldots, n\}$. We define the loss function for a single mini-batch as,

$$L_{\mathcal{I}}(\mathbf{x}, y, \theta) = \frac{1}{m}\sum_{i\in\mathcal{I}} \ell(\mathbf{x}^{(i)}, y^{(i)}, \theta) \tag{3}$$

1. Given a set $\{a_1, \ldots, a_n\}$ and random mini-batches $\mathcal{I}$ of size m, show that

$$\mathbb{E}_{\mathcal{I}}\left[\frac{1}{m}\sum_{i\in\mathcal{I}} a_i\right] = \frac{1}{n}\sum_{i=1}^{n} a_i$$



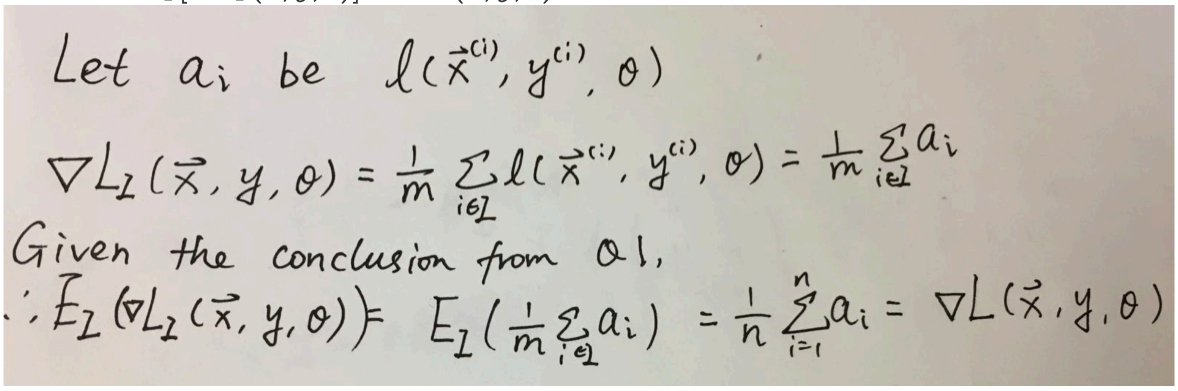
$E_I[\frac{1}{m}\sum_{i\in I} a_i] = \frac{1}{m}[E_I(a_{i^{(1)}}) + E_I(a_{i^{(2)}}) + \cdots + E_I(a_{i^{(m)}})]$, where random variables $a_{i^{(k)}}$ ($k \in \{1, \cdots, m\}$) are all sampled from $\{a_1, \cdots, a_n\}$

Given that $i^{(k)}$ is uniformly sampled from $\{1, \cdots, n\}$,

$E_I(a_{i^{(k)}}) = \frac{1}{n}\sum_{i=1}^{n} a_i = E(a)$

Thus $E_I[\frac{1}{m}\sum_{i=I} a_i] = \frac{1}{m} \cdot m \cdot E(a) = \frac{1}{n}\sum_{i=1}^{n} a_i$

2. Show that $E_I[\nabla L_I(\vec{x}, y, \theta)] = \nabla L(\vec{x}, y, \theta)$



Let a_i be $l(\vec{x}^{(i)}, y^{(i)}, \theta)$

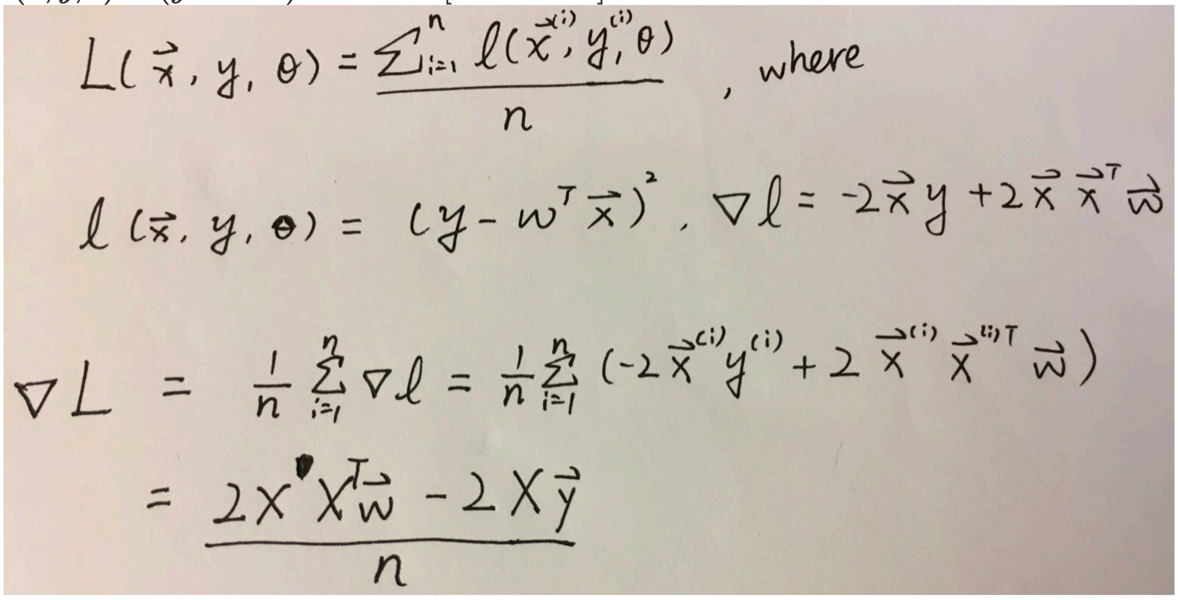
$\nabla L_I(\vec{x}, y, \theta) = \frac{1}{m}\sum_{i\in I} l(\vec{x}^{(i)}, y^{(i)}, \theta) = \frac{1}{m}\sum_{i\in I} a_i$

Given the conclusion from Q1,

$\therefore E_I(\nabla L_I(\vec{x}, y, \theta)) = E_I(\frac{1}{m}\sum_{i\in I} a_i) = \frac{1}{n}\sum_{i=1}^{n} a_i = \nabla L(\vec{x}, y, \theta)$

3. Write, in a sentence, the importance of this result. It means that the result from a stochastic mini-batch gradient descent is a unbiased estimate of that from a batch descent, so we can trust the result of SGD to be accurate.

4. (a) Write down the gradient, ∇L above, for a linear regression model with cost function $l(\vec{x}, y, \theta) = (y - \vec{w}^T\vec{x})^2$. Let $X = [\vec{x}^{(1)} \dots \vec{x}^{(n)}]$



$$L(\vec{x}, y, \theta) = \frac{\sum_{i=1}^{n} l(\vec{x}^{(i)}, y^{(i)}, \theta)}{n} \quad , \text{ where}$$

$$l(\vec{x}, y, \theta) = (y - w^T \vec{x})^2 \, , \quad \nabla l = -2\vec{x}y + 2\vec{x}\vec{x}^T\vec{w}$$

$$\nabla L = \frac{1}{n}\sum_{i=1}^{n} \nabla l = \frac{1}{n}\sum_{i=1}^{n} (-2\vec{x}^{(i)}y^{(i)} + 2\vec{x}^{(i)}\vec{x}^{(i)T}\vec{w})$$

$$= \frac{2XX^T\vec{w} - 2X\vec{y}}{n}$$

(b) Write code to compute this gradient.

- As seen in code.

5. Using your code from the previous section, for $m = 50$ and $K = 500$ compute

$$\frac{1}{K}\sum_{k=1}^{K} \nabla L_{\mathcal{I}_k}(\mathbf{x}, y, \theta)$$

, where $\mathcal{I}_k$ is the mini-batch sampled for the kth time.

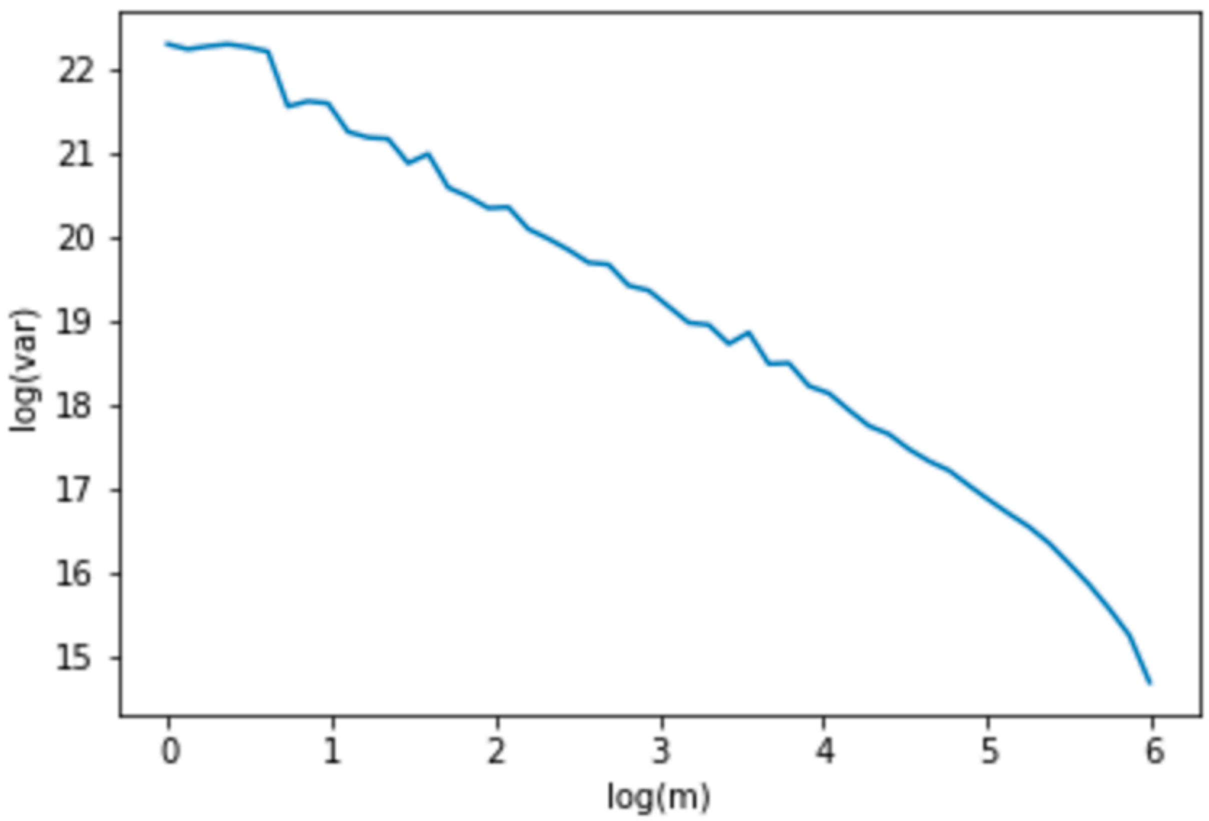
Randomly initialize the weight parameters for your model from a $\mathcal{N}(0, I)$ distribution. Compare the value you have computed to the true gradient, ∇L, using both the squared distance metric and cosine similarity. Which is a more meaningful measure in this case and why?

[Note: Cosine similarity between two vectors $\mathbf{a}$ and $\mathbf{b}$ is given by $\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||_2||\mathbf{b}||_2}$.]

```
Distance: 766866.395783
Cosine: 0.999997973411
```

Both are meaningful, but cosine is a bit better because it has a fixed scale within [-1, 1] to tell us about the deviation of direction between these 2 gradients. In this case they are remarkably close to each other (cosine value very close to 1, meaning the angle inbetween is very close to 0). On the contrary, distance metrics is not as intuitive because it is easily affected by the scale of the gradients themselves, thus its value cannot directly tell us the closeness between the gradients.

6. For a single parameter, w_j, compare the sample variance, $\tilde{\sigma}_j$, of the mini-batch gradient estimate for values of m in the range [1,400] (using K = 500 again). Plot $log(\tilde{\sigma}_j)$ against $log(m)$.



log(var)
log(m)
22
21
20
19
18
17
16
15
0
1
2
3
4
5
6