CSC2515： Assignment #1
Due: 10pm, 5th October

YIFAN ZHANG

Student Number: 1004245952

October 1, 2017

Problem 1: Learning basic of regression in Python

1. Load the Boston housing data from the sklearn datasets module.

2. Dataset description

   The dimension of dataset is (506*13).

   The number of target is 506, the shape of target' matrix is (506, 1).

   There are 13 features, including 'CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT'.

3. The grid containing plots for each feature against the target presents in Figure1-1.

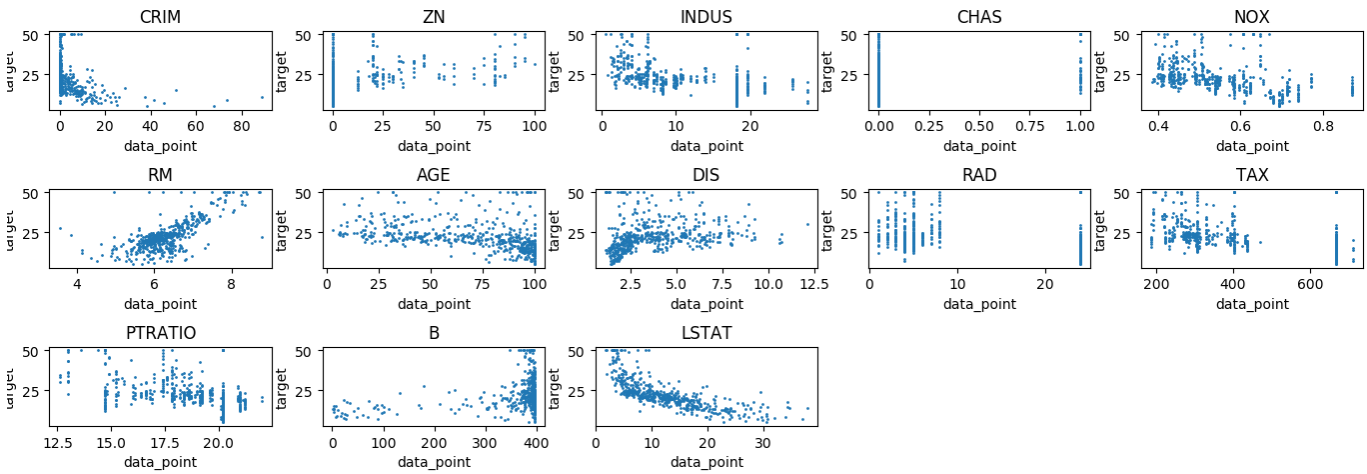

Figure1-1 Features against the target

4. Divide your data into training and test sets, where the training set consists of 80% of the data points (chosen at random). *Hint: You may find numpy.random.choice useful*

5. Write code to perform linear regression to predict the targets using the training data. Remember to add a bias term to your model.

   Theorem: The optimal $\mathbf{w}$ *w.r.t* $w^* = \arg\min \sum_{i=1}^{N} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$ is:

   $$w^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

   By applied above theorem, we could calculate w^* by the data and target of Boston dataset. Then the prediction of y are $\hat{y} = Xw^*$, we could present the

prediction of y against true y in figure1-2.

Figure1-2

6. 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 column ('INDUS') of this table. Does the sign match what you expected? Why?

The sign of the weight (INDUS) is positive, which means that the price would increase with increasing INDUS (proportion of non-retail business acres per town). This result match what I expected. Because while the proportion of non-retail business is increasing, the proportion of retail business will be likely decreasing. The house price will rise for short of supply.

Features	**Bias**	**CRIN**	**ZN**	**INDUS**	**CHAS**	**NOX**	**RN**
Weight	3.8395 3173e+01	-7.5875214 8e-02	5.68787 495e-02	4.93833 174e-02	2.81132986 e+00	-1.8655031 0e+01	3.7236480 8e+00
Features	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATTO**	**B**	**LSTAT**
Weight	7.54311 409e-	-1.4684180	2.87455 496e-01	-1.27177	-9.98740168	8.4547373 6e-03	-5.5578888

03	0e+00	867e-02	e-01	7e-01]

Table 1 Features along with weight

7. Test the fitted model on your test set and calculate the Mean Square Error of the result. We could calculate MSE by following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

The result is :MSE = 17.890943938

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

MAE and RMSE could also be used to measure accuracy for prediction in this case.

They express average model prediction error in units of the variable of interest.

We could calculate MAE by following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y} - y|^2$$

The result is: MAE = 3.18369250713

We could calculate MSE by following formula:

$$RMSE = \sqrt{MSE}$$

The result is: RMSE = 4.22976878067

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

In my opinion, the **LSTAT** could be the most significant features that beat predict the price. I calculate the correlation coefficient between prediction and each feature to justify my assumption, which is presented in the Table 2. It's easy to find out that the **LSTAT** is the closest one to 1/-1.

Features		CRIN	ZN	INDUS	CHAS	NOX	RN
Corr		-0.421234197388	0.512119803287	-0.618834937428	0.172803046661	-0.531803702096	0.745177231328
Features	**AGE**	**DIS**	**RAD**	**TAX**	**PTRATTO**	**B**	**LSTAT**
Corr	-0.525536824289	0.374639956894	-0.391954360148	-0.478882604652	-0.583414415458	0.440113229326	-0.827163970919

Table 2 Variance against each features (loop=40)

Problem 2: Locally reweighted regression

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

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

is given by the formula

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

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

Solution:

Prediction vector are $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, rewriting L($\mathbf{w}$) as following:

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

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

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

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

Important things to notice while implementing: First, do not invert any matrix, use a linear solver (*numpy.linalg.solve* is one example). Second, notice that $\frac{\exp(A_i)}{\sum_j \exp(A_j)} = \frac{\exp(A_i - B)}{\sum_j \exp(A_j - B)}$ but if we use $B = \max_j A_j$ it is much numerically stable as $\frac{\exp(A_i)}{\sum_j \exp(A_j)}$ overflows/underflows easily. *This is handled automatically in the scipy*

package with the scipy.misc.logsumexp function.

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 .

   The loss values for each choice of τ is presented in figure2-1.

   min loss = 10.045420877021392

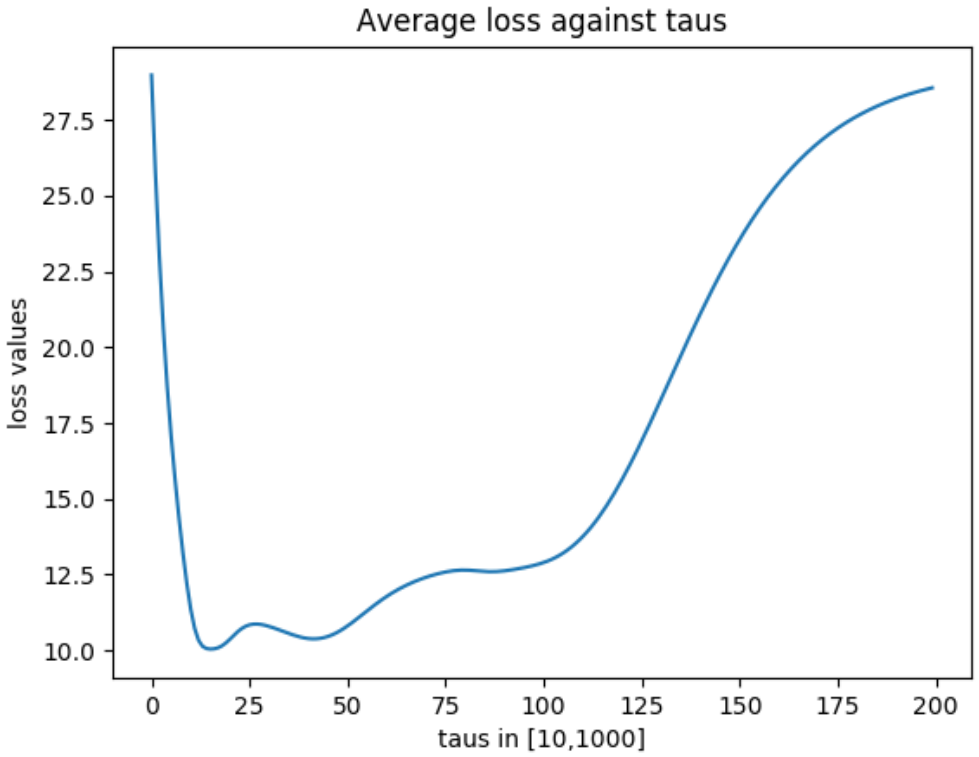

Figure2-1 Average Loss against taus

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

   From the Figure2-1, we could conclude that:

   When $\tau \to 0$, Loss value $\to \infty$.

   When $\tau \to \infty$, Loss value would converge to a fixed value.

   The conclusion could be proved by following:

$$a^{(i)} = \frac{\exp(-\left\|x - x^{(i)}\right\|^2 / 2\tau^2)}{\sum_j \exp(-\left\|x - x^{(j)}\right\|^2 / 2\tau^2)} \Rightarrow \frac{1}{a^{(i)}} = \sum_j \exp(\frac{-\left\|x - x^{(j)}\right\|^2 + \left\|x - x^{(i)}\right\|^2}{2\tau^2})$$

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

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

$$\tau \to 0, \quad a^{(i)} \to 0, \quad Loss \to \infty$$

$$\tau \to \infty, \quad a^{(i)} \to \sum_j 1 = j, \quad Loss \to \textit{fixed} \quad \textit{value}$$

Problem 3: Mini-batch SGD Gradient Estimator

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

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

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

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

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

Solution:

Because each a_i in $\{a_1,...,a_n\}$ has equal possibility, and there are $\frac{n}{m}$ patches in sum. So that the possibility for each patch is $\frac{1}{\frac{n}{m}} = \frac{m}{n}$.

$$\mathrm{E}_I[\frac{1}{m}\sum_{i\in I} a_i]$$
$$= \frac{1}{m}\mathrm{E}_I[(\sum_{i\in I_1} a_i + ... + \sum_{i\in I_m} a_i)]$$
$$= \frac{1}{m}[\frac{m}{n}(\sum_{i\in I_1} a_i + ... + \sum_{i\in I_m} a_i)] = \frac{1}{n}\sum_{i=1}^{n} a_i$$

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

Solution:

$$E[\nabla L_I(x, y, \theta)]$$
$$= E[\frac{1}{m}\sum_{i=1}^{m} \nabla L(x, y, \theta)] = \frac{1}{m}\sum_{i=1}^{m} E[\nabla L(x, y, \theta)]$$

Because batches are chosen totally by random, the possibility of each gradient is equal to $\frac{1}{n}$. So that the equation could be rewriting as:

$$\frac{1}{m}\sum_{i\in I}\nabla\left(\frac{1}{n}\sum_{j=1}^{n}L(x,y,\theta)\right)$$
$$=\nabla\left(\frac{1}{m}\sum_{i\in I}\frac{1}{n}\sum_{j=1}^{n}L(x,y,\theta)\right)$$
$$=\nabla L(x,y,\theta)$$

3. Write, in a sentence, the importance of this result.

We could get the gradient of the whole dataset by calculating the expectation of gradient of Mini-batch, which could reduce the amount of calculation.

4. (a) Write down the gradient, ∇L above, for a linear regression model with cost function $l(\mathbf{x},y,\theta)=(y-w^T\mathbf{x})^2$.

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

(b) Write code to compute this gradient.

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

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

, where I_k is the mini-batch sampled for the kth time.

Randomly initialize the weight parameters for your model from a $N(0,I)$ distribution.

Compare the value you have computed to the true gradient, rL, 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 a and b is given by $\cos(\theta)=\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|_2\|\mathbf{b}\|_2}$].

Square distance =341606.277002

Cosine = 0.999999817405

Cosine is more meaningful in this case, because by calculating cosine, we could see

that the direction of vector true gradient is almost same with the direction of vector Mini-batch But for square distance, we don't know any other data or boundaries to compare with.

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 $\log m$.

$\log \tilde{\sigma}_j$ Against $\log m$ is presented in figure3-1.

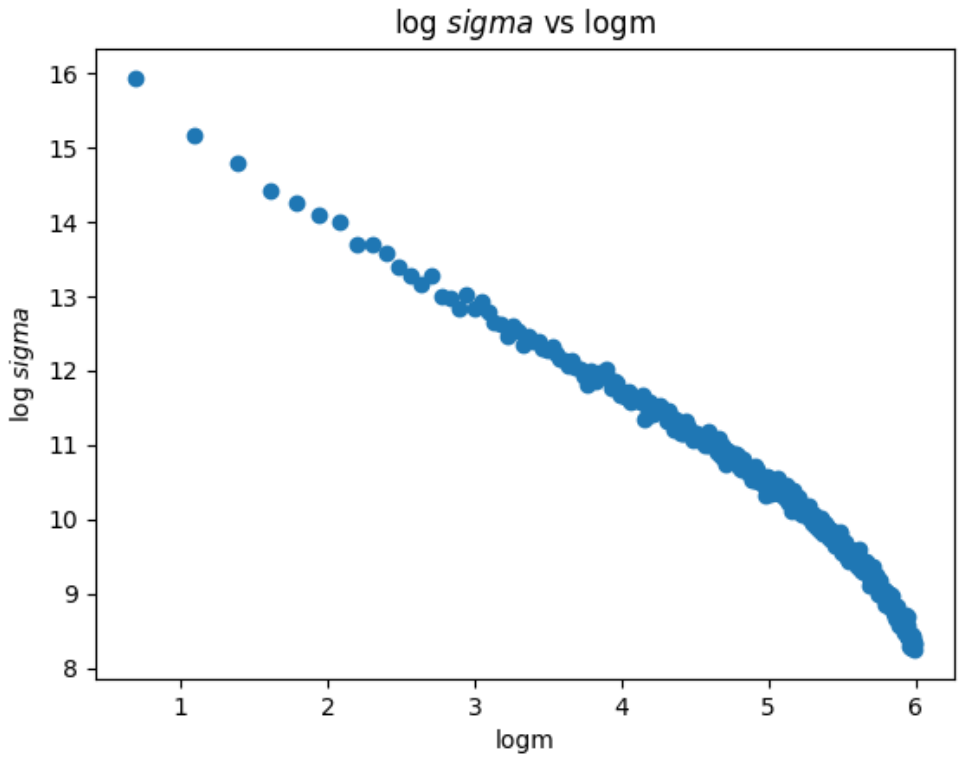

Figure3-1 $\log \tilde{\sigma}_j$ against $\log m$