CSC411 Fall 2017
Assignment 1 Report

Tianbao Li

2017/09/30

In this assignemnt, we use the dataset of Boston Housing data.

1 Q1: Leanring basics of regressoion in Python

1.1 Data loading

Here, we use function in sklearn.dataset() to read Boston Housing data.

```
def load_data():
    boston = datasets.load_boston()
    X = boston.data
    y = boston.target
    features = boston.feature_names
    return X,y,features
```

In the return variables, X is several lines of data on each feature, y is the target value for each line of data, $features$ contains the names of all the features.

1.2 Data summarization

The input dataset has the folowing properties:

- number of data points: 506
- dimensions: 13
- features: ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT']
- Mean house price: 22.5328063241
- Standard deviation of house price: 9.18801154528

1.3 Feature visualization

Data distribution on each feature is shown in Figure 1.

1.4 Data division

To divide training data set and test data set, in Q1, we use function numpy.random.choice() to implement function split_data(X, y, training_ratio = 0.2).

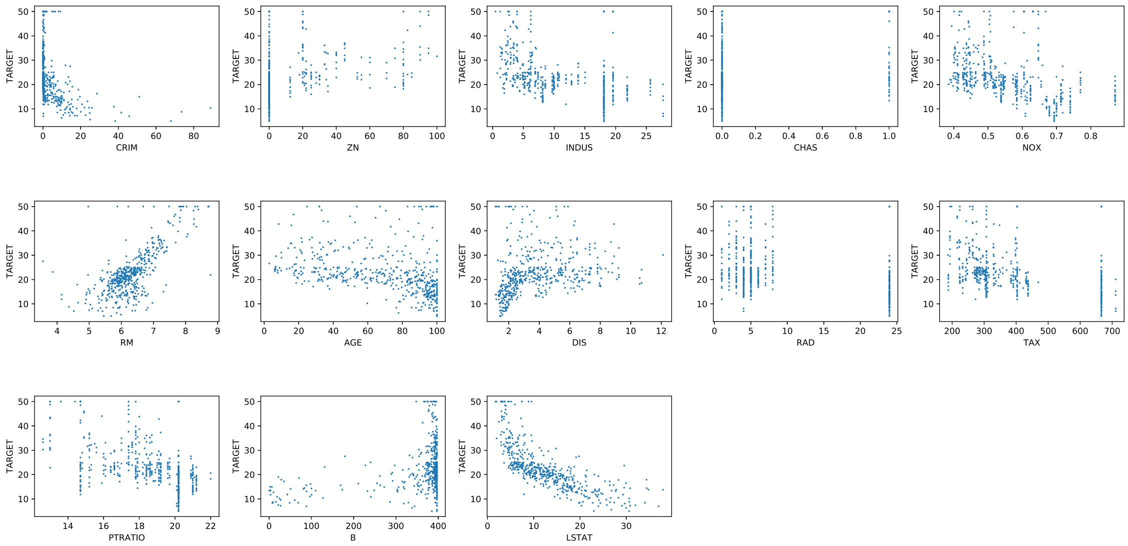

Figure 1: Data points against target for each feature

```
def split_data(X, y, training_ratio = 0.2):
    test_chosen = np.random.choice(len(X), int(len(X) * training_ratio))
    ......
    return training_set_x , test_set_x , training_set_y , test_set_y
```

In the function, we split X and y into $training_set_x$, $test_set_x$, $training_set_y$, $test_set_y$ according to $training_ratio$ for future usage.

1.5 Linear regression

To solve linear regression problem and get the weightes w, we use numpy.linalg.solve() on the formula:

$$X^{\mathrm{T}}Xw^* = X^{\mathrm{T}}y \tag{1}$$

1.6 Feature weights

After the calculation, we can get the weight on each feature as shown in Table 1.

Here, we take the feature 'INDUS' (proportion of non-retail business acres per town) as an analysis example. Shown in Table1, we can see feature 'INDUS' has a weight of 0.056336470652065491. As a positive weight, the sign means that is has a positive influence in the target, the housing price. In general ideas, people likes to live in the areas with more shops, which means that house price could be higher. That just fits the result.

1.7 Model test

We use 20% of the data as test date to just the fitness of the model. Here are some results of different metics:

- MSE: 18.268879264780917
- MAE: 3.1172529292203892

Feature	Weight	Mean
CRIM	41.795688882257188	3.5937607114624512
ZN	-0.13159561392787489	11.363636363636363
INDUS	0.056336470652065491	11.136778656126481
CHAS	0.033639549287715828	0.069169960474308304
NOX	3.7698365450326135	0.55469505928853757
RM	-19.35371695539391	6.2846343873517787
AGE	3.1091523339672826	68.574901185770756
DIS	0.010472276182880488	3.7950426877470358
RAD	-1.5681353658907364	9.5494071146245059
TAX	0.33065329716511299	408.23715415019763
PTRATIO	-0.011395113448445227	18.455533596837945
B	-0.97967928714208496	356.67403162055342
LSTAT	0.0096343495439206173	12.653063241106722

Table 1: Feature weights

- R2: 0.75025578036036022

These numbers could change due to different training-test dataset division.

1.8 Feature selection

To choose the most significan feature, weight of teh featuers are a good metrics. However, due to the different magnitudes of weights, we decide to multiply weight and mean for each feature and use it as the influnce on the target. It is shown as Table 2.

Feature	Inluence
CRIM	150.2037046136
ZN	-1.4954047037
INDUS	0.6274068039
CHAS	0.0023268463
NOX	2.0911097059
RM	-121.6310351009
AGE	213.2098140733
DIS	0.0397427352
RAD	-14.9747630197
TAX	134.9849610451
PTRATIO	-0.2103028991
B	-349.4261610401
LSTAT	0.1219040341

Table 2: Feature weights

From the table, we can see that the feature 'B' has the largest absolute value, which means 'B' predicts the price best.

2 Q2: Locally reweighted regression

2.1 w^* solution proof

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

$$w^* = \text{argmin}\, \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - w^{\mathrm{T}}x^{(i)})^2 + \frac{\lambda}{2}\|w\|^2 \tag{2}$$

is given bu the formula

$$w^* = (X^{\mathrm{T}}AX + \lambda I)^{-1}X^{\mathrm{T}}Ay \tag{3}$$

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

Proof.

$$\begin{aligned}
w^* &= \text{argmin}\, \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - w^{\mathrm{T}}x^{(i)})^2 + \frac{\lambda}{2}\|w\|^2 \\
L(w^*) &= \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - w^{\mathrm{T}}x^{(i)})^2 + \frac{\lambda}{2}\|w\|^2 \\
&= \frac{1}{2}A\|y - Xw\|^2 + \frac{\lambda}{2}\|w\|^2 \\
&= \frac{1}{2}(y - Xw)^{\mathrm{T}}A(y - Xw) + \frac{\lambda}{2}\|w\|^2 \\
&= \frac{1}{2}(y^{\mathrm{T}}Ay - 2w^{\mathrm{T}}X^{\mathrm{T}}Ay + w^{\mathrm{T}}X^{\mathrm{T}}AXw) + \frac{\lambda}{2}\|w\|^2 \\
\nabla L(w^*) &= \frac{1}{2}(-2X^{\mathrm{T}}Ay + 2X^{\mathrm{T}}AXw) + \lambda w \\
&= X^{\mathrm{T}}AXw + \lambda w - X^{\mathrm{T}}Ay
\end{aligned}$$

Setting this gradient to zero gives

$$\begin{aligned}
X^{\mathrm{T}}Ay &= X^{\mathrm{T}}AXw + \lambda w \\
w^* &= (X^{\mathrm{T}}AX + \lambda I)^{-1}X^{\mathrm{T}}Ay
\end{aligned}$$

□

2.2 LRLS implementation

Some essential things in LRLS implemented:

- $\|x - x^{(i)}\|^2$ in $a^{(i)}$ calculation: using provided function l2(A,B)
- solving w^*: using numpy.linalg.solve()
- avoiding overflows/underflows: adding $-max_j A_j$in square distance calculation

2.3 K-fold corss-validation

In this question, we choose τ from$[10, 1000]$ and proceed cross-validation on different τvalue. The variation of loss value is shown in Figure 2. For τ in$[10, 1000]$, mean value of the loss is 16.9512301253, and the minimun value of the loss is 11.9637899295.

From the figure we can see that, loss value has the following tending:

- when $\tau \to 0$, loss value tends towards $+\infty$
- when $\tau \to +\infty$, loss value tends towards a certain small value

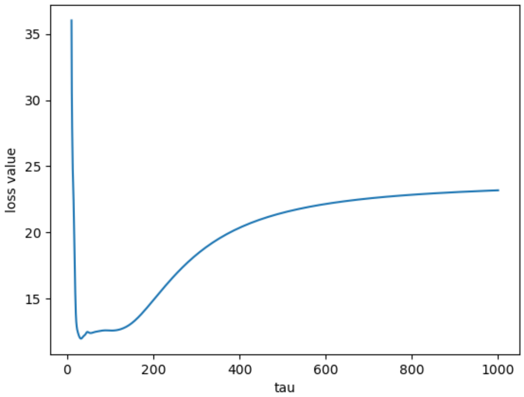

Figure 2: Loss value for each τ

Proof.

$$
\begin{aligned}
a^{(i)} &= \frac{e^{-\frac{\|x-x^{(i)}\|^2}{2\tau^2}}}{\sum_j e^{-\frac{\|x-x^{(j)}\|^2}{2\tau^2}}} \\
\frac{1}{a^{(i)}} &= \frac{\sum_j e^{-\frac{\|x-x^{(j)}\|^2}{2\tau^2}}}{e^{-\frac{\|x-x^{(i)}\|^2}{2\tau^2}}} \\
&= 1 + \frac{\sum_{i\neq j} e^{-\frac{\|x-x^{(j)}\|^2}{2\tau^2}}}{e^{-\frac{\|x-x^{(i)}\|^2}{2\tau^2}}} \\
&= 1 + \sum_{i\neq j} \frac{e^{-\frac{\|x-x^{(j)}\|^2}{2\tau^2}}}{e^{-\frac{\|x-x^{(i)}\|^2}{2\tau^2}}} \\
&= 1 + \sum_{i\neq j} e^{\frac{\|x-x^{(i)}\|^2 - \|x-x^{(j)}\|^2}{2\tau^2}}
\end{aligned}
$$

$$
\begin{aligned}
when \quad & \tau \to 0, \frac{1}{a^{(i)}} \to +\infty, a^{(i)} \to 0, A \to 0 \\
when \quad & \tau \to +\infty, \frac{1}{a^{(i)}} \to 1 + \sum_{i\neq j} 1 = N, a^{(i)} \to \frac{1}{N}, A \to \frac{1}{N} I
\end{aligned}
$$

$$
\begin{aligned}
\because w^* &= \frac{X^{\mathrm{T}}Ay}{X^{\mathrm{T}}AX + \lambda I} \\
\therefore loss &= (\hat{y} - y)^2 \\
&= (X^{\mathrm{T}}w^* - y)^2 \\
&= (\frac{X^{\mathrm{T}}AXy}{X^{\mathrm{T}}AX + \lambda I} - y)^2 \\
&= (\frac{\lambda I y}{X^{\mathrm{T}}AX + \lambda I})^2
\end{aligned}
$$

$$
\begin{aligned}
&when \quad \tau \to 0, A \to 0, loss \to 0 \\
&when \quad \tau \to +\infty, A \to \frac{1}{N}I, loss \to (\frac{\lambda I y}{\frac{1}{N}X^{\mathrm{T}}IX + \lambda I})^2 = (\frac{N\lambda y}{X^{\mathrm{T}}X + \lambda})^2
\end{aligned}
$$

□

3 Q3: Mini-batch SGD gradient estimator

3.1 Mini-batch mean value proof

Given a set $\{a_1, \dots, a_n\}$ and random mini-batchs $\mathcal{I}$ of size m, show that

$$\mathbb{E}_{\mathcal{I}}[\frac{1}{m}\sum_{i\in\mathcal{I}} a_i] = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{4}$$

Proof. As a_i is randomly sampled

$$
\begin{aligned}
p(a_1) = p(a_2) = \cdots = p(a_i) = \cdots = p(a_n) \\
\mathbb{E}(a_1) = \mathbb{E}(a_2) = \cdots = \mathbb{E}(a_i) = \cdots = \mathbb{E}(a_n)
\end{aligned}
$$

$$
\begin{aligned}
\therefore \mathbb{E}_{\mathcal{I}}[\frac{1}{m}\sum_{i\in\mathcal{I}} a_i] &= \frac{1}{m}\mathbb{E}_{\mathcal{I}}[\sum_{i\in\mathcal{I}} a_i] \\
&= \frac{1}{m}[\mathbb{E}(a_1) + \cdots + \mathbb{E}(a_m)] \\
&= \frac{1}{m} * [m * \mathbb{E}(a)] \\
&= \mathbb{E}(a) \\
&= \frac{1}{n}\sum_{i=1}^{n} a_i
\end{aligned}
$$

□

3.2 Mini-batch gradient proof

Show that $\mathbb{E}_{\mathcal{I}}[\nabla L_{\mathcal{I}}(x, y, \theta)] = \nabla L(x, y, \theta)$

Proof.

$$\begin{aligned}
\mathbb{E}_{\mathcal{I}}[\nabla L_{\mathcal{I}}(x,y,\theta)] &= \mathbb{E}_{\mathcal{I}}[\nabla(\frac{1}{m}\sum_{i\in\mathcal{I}}\ell(x^{(i)},y^{(i)},\theta))] \\
&= \frac{1}{m}\mathbb{E}_{\mathcal{I}}[\nabla(\sum_{i\in\mathcal{I}}\ell(x^{(i)},y^{(i)},\theta))] \\
&= \frac{1}{m}\mathbb{E}_{\mathcal{I}}[\sum_{i\in\mathcal{I}}\nabla\ell(x^{(i)},y^{(i)},\theta)] \\
&= \frac{1}{m}\mathbb{E}_{\mathcal{I}}[\sum_{i\in\mathcal{I}}\ell(x^{(i)},y^{(i)},\theta+\Delta\theta)-\ell(x^{(i)},y^{(i)},\theta)] \\
&= \frac{1}{m}\mathbb{E}_{\mathcal{I}}[(\ell(x^{(1)},y^{(1)},\theta+\Delta\theta)-\ell(x^{(1)},y^{(1)},\theta))+\dots \\
&\quad +(\ell(x^{(m)},y^{(m)},\theta+\Delta\theta)-\ell(x^{(m)},y^{(m)},\theta))] \\
&= \frac{1}{m}[\mathbb{E}(\ell(x^{(1)},y^{(1)},\theta+\Delta\theta)-\ell(x^{(1)},y^{(1)},\theta))+\dots \\
&\quad +\mathbb{E}(\ell(x^{(m)},y^{(m)},\theta+\Delta\theta)-\ell(x^{(m)},y^{(m)},\theta))] \\
&= \frac{1}{m}[m*\mathbb{E}(\ell(x,y,\theta+\Delta\theta)-\ell(x,y,\theta))] \\
&= \mathbb{E}(\ell(x,y,\theta+\Delta\theta)-\ell(x,y,\theta)) \\
&= \mathbb{E}(\nabla\ell(x,y,\theta)) \\
\nabla L(x,y,\theta) &= \nabla[\frac{1}{n}\sum_{i=1}^{n}\ell(x^{(i)},y^{(i)},\theta)] \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla\ell(x^{(i)},y^{(i)},\theta) \\
&= \mathbb{E}(\nabla\ell(x,y,\theta)) \\
&\therefore \mathbb{E}_{\mathcal{I}}[\nabla L_{\mathcal{I}}(x,y,\theta)] = \nabla L(x,y,\theta)
\end{aligned}$$

□

3.3 Mini-batch expectations

As the proofs above, expectations of mean value and gradient of mini-batch is just the same like the ones of full-batch, which means different data choices of mini-batch and full-batch return the same result.

3.4 Gradient implementation

For loss funtion $\ell(x,y,w)=(y-w^{\mathrm{T}}x)^2$, its gradient can be presented as:

$$\nabla\ell(x,y,w)=2x(w^{\mathrm{T}}x-y) \tag{5}$$

In the code implementation, gradient is computed by the formula above.

3.5 Mini-batch gradient VS true gradient

In this part, we set $m=50$ and $K=500$. To compare the gradient value between mini-batch and the true gradient (entire data set), we use following metrics:

- squared distance metric: 36232.487235814442

- cosine similarity: 0.999997836597549l5

For these two metrics, cosine similarity is a more meaningful measure in this case. Square distance metric is hard to judge just because we can not tell the difference simply from one value. However, for cosine similarity valued in $[0, 1]$, value tending to 1 means the gradients are very similar. We can easily judge this.

3.6 Affect of batch size

Here, we give out the figures of $\log \tilde{\sigma}_j$ against $\log(m)$ on all features to show the influence of batch size om the gradient varience. It is shown as Figure 3.

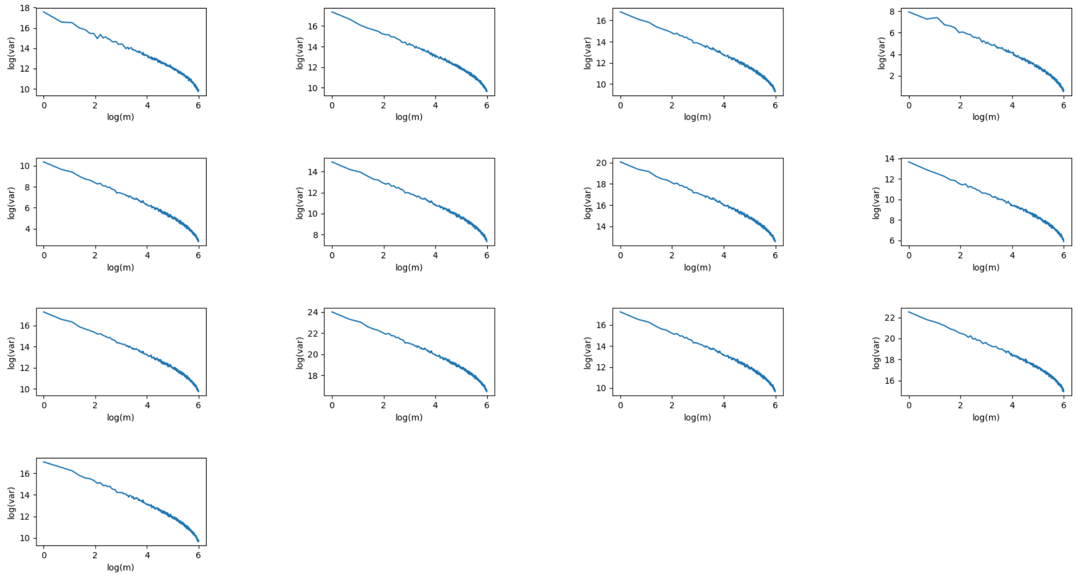

Figure 3: Log(m) against log(gradient variance) for each feature