DDA3020 Machine Learning

Assignment 1 Written & Report

----

1. Written Question

1.1

1) consider $\underset{1\times n}{y^T} X \underset{d\times 1}{w}$ is a scalar function

$$\underbrace{y^T X w}_{1\times d} = [X^T y]^T w = z^T w = [z_1, z_2, \dots, z_d]\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \underbrace{\sum_{i=1}^{d} z_i w_i}$$

(let $X^T y = z \in d\times 1$)

$$\therefore \frac{d\, y^T X w}{dw} = \frac{d\, z^T w}{dw} = \begin{bmatrix} \frac{\partial z^T w}{\partial w_1} \\ \frac{\partial z^T w}{\partial w_2} \\ \vdots \\ \frac{\partial z^T w}{\partial w_d} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix} = z = X^T y$$

2) $w^T w = \sum w_i^2$

$$\therefore \frac{d w^T w}{dw} = \begin{bmatrix} \frac{\partial w^T w}{\partial w_1} \\ \frac{\partial w^T w}{\partial w_2} \\ \vdots \\ \frac{\partial w^T w}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 2w_1 \\ 2w_2 \\ \vdots \\ 2w_d \end{bmatrix} = 2w$$

3) Let $w = [w_1, w_2, \dots w_d]^T$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & & & \vdots \\ x_{d1} & & \cdots & x_{dd} \end{bmatrix}$$

$$w^T X w = [\sum w_i x_{i1}, \sum w_i x_{i2}, \cdots, \sum w_i x_{id}] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

$$= w_1 \sum w_i x_{i1} + w_2 \sum w_i x_{i2} + \cdots + w_d \sum w_i x_{id}$$

$$= \sum_{i=1}^{d} w_i^2 \cdot x_{ii} + \sum_{i\neq j}\sum w_i w_j (x_{ij} + x_{ji}) \text{ is a scalar function}$$

$$\therefore \frac{d w^T X w}{dw} = \begin{bmatrix} \frac{\partial w^T X w}{\partial w_1} \\ \frac{\partial w^T X w}{\partial w_2} \\ \vdots \\ \frac{\partial w^T X w}{\partial w_3} \end{bmatrix} = \begin{bmatrix} 2w_1 x_{11} + \sum_{j\neq 1} w_j (x_{1j} + x_{j1}) \\ 2w_2 x_{22} + \sum_{j\neq 2} w_j (x_{2j} + x_{j2}) \\ \vdots \\ 2w_d x_{dd} + \sum_{j\neq d} w_j (x_{dj} + x_{jd}) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{d} w_j (x_{1j} + x_{j1}) \\ \sum_{j=1}^{d} w_j (x_{2j} + x_{j2}) \\ \vdots \\ \sum_{j=1}^{d} w_j (x_{dj} + x_{jd}) \end{bmatrix}$$

consider $(X + X^T)w$

$$[\{x_{ij} + x_{ji}\}] \cdot [w_1, w_2, \cdots w_d]^T$$

$$= \begin{bmatrix} \Sigma w_i (x_{1j} + x_{j1}) \\ \Sigma w_i (x_{2j} + x_{j2}) \\ \vdots \\ \Sigma w_i (x_{dj} + x_{jd}) \end{bmatrix} = \frac{dw^TXw}{dw}$$

$\square$

1.2

1.3

1.4

DDA3020 Machine Learning

Assignment 1 Written & Report

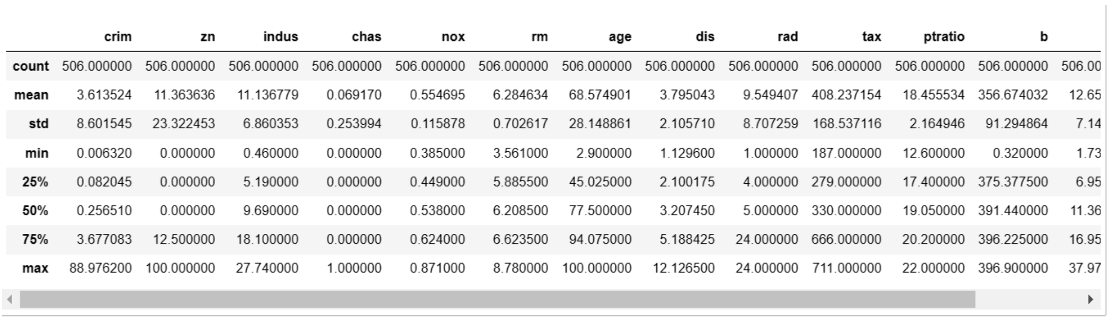
121090408 Ma Xinxian

2. Programming Question

Description: In this assignment, we need to use appropriate attributes in 'crim, zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, b, lstat' to predict the last attributes 'MEDV' by the linear regression model.

2.1. Loading data [Step1]

By using *pandas.isna().sum()*, we find that there is not incomplete data point in the dataset. And also *pandas.describe()* and *shape()* function are used to summarized the data: there are 506 data samples and 14 feature columns, get more information from the table below (not complete).



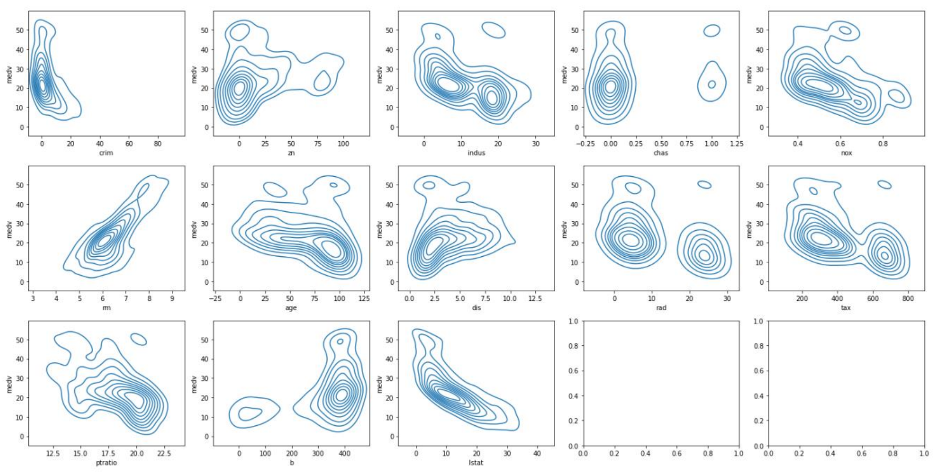
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.65
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.14
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.73
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.95
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.36
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.95
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.97

Besides, I guess 'lstat' is the most relevant attribute for MEDV due to their relatively similar characteristic and my personal understanding about MEDV.

2.2. Data visualized [Step2&3&4]

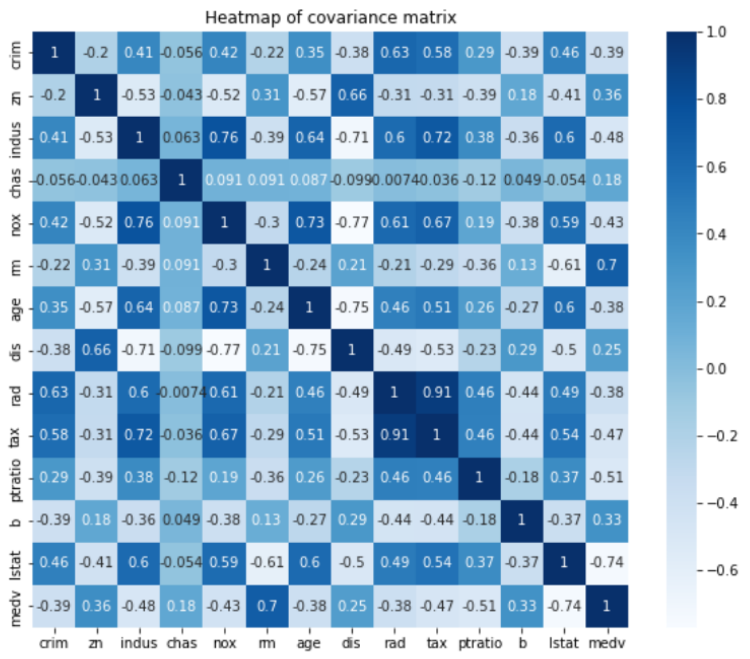
In order to correct prediction, we use seaborn to visualized the data:

I) Kdeplot() with bivariate distribution



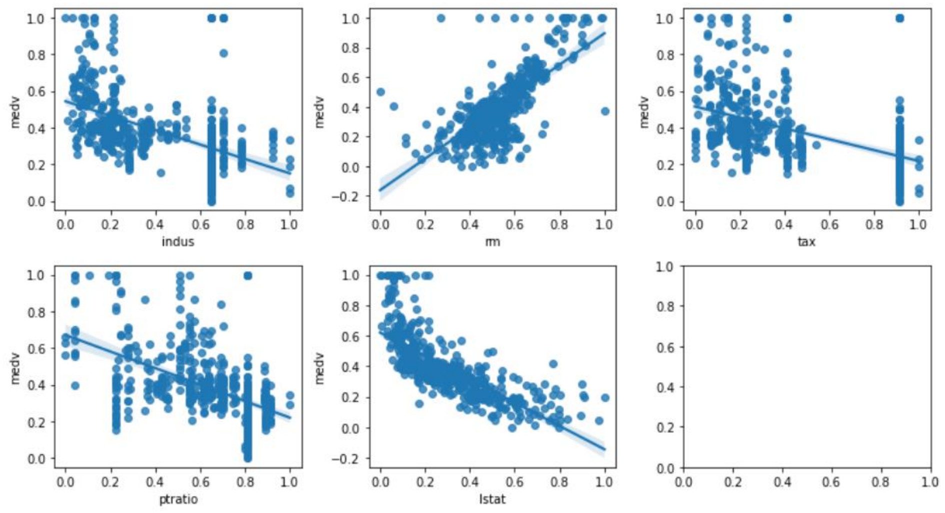
This bivariate distribution figures clearly display the relationship between different attributes. Considering that an oblate image indicates a potential linear relationship between them, we guess "crim", "nox", "rm", "age', "dis", "ptratio" and "lstat" are relatively higher relevant attributes for MEDV.

II) Heatmap() with pairwise correlation matrix



Just focus on the last column. Larger the absolute value of corrcoeffient is, more linear correlation with MEDV. We set 0.45 be a threshold value. As a result, "indus", "rm", "tax", "ptratio" and "lstat" are selected as the good indications of using as predictors.

III) Regplot() plot the relevance against MEDV with 95% confidence interval



They all have a relatively strong linear relationship with MEDV (You can see it more obvious by using *estimator=mean()* in my code file). So our selection is good.

2.3. Linear model(Step5&6)

I) Linear hypothesis function:

for a given data x with d dimension:

$$f_{\boldsymbol{w},b}(x) = \boldsymbol{x}^T\boldsymbol{w} + b$$

where **w** is a d-dimensional vector of parameters, bias b is a real number.

for a given data set **X** with shape m*(d+1):

We can modify first equation into:

$$f_{\boldsymbol{w}}(\boldsymbol{X}) = X\boldsymbol{w}$$

where **w** contains the bias and becomes a (d+1) *1 matrix. (m is the number of data) and the output is a m*1 result matrix.

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{m1} & x_{m2} & \dots & x_{md} \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \dots \\ w_d \end{bmatrix}$$

II) Cost function:

For a given dataset defined in I), the cost function for linear regression model J($\boldsymbol{w}$) is defined as:

$$J(\boldsymbol{w}) = \frac{1}{2m}(\boldsymbol{Xw}-\boldsymbol{y})^2 = \frac{1}{2m}(\boldsymbol{Xw}-\boldsymbol{y})^T(\boldsymbol{Xw}-\boldsymbol{y})$$

Where y is a m*1 result matrix, the output is somehow an average error.

We divide m in the equation to make it easier to get RMSE later. In the code, the cost function is defined in the function called Lossf(X, w, y)

III) Split data

IV) Training with Gradient Descent method (hyperparameter settings):

-According to Gradient Descent algorithms, **w** is updated with following equations:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \quad , \quad \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{m}\mathbf{X}^T(\mathbf{Xw}-\mathbf{y})$$

Where α is called step-size or learning rate.

Stopping criterion: practically, when the **iteration times** is reached.

(There are also some other ways, but I choose it here since it is practical)

-This leads to our hyperparameter settings:

1. **Initial parameter matrix** $\boldsymbol{w}$ is randomly set by using *init_w*() function:

```
def init_w():
    np.random.seed(1674)
    return np.random.randint(-15, 15, (6, 1))
```

(-15, 15) can be changed. This $\boldsymbol{w}$ is the initial parameter matrix being updated.

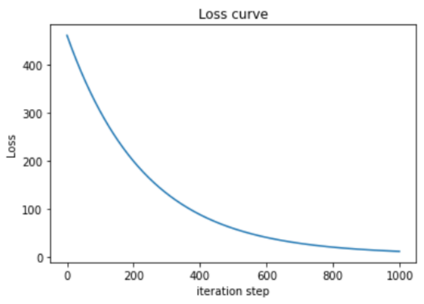
2. **Step size $\boldsymbol{\alpha}$** is set to 10^{-3} as default. It will affect the learning rate of the model. One can also use exact line search and backtracking line search method.
3. **Iteration times** is set to 20 as default. This is related to the stopping criterion.

-We start training now and get the error in ***RMSE***:

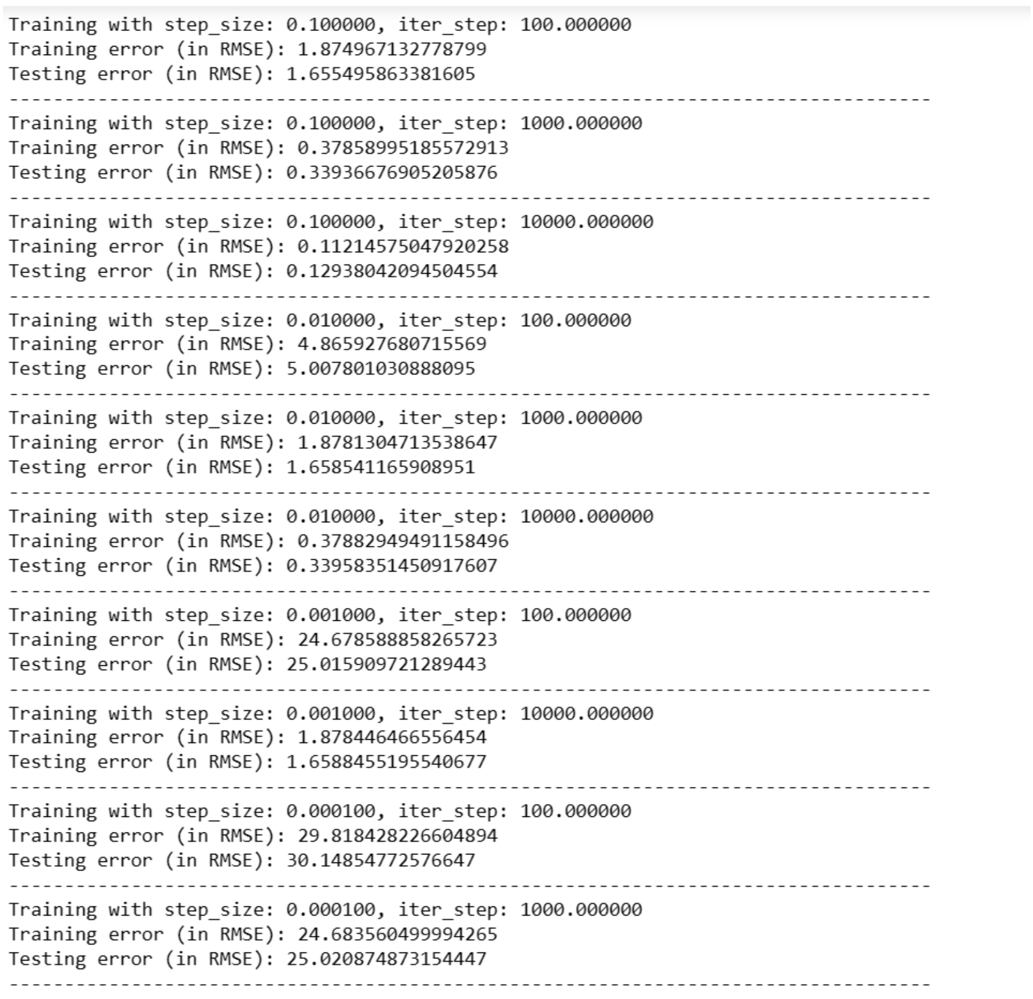
$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(Y_i - f(x_i))^2} \quad \text{so} \quad RMSE = \sqrt{2 \times J(\mathbf{w})}$$

Result:

```
Training error (in RMSE): 4.918903212701808
Testing error (in RMSE): 5.065365234927568
```



-Repeat above process for 10 times with different parameters (step size and iteration times).



```
Training with step_size: 0.100000, iter_step: 100.000000
Training error (in RMSE): 1.874967132778799
Testing error (in RMSE): 1.655495863381605
---------------------------------------------------------------------------
Training with step_size: 0.100000, iter_step: 1000.000000
Training error (in RMSE): 0.37858995185572913
Testing error (in RMSE): 0.33936676905205876
---------------------------------------------------------------------------
Training with step_size: 0.100000, iter_step: 10000.000000
Training error (in RMSE): 0.11214575047920258
Testing error (in RMSE): 0.12938042094504554
---------------------------------------------------------------------------
Training with step_size: 0.010000, iter_step: 100.000000
Training error (in RMSE): 4.865927680715569
Testing error (in RMSE): 5.007801030888095
---------------------------------------------------------------------------
Training with step_size: 0.010000, iter_step: 1000.000000
Training error (in RMSE): 1.8781304713538647
Testing error (in RMSE): 1.658541165908951
---------------------------------------------------------------------------
Training with step_size: 0.010000, iter_step: 10000.000000
Training error (in RMSE): 0.37882949491158496
Testing error (in RMSE): 0.33958351450917607
---------------------------------------------------------------------------
Training with step_size: 0.001000, iter_step: 100.000000
Training error (in RMSE): 24.678588858265723
Testing error (in RMSE): 25.015909721289443
---------------------------------------------------------------------------
Training with step_size: 0.001000, iter_step: 10000.000000
Training error (in RMSE): 1.878446466556454
Testing error (in RMSE): 1.6588455195540677
---------------------------------------------------------------------------
Training with step_size: 0.000100, iter_step: 100.000000
Training error (in RMSE): 29.818428226604894
Testing error (in RMSE): 30.14854772576647
---------------------------------------------------------------------------
Training with step_size: 0.000100, iter_step: 1000.000000
Training error (in RMSE): 24.683560499994265
Testing error (in RMSE): 25.020874873154447
---------------------------------------------------------------------------
```

Influence of different parameters on ***RMSE***:

1. **Step size $\boldsymbol{\alpha}$:**
   focus on iteration 1,4,7,9 where iteration times are 100, we find that as $\boldsymbol{\alpha}$ become smaller, the error in training and testing are become larger. For other iteration times like 1000, 10000, we get the same observation but the difference gap become smaller. This is because the $\boldsymbol{\alpha}$ affects the velocity of learning the parameter. Considering that we make $\boldsymbol{\alpha}$ be a constant in our training, the larger it is, larger distance will the parameter move on the better position, the RMSE error **<u>may</u>** become smaller faster. But this may not correct since if $\boldsymbol{\alpha}$ is too large, it may move over and hard to move to the optimal $\boldsymbol{w}$**.** We should be care about it in different specific training or use other method like exact line search and backtracking line search.
2. **Iteration times:**
   focus on iteration 1,2,3 where $\boldsymbol{\alpha}$ are 100000, we find that as **iteration times** become larger, the error in training and testing are become smaller. For other $\boldsymbol{\alpha}$, we get the same observation. This is because **iteration times** is related to the stopping criterion. If **iteration times** is large, more work will w do to approach the optimal one, then the error **<u>may</u>** be smaller. But as we discuss in $\boldsymbol{\alpha}$, its influence also depends other parameter setting.