به نام خدا

گزارش سوال ۶ تکلیف اول
مهدی کافی ۹۹۲۱۰۷۵۳

در ابتدا داده‌ها را خوانده و برای آشنایی با داده و کنترل کیفیت به صورت زیر عمل می‌کنیم.

```python
# Packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Load Data
train_data  = pd.read_csv("regression/train.csv")
test_data = pd.read_csv("regression/test.csv")
train_data
```

	age	gender	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
995	39	female	23.275	3	no	northeast	7986.47525
996	39	female	34.100	3	no	southwest	7418.52200
997	63	female	36.850	0	no	southeast	13887.96850
998	33	female	36.290	3	no	northeast	6551.75010
999	36	female	26.885	0	no	northwest	5267.81815

1000 rows × 7 columns

```python
# Quality Control
train_data.isnull().sum()
```

```
age         0
gender      0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

```python
train_data.isna().sum()
```

```
age         0
gender      0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

```python
train_data.describe()
```

	age	bmi	children	charges
count	1000.000000	1000.00000	1000.000000	1000.000000
mean	39.615000	30.86338	1.080000	13075.755883
std	14.153908	6.04744	1.198765	11985.924552
min	18.000000	15.96000	0.000000	1121.873900
25%	27.000000	26.60000	0.000000	4719.683425
50%	40.000000	30.59000	1.000000	9283.021300
75%	52.000000	35.11250	2.000000	15882.795437
max	64.000000	50.38000	5.000000	63770.428010

سپس داده آموزش و تست را به صورت Integer Encoding و One Hot Encoding در می‌آوریم.

```python
# Integer Encoding and One Hot Encoding
train_data["gender"].replace(to_replace=["female", "male"], value=[0, 1], inplace=True)
train_data["smoker"].replace(to_replace=["yes", "no"], value=[1, 0], inplace=True)
train_data = pd.get_dummies(train_data, prefix="region_", columns=["region"])
headers = train_data.columns.tolist()
headers = headers[:5] + headers[6:] + [headers[5]]
train_data = train_data[headers]
train_data
```

	age	gender	bmi	children	smoker	region__northeast	region__northwest	region__southeast	region__southwest	charges
0	19	0	27.900	0	1	0	0	0	1	16884.92400
1	18	1	33.770	1	0	0	0	1	0	1725.55230
2	28	1	33.000	3	0	0	0	1	0	4449.46200
3	33	1	22.705	0	0	0	1	0	0	21984.47061
4	32	1	28.880	0	0	0	1	0	0	3866.85520
...	...	...	...	...	...	...	...	...	...	...
995	39	0	23.275	3	0	1	0	0	0	7986.47525
996	39	0	34.100	3	0	0	0	0	1	7418.52200
997	63	0	36.850	0	0	0	0	1	0	13887.96850
998	33	0	36.290	3	0	1	0	0	0	6551.75010
999	36	0	26.885	0	0	0	1	0	0	5267.81815

1000 rows × 10 columns

۱.۶

در این بخش به دلیل اینکه basis function برای ویژگی وزن به صورت x^2 است و برای سایر ویژگی‌ها به صورت همانی است، ماتریس داده‌ها را به ماتریس ϕ تبدیل می‌کنیم و به جای سن افراد، مجذور سن را قرار می‌دهیم.

w_hat = inverse(phi.T @ phi) @ phi.T @ y equals pseudo_inverse(phi) @ y

```python
#convert X features to phi (basis functions) matrix
phi_train = X_train.copy()
phi_test = X_test.copy()
phi_train.loc[:, "age"] = X_train.loc[:, "age"] ** 2
phi_test.loc[:, "age"] = X_test.loc[:, "age"] ** 2
```

فرم بسته بردار وزن‌ها به صورت زیر است که در ادامه آن را پیاده سازی کردیم.

$$w = (\boldsymbol{\varphi}^T \boldsymbol{\varphi})^{-1} \boldsymbol{\varphi}^T y$$

```python
# Calculate w_hat which is argmin_w(J(w))
w_hat = np.linalg.pinv(X_train.values) @ y_train
w_hat = w_hat.reshape((w_hat.shape[0], 1))
w_hat
```

```
array([[-10265.47558833],
       [   264.25843027],
       [  -288.53386146],
       [   339.90865802],
       [   410.23628453],
       [ 23832.3784891 ],
       [ -1817.78531587],
       [ -2257.68184123],
       [ -3109.07519302],
       [ -3080.93323821]])
```

```python
# Predict outputs for test features
def predict(X, w):
    y_hat = X @ w
    return y_hat
y_hat = predict(phi_test, w_hat)
y_hat
```

	0
0	257488.247054
1	173525.304751
2	147347.556522
3	605277.776921
4	578308.161122
...	...
145	714114.319267
146	972676.788739
147	104030.099040
148	793465.081297
149	464107.813538

تابع predict با ضرب کردن ماتریس داده ورودی در بردار وزن ورودی، بردار تخمین خروجی را محاسبه می‌کند.
در نهایت تابع cost مقدار sum of squared error را محاسبه می‌کند. با استفاده از این تابع مقدار هزینه را برای تست، با وزن به دست آمده در مرحله قبل محاسبه می‌کنیم.

```python
def cost(y_test, y_hat):
    error = y_test - y_hat
    squared_error = error ** 2
    sse = squared_error.sum()
    return sse
cost(y_test, y_hat[0])
```

```
43472466662997.33
```

۲.۶

در این بخش باید الگوریتم SGD را پیاده‌سازی کنیم برای این منظور در ابتدا داده‌های ماتریس φ را استاندارد می‌کنیم که میانگین ستون‌های عددی برابر با صفر با انحراف معیار ۱ می‌شود.

```python
# Standardizing Data
headers = phi_train.columns.to_list()
for col in ["age", "bmi", "children"]:
    min_col = phi_train[col].min()
    max_col = phi_train[col].max()
    mean = phi_train[col].mean()
    std = phi_train[col].std()
    phi_train[col] = (phi_train[col] - mean) / std
for col in ["age", "bmi", "children"]:
    min_col = phi_test[col].min()
    max_col = phi_test[col].max()
    mean = phi_test[col].mean()
    std = phi_test[col].std()
    phi_test[col] = (phi_test[col] - mean)/std
phi_train.describe()
```

	Ones	age	gender	bmi	children	smoker	region__northeast	region__northwest	region__southeast	region__southwest
count	1000.0	1.000000e+03	1000.000000	1.000000e+03	1.000000e+03	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	1.0	2.858824e-17	0.505000	1.188338e-14	-8.082424e-17	0.196000	0.247000	0.231000	0.278000	0.244000
std	0.0	1.000000e+00	0.500225	1.000000e+00	1.000000e+00	0.397167	0.431483	0.421683	0.448238	0.429708
min	1.0	-1.268113e+00	0.000000	-2.464411e+00	-9.009274e-01	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.0	-9.128089e-01	0.000000	-7.049892e-01	-9.009274e-01	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.0	-1.486849e-01	1.000000	-4.520590e-02	-6.673536e-02	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.0	8.198490e-01	1.000000	7.026311e-01	7.674566e-01	0.000000	0.000000	0.000000	1.000000	0.000000
max	1.0	2.041044e+00	1.000000	3.227253e+00	3.270033e+00	1.000000	1.000000	1.000000	1.000000	1.000000

سپس باید با نشان دادن هر یک از نمونه‌های داده آموزش به الگوریتم، بردار وزن‌ها را آپدیت کنیم و برای هر بردار وزن به دست‌آمده مقدار هزینه برای داده تست را محاسبه می‌کنیم و در انتها بهترین بردار وزن که کمترین هزینه روی داده تست را ایجاد می‌کرد را به دست می‌آوریم.

```python
n, d = phi_train.shape
learning_rate = 0.001
w = np.zeros((d, 1))
n_iteration = 100_000
epoch = 1
best_weight = None
best_error = float('inf')
for t in range(1, n_iteration+1):
    i = t % n
    if i == 0:
        print(f">>epoch {epoch} | cost {cost(y_test, predict(phi_test, w)[0])}")
        epoch += 1
    x = phi_train.iloc[i].values.reshape((1, d))
    y = y_train.iloc[i]
    w = w + learning_rate * x.T * (y - x@w)
    y_hat = predict(phi_test, w)
    error = cost(y_test, y_hat[0])
    if error < best_error:
        best_error = error
        best_weight = w
print("minimum cost:", best_error)
print("best weights:\n", best_weight)
```

```
>>epoch 1 | cost 19708667032.7261
>>epoch 2 | cost 14786311466.003078
>>epoch 3 | cost 12554746966.996471
>>epoch 4 | cost 11023528167.731506
>>epoch 5 | cost 9898128841.502113
>>epoch 6 | cost 9061794702.829035
>>epoch 7 | cost 8438985530.247275
>>epoch 8 | cost 7974629733.644092
>>epoch 9 | cost 7627860380.4411545
>>epoch 10 | cost 7368367590.827873
>>epoch 11 | cost 7173712374.193759
>>epoch 12 | cost 7027293667.82103
>>epoch 13 | cost 6916825117.27726
>>epoch 14 | cost 6833204749.430954
```

```
>>epoch 93 | cost 6548364784.930328
>>epoch 94 | cost 6548364699.970087
>>epoch 95 | cost 6548364626.874551
>>epoch 96 | cost 6548364563.986748
>>epoch 97 | cost 6548364509.881138
>>epoch 98 | cost 6548364463.331261
>>epoch 99 | cost 6548364423.281958
>>epoch 100 | cost 6548364388.825413
minimum cost: 6418436440.062544
best weights:
 [[ 6941.1312347 ]
 [ 3727.93132529]
 [ -242.03568827]
 [ 2034.61956002]
 [  514.50274386]
 [23920.58380653]
 [ 2489.63093586]
 [ 2017.48556248]
 [ 1229.641901  ]
 [ 1204.37283536]]
```

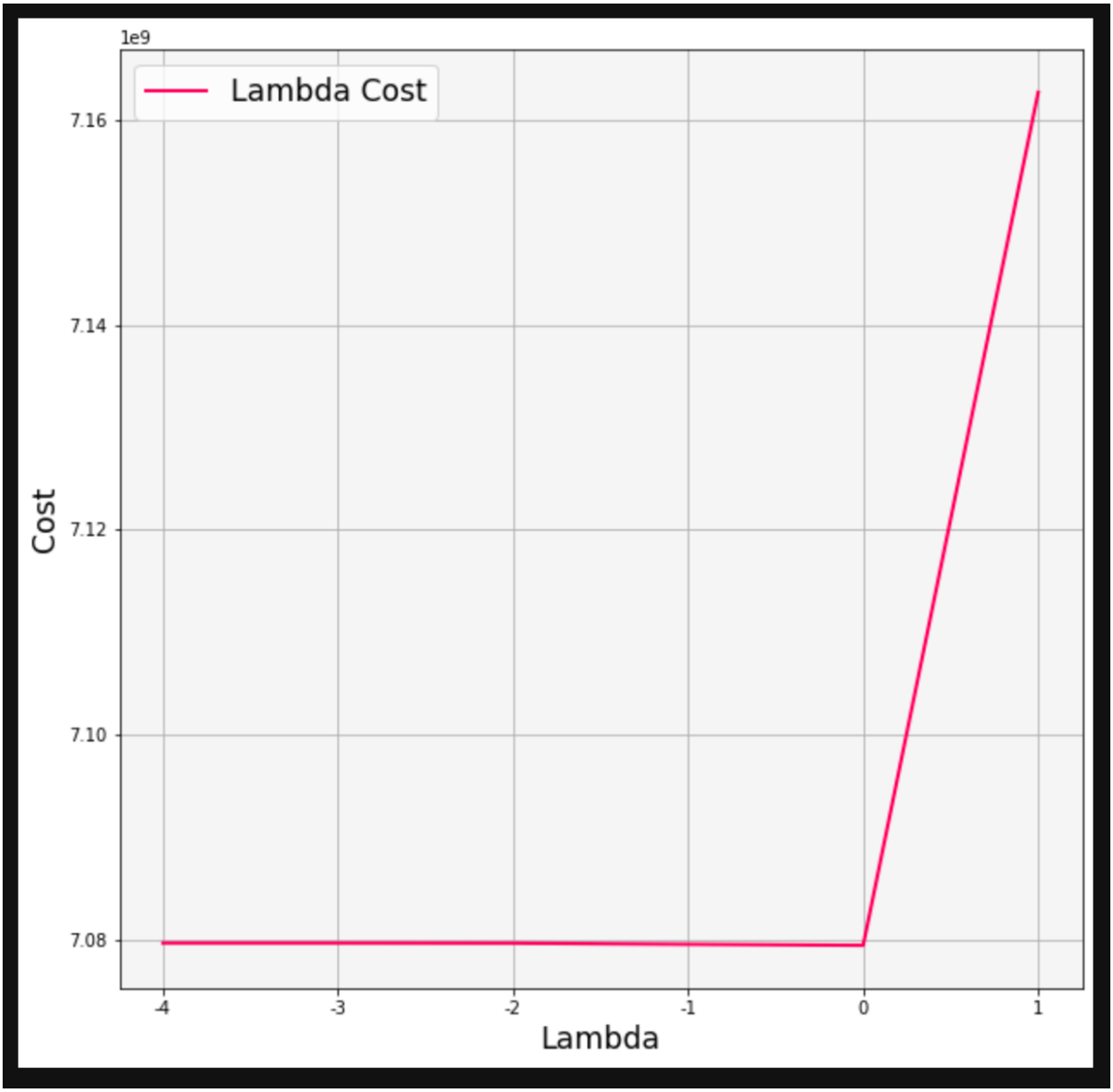
۳.۶

در این بخش داده را به ۵ قسمت تقسیم می‌کنیم و هر بار یکی از بخش‌ها به عنوان validation و بقیه بخش‌ها به عنوان داده آموزش انتخاب می‌شوند و با فرم بسته، بردار وزن‌ها را محاسبه می‌کنیم و هزینه را روی بخش validation محاسبه می‌کنیم و در نهایت ۵ هزینه به دست آمده به ازای هر یک از بخش‌های validation را میانگین میگیریم و این مقدار را به عنوان هزینه به ازای مقدار متفاوت ضریب رگولاریزیشن نگهداری می‌کنیم و سپس ضریبی که کمترین هزینه را منجر میشد، به دست می‌آوریم.

```python
n, d = phi_train.shape
boundaries = np.linspace(0, n, num= 6, dtype=np.int16)
lambda_costs = {}
lambda_weights = {}
for lambda_exp in [-4, -3, -2, -1, 0, 1]:
    costs = []
    for idx in range(len(boundaries) - 1):
        c_x_valid = phi_train.iloc[boundaries[idx]: boundaries[idx+1]]
        c_x_train = phi_train.drop(np.r_[boundaries[idx]:boundaries[idx+1]], axis=0)
        c_y_valid = y_train.iloc[boundaries[idx]: boundaries[idx+1]]
        c_y_train = y_train.drop(np.r_[boundaries[idx]:boundaries[idx+1]], axis=0)
        w = np.linalg.inv((c_x_train.T @ c_x_train) + (10**lambda_exp) * np.identity(d)) @ c_x_train.T @ c_y_
        w = w.values.reshape((d, 1))
        c_y_hat = predict(c_x_valid, w)[0]
        c_cost = cost(c_y_valid, c_y_hat)
        costs.append(c_cost)
    lambda_costs[str(lambda_exp)] = np.mean(costs)
    lambda_weights[str(lambda_exp)] = w
lambda_costs
```

```
{'-4': 7079711287.904966,
 '-3': 7079710089.165428,
 '-2': 7079698203.0013485,
 '-1': 7079589345.059461,
 '0': 7079485149.329951,
 '1': 7162700538.615109}
```

در ادامه نمودار هزینه به ازای مقادیر مختلف ضریب را رسم می‌کنیم.



سپس دیده می‌شود که کمترین هزینه به ازای ضریب صفر به دست می‌آید و با وزن‌های به دست آمده از این ضریب مقدار هزینه را برای داده‌های آموزش و تست محاسبه کرده‌ایم.

```python
best_lambda = min(lambda_costs, key=lambda_costs.get)
best_lambda
```

```
'0'
```

```python
w = lambda_weights[best_lambda]
train_cost = cost(y_train, predict(X_train, w)[0])
test_cost = cost(y_test, predict(X_test, w)[0])
print("Train cost:", train_cost)
print("Test cost:", test_cost)
```

```
Train cost: 47161505628584.99
Test cost: 6924324901770.429
```