Big Data-Climate Change

1. Problem one—Creating Your First Model

(1) Implement a function closed_form_1 that computes this closed form solution given the features X, labels Y(using Python or Matlab)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm

df = pd.read_csv('climate_change_1.csv')

def closed_form_1():
    traindata = df[df.Year<=2006]
    testdata = df[df.Year>2006]
    X = traindata[['MEI','CO2','CH4','N2O','CFC-11','CFC-12','TSI','Aerosols']]
    Y = traindata['Temp']
    est = sm.OLS(Y, sm.add_constant(X)).fit()
    print(est.summary())
    print(est.params)

closed_form_1()
```

(2) Write down the mathematical formula for the linear model and evaluate the model R^2 on the training set and the testing set.

$$\begin{aligned} \mathrm{y} = -124.594260 + 0.064205 \times \mathrm{MEI} + 0.006457 \times \mathrm{CO2} + 0.000124 \times \mathrm{CH4} \\ - 0.016528 \times \mathrm{N2O} - 0.006630 \times \mathrm{CFC} - 11 + 0.003808 \times \mathrm{CFC} - 12 \\ + 0.093141 \times \mathrm{TSI} - 1.537613 \times \mathrm{Aerosols} \end{aligned}$$

The R^2 on training set is 0.744

The R^2 on testing set is 0.425

(3) Which variables are significant in the model?

If the p-value is below 0.05, we will choose MEI, CO2, CFC-11, CFC-12, TSI, Aerosols.

(4) Write down the necessary conditions for using the closed form solution. And you can apply it to the dataset climate_change_2.csv, explain the solution is unreasonable.

Necessary conditions: MEI, CO2, CH4, N2O, CFC-11, CFC-12, TSI, Aerosols should not be correlated with each other.

Unreasonable: N2O and CFC-11 are greenhouse gases and the regression coefficients of them should be positive but the results are negative. Therefore, N2O and CFC-11 are correlated with other variables in the data set.

2. Problem two—Regularization

Regularization is a method to boost robustness of model, including L_1 regularization and L_2 regularization.

(1) Please write down the loss function for linear model with L_1 regularization and L_2 regularization, respectively.

L_1 Regularization:

$$\mathrm{C} = C_0 + \frac{\lambda}{n}\sum_{\omega}|\omega|$$

L_2 Regularization:

$$\mathrm{C} = C_0 + \frac{\lambda}{2n}\sum_{\omega}\omega^2$$

C_0 represents the original cost function.

(2) The closed form solution for linear model with L_2 regularization.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm

df = pd.read_csv('climate_change_1.csv')

def closed_form_2():
    lamb = 0.5
    df['const'] = 1
    traindata = df[df.Year<=2006]
    testdata = df[df.Year>2006]
    X = traindata[['MEI','CO2','CH4','N2O','CFC-11','CFC-12','TSI','Aerosols','const']]
    Y = traindata['Temp']

    Theta2 = np.dot(
        np.dot(np.linalg.inv((np.dot(X.T, X) + lamb * np.eye(X.shape[1]))), X.T), Y)
    print(Theta2)

closed_form_2()
```

Result:

Item	θ
MEI	4.55768014e-02
CO2	7.80443532e-03

CH4	1.95701031e-04
N2O	-1.64893727e-02
CFC-11	-6.38359095e-03
CFC-12	3.74766007e-03
TSI	1.44919104e-03
Aerosols	-3.65599605e-01
Const	-4.68953239e-03

(3) Compare the two solutions in problem 1 and problem 2 and explain the reason why linear model with L_2 regularization is robust.

The reason why the L_2 method is better than before is that the size of the coefficient of the constant term of the L_2 method is basically the same as the value of the temp result, which will not cause a large deviation of the result due to the error of the constant term estimation.

(4) You can change the regularization parameter λ to get different solutions for this problem.

λ	R^2 of Training Set	R^2 of Testing Set
10	0.6746079231515448	0.9408716921954439
1	0.6794692110104662	0.8467501178134881
0.1	0.6944684109361836	0.6732879125422909
0.01	0.7116529617244923	0.5852763146622774
0.001	0.7148330433597858	0.5625217958135218

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm

df = pd.read_csv('climate_change_1.csv')

def R_square(Y_pred, Y):
        ESS = np.sum((Y_pred - Y.mean())**2)
        TSS = np.sum((Y - Y.mean())**2)
        R2 = ESS / TSS
        print(R2)
```

```
def closed_form_2():
    lamb = 0.001
    df['const'] = 1
    traindata = df[df.Year<=2006]
    testdata = df[df.Year>2006]
    X = traindata[['MEI','CO2','CH4','N2O','CFC-11','CFC-12','TSI','Aerosols','const']]
    Y = traindata['Temp']

    Theta2 = np.dot(
        np.dot(np.linalg.inv((np.dot(X.T, X) + lamb * np.eye(X.shape[1]))), X.T), Y)
    print(Theta2)

    Y_pred = np.dot(X,Theta2)
    R_square(Y_pred,Y)

closed_form_2()
```

3. Problem three—Feature Selection

(1) From Problem 1， you can know which variables are significant, therefore you can use less variables to train model. For example, remove highly correlated and redundant features. You can propose a workflow to select feature.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm

df = pd.read_csv('climate_change_1.csv')
climate_change_1_train = df[df.Year<=2006]
climate_change_1_test = df[df.Year>2006]

climate_change_1_train_X = climate_change_1_train.iloc[:,2:10]
climate_change_1_train_Y = climate_change_1_train.iloc[:,10]

print(climate_change_1_train_X.corr())
#80% above is considered as highly related
#CO2 CH4 N2O CFC-12 are highly correlated with each other
#CFC-11 CFC-12 are correlated with each other
```

```
              MEI       CO2       CH4       N2O    CFC-11
MEI       1.000000 -0.041147 -0.033419 -0.050820  0.069000
CO2      -0.041147  1.000000  0.877280  0.976720  0.514060
CH4      -0.033419  0.877280  1.000000  0.899839  0.779904
N2O      -0.050820  0.976720  0.899839  1.000000  0.522477
CFC-11    0.069000  0.514060  0.779904  0.522477  1.000000
CFC-12    0.008286  0.852690  0.963616  0.867931  0.868985
TSI      -0.154492  0.177429  0.245528  0.199757  0.272046
Aerosols  0.340238 -0.356155 -0.267809 -0.337055 -0.043921
```

#delete N2O,CH4,CFC-12
climate_change_1_train_X_new = climate_change_1_train.loc[:,['MEI','CO2','CFC-11','TSI','Aerosols']]

# 1.2 run a regression remove not significant with alpha = 0.01
import statsmodels.api as sm
from statsmodels import regression
climate_change_1_train_X_new=sm.add_constant(climate_change_1_train_X_new)
model=sm.OLS(climate_change_1_train_Y,climate_change_1_train_X_new)
res=model.fit()
print(res.summary())

#delete CFC-11 which is not significant

```
                            OLS Regression Results
==========================================================================
Dep. Variable:                   Temp   R-squared:
Model:                            OLS   Adj. R-squared:
Method:                 Least Squares   F-statistic:
Date:                Wed, 25 Dec 2019   Prob (F-statistic)
Time:                        16:55:11   Log-Likelihood:
No. Observations:                 284   AIC:
Df Residuals:                     278   BIC:
Df Model:                           5
Covariance Type:            nonrobust
==========================================================================
                 coef    std err          t      P>|t|
--------------------------------------------------------------------------
const       -122.6255     20.383     -6.016      0.000
MEI            0.0626      0.007      9.481      0.000
CO2            0.0110      0.001     17.621      0.000
CFC-11        -0.0003      0.000     -0.923      0.357
TSI            0.0871      0.015      5.827      0.000
Aerosols      -1.5629      0.218     -7.161      0.000
==========================================================================
Omnibus:                       14.935   Durbin-Watson:
Prob(Omnibus):                  0.001   Jarque-Bera (JB):
Skew:                           0.449   Prob(JB):
Kurtosis:                       3.842   Cond. No.
==========================================================================
```

# 1.3 run the final model
climate_change_1_train_X_new_2 = climate_change_1_train.loc[:,['MEI','CO2','TSI','Aerosols']]

climate_change_1_train_X_new_2=sm.add_constant(climate_change_1_train_X_new_2)
model=sm.OLS(climate_change_1_train_Y,climate_change_1_train_X_new_2)
res=model.fit()
print(res.summary())

```
                            OLS Regression Results
==========================================================
Dep. Variable:                   Temp   R-squared:
Model:                            OLS   Adj. R-squared:
Method:                 Least Squares   F-statistic:
Date:                Wed, 25 Dec 2019   Prob (F-statistic)
Time:                        16:55:11   Log-Likelihood:
No. Observations:                 284   AIC:
Df Residuals:                     279   BIC:
Df Model:                           4
Covariance Type:            nonrobust
==========================================================
                 coef    std err          t      P>|t|
----------------------------------------------------------
const       -118.6016     19.906     -5.958      0.000
MEI            0.0620      0.007      9.438      0.000
CO2            0.0107      0.001     19.777      0.000
TSI            0.0842      0.015      5.764      0.000
Aerosols      -1.5844      0.217     -7.303      0.000
==========================================================
             5.03e+06
============================================================
```

Result:
We will use MEI, CO2, TSI and Aerosols.

(2) Train a better model than the model in Problem 2.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm

df = pd.read_csv('climate_change_1.csv')

def R_square(Y_pred, Y):
    ESS = np.sum((Y_pred - Y.mean())**2)
    TSS = np.sum((Y - Y.mean())**2)
    R2 = ESS / TSS
    print(R2)


def closed_form_2():
    lamb = 0.1
```

```
    df['const'] = 1
    traindata = df[df.Year<=2006]
    testdata = df[df.Year>2006]
    X = traindata[['MEI','CO2','TSI','Aerosols','const']]
    Y = traindata['Temp']

    Theta2 = np.dot(
        np.dot(np.linalg.inv((np.dot(X.T, X) + lamb * np.eye(X.shape[1]))), X.T), Y)
    print(Theta2)

    Y_pred = np.dot(X,Theta2)
    R_square(Y_pred,Y)

closed_form_2()
```

4. Problem Four-Gradient Descent

Iterative Expression:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm
from sklearn.preprocessing import normalize

df = pd.read_csv('climate_change_1.csv')

def Gradient_Descent(alpha, theta_0, x, y, tol):
    theta = theta_0
    cost = (1./(2*len(x))) * (np.sum((x @ theta_0 - y)**2))
    count = 0
    while not ((cost <= tol) | (count > 10000)):
        count += 1
        theta = theta - alpha * (1./len(x)) * (x.T @ (x @ theta - y))
        cost = (1./(2*len(x))) * (np.sum((x @ theta_0 - y)**2))
        print(theta)
    print('迭代共{}次'.format(count))
    print(theta)

def regularit(df):
    newDataFrame = pd.DataFrame(index=df.index)
```

```
    columns = df.columns.tolist()
    for c in columns:
        if c != 'const':
            d = df[c]
            MAX = d.max()
            MIN = d.min()
            newDataFrame[c] = ((d - MIN) / (MAX - MIN)).tolist()
        else:
            newDataFrame[c] = 1
    return newDataFrame

def closed_form_2():
    lamb = 0.1
    df['const'] = 1
    traindata = df[df.Year<=2006]
    testdata = df[df.Year>2006]
    X = traindata[['MEI','CO2','CH4','N2O','CFC-11','CFC-12','TSI','Aerosols','const']]
    Y = traindata['Temp']
    X = regularit(X)
    Y = (Y-Y.mean())/(Y.max()-Y.min())
    print(X)
    print(Y)

    Theta2 = [1,1,1,1,1,1,1,1,1]

    Gradient_Descent(0.001, Theta2, X, Y, 10)

closed_form_2()
```