Homework 2

Yuchuan Xu   1801212958   Fintech 2018

Problem 1 —Creating Your First Model

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

```
import statsmodels.api as sm
import matplotlib.pyplot as plt

CLEX = '/user/Desktop/climate_change_1.xlsx'
df = pd.DataFrame(pd.read_excel(CLEX))
train = df[df['Year']<=2006]
test = df[df['Year']>2006]

x = train[['MEI','CO2','CH4','N2O','CFC-11','CFC-12','TSI','Aerosols']]
x = sm.add_constant(x)
y = train['Temp']

def close_form_1(x,y):
    model = sm.OLS(y,x)
    results = model.fit()
    print(results.params)
    print(results.summary())

colse_form_1(x,y)
```

2. **Write down the mathematical formula for the linear model and evaluate the model R2 on the training set and the testing set.**

   Firstly, for training data set, the regression result:

Dep. Variable:	Temp	R-squared:	0.751
Model:	OLS	Adj. R-squared:	0.744
Method:	Least Squares	F-statistic:	103.6
Date:	Sat, 14 Dec 2019	Prob (F-statistic):	1.94e-78
Time:	16:06:23	Log-Likelihood:	280.10
No. Observations:	284	AIC:	-542.2
Df Residuals:	275	BIC:	-509.4
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-124.5943	19.887	-6.265	0.000	-163.744	-85.445
MEI	0.0642	0.006	9.923	0.000	0.051	0.077
CO2	0.0065	0.002	2.826	0.005	0.002	0.011
CH4	0.0001	0.001	0.240	0.810	-0.001	0.001
N2O	-0.0165	0.009	-1.930	0.055	-0.033	0.000
CFC-11	-0.0066	0.002	-4.078	0.000	-0.010	-0.003
CFC-12	0.0038	0.001	3.757	0.000	0.002	0.006
TSI	0.0931	0.015	6.313	0.000	0.064	0.122
Aerosols	-1.5376	0.213	-7.210	0.000	-1.957	-1.118

Based on the result, we can obtain the formula:

$Temp = 0.0642 * MEI + 0.0065 * CO2 + 0.0001 * CH4 - 0.0165 * N2O - 0.0066 * CFC11 + 0.0038 * CFC12 + 0.0931 * TSI - 1.5376 * Aerosols - 124.59$

$R2=0.751$, Adj $R2=0.744$

Secondly, for testing data set, the regression result:

Dep. Variable:	Temp	R-squared:	0.625
Model:	OLS	Adj. R-squared:	0.425
Method:	Least Squares	F-statistic:	3.124
Date:	Sat, 14 Dec 2019	Prob (F-statistic):	0.0273
Time:	16:18:14	Log-Likelihood:	31.674
No. Observations:	24	AIC:	-45.35
Df Residuals:	15	BIC:	-34.74
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1517.2210	1158.493	1.310	0.210	-952.049	3986.491
MEI	0.0317	0.046	0.695	0.498	-0.066	0.129
CO2	0.0044	0.018	0.245	0.809	-0.034	0.042
CH4	-0.0023	0.004	-0.532	0.602	-0.011	0.007
N2O	-0.1180	0.159	-0.742	0.470	-0.457	0.221
CFC-11	0.0382	0.149	0.255	0.802	-0.280	0.357
CFC-12	-0.0736	0.163	-0.452	0.658	-0.421	0.274
TSI	-1.0596	0.815	-1.301	0.213	-2.796	0.677
Aerosols	148.4231	65.593	2.263	0.039	8.614	288.232

Temp=0.0317* MEI + 0.0044 * CO2 - 0.0023 * CH4 - 0.1180 * N20 - 0.0382 * CFC11 -0.0736* CFC12 - 1.0596 * TSI - 148.4231 * Aerosols + 1517.22

$R2=0.625$, Adj $R2=0.425$

3. **Which variables are significant in the model?**
   First, for training data set: taking 0.05 as the significant level, the significant variables are MEI, CO2, CFC-11, CFC-12, TSI and Aerosols in this model.
   Second, for testing data set: different from the training data set, taking 0.05 as the significant level, the significant variable is only Aerosols in this model.

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 condition:
   1) no correlation between random error term and explanatory variable, COV (Xi, μi) == 0:
   2) there is no severe multicollinearity problem between explanatory variables: VIF < 10;
   3) Random error term obeys normal distribution of zero mean and same variance.

For training data:
The result of VIF test is:

Variable	VIF	1/VIF
CFC12	120.50	0.008299
N2O	55.89	0.017892
CFC11	39.18	0.025521
CO2	22.98	0.043511
CH4	18.67	0.053553
Aerosols	1.38	0.725477
MEI	1.22	0.821477
TSI	1.18	0.847789
Mean VIF	32.63	

VIF>10, severe multicollinearity problem between explanatory variables.

	Jarque-Bera test	Shapiro-Wilk test
Under H0	res ~ iid Normal	res is Gaussian
Implication	Prob>chi2=0.0171, we reject that the residuals come from a Gaussian	W=0.9896, p>z=0.0403<0.05, we reject that the residuals are

	distribution at a 5% level.	Gaussian at the 5% level. The statistic is too far below 1

So, as we can see from the test results, we cannot apply the linear regression to the training data set.

For testing data:

	Jarque-Bera test	Shapiro-Wilk test
Under H0	res ~ iid Normal	res is Gaussian
Implication	Prob>chi2=0.1474, we accept that the residuals come from a Gaussian distribution at a 5% level.	W=0.9612, p>z=0.4636>0.05, we accept that the residuals are Gaussian at the 5% level.

Therefore, the residuals are Gaussian distributed.

The result of VIF test is:

Variable	VIF	1/VIF
CFC12	186.12	0.005373
CFC11	144.49	0.006921
N2O	25.63	0.039021
CH4	8.07	0.123935
Aerosols	5.42	0.184424
CO2	4.58	0.218557
MEI	2.95	0.339302
TSI	2.30	0.435274
Mean VIF	47.44	

Since VIF> 10, the model will face serious multicollinearity problems, therefore, we cannot apply the linear regression to the testing data set.

Problem 2 — Regularization

1. **Please write down the loss function for linear model with L1 regularization, L2 regularization, respectively.**

For L1:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|$$

And J = J0 + α∑ |w|

For L2:

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$$

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

And J = J0 + α$\sum$w^2

2. **Write a function closed_form_2 that computes this closed form solution given the features X, labels Y and the regularization parameter λ.**

```
from sklearn.linear_model import LinearRegression, Lasso, Ridge

def closed_form_2(x,y,a):
    ridge = Ridge(alpha=a)
    ridge.fit(x, y)

    print('The coefficients are:',ridge.coef_)
    print('The intercept is:',ridge.intercept_)
    print('The R square is:',ridge.score(x, y))
```

3. **Compare the two solutions in problem 1 and problem 2 and explain the reason why linear model with L2 regularization is robust.**

L2 is robust because it limits the range of W, by which it can avoid part of the overfitting issues.

4. **You can change the regularization parameter λ to get different solutions for this problem. Suppose we set λ = 10, 1, 0.1, 0.01, 0.001, and please evaluate the model R2 on the training set and the testing set. Finally, please decide the best regularization parameter λ.**

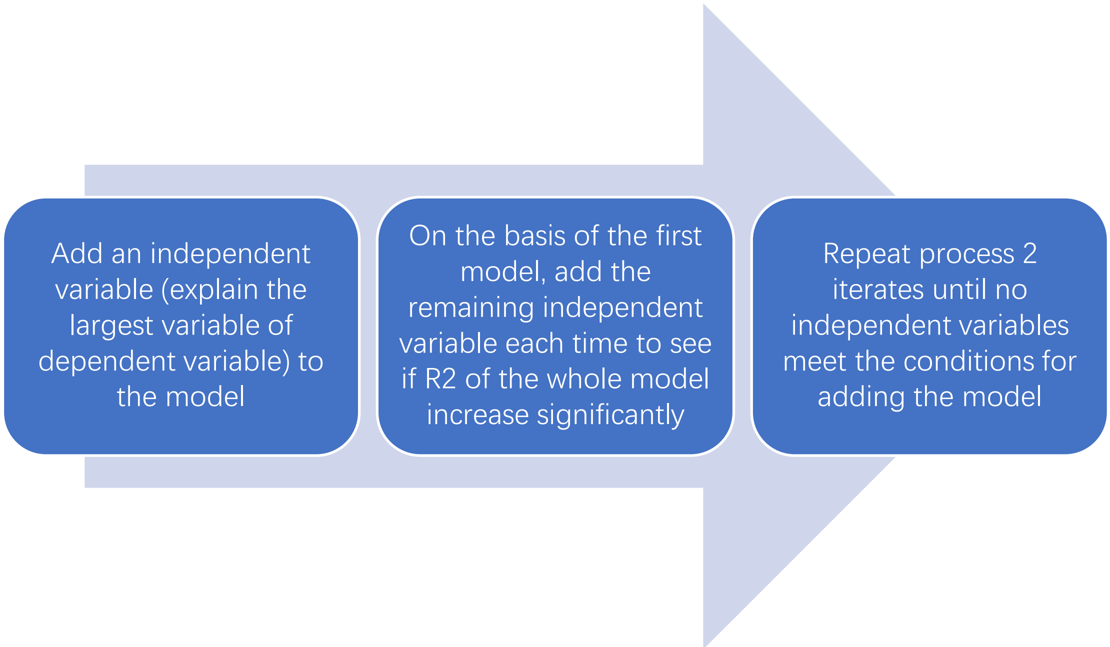
```
In [ ]: from sklearn.linear_model import RidgeCV

        ridge1 = RidgeCV (alphas = [10, 1, 0.1, 0.01, 0.001])
        ridge1.fit(x,y)

        print('For traing data:', ridge1.alpha_)

        ridge1.fit(x2,y2)
        print('For testing data:', ridge1.alpha_)
```

For training data: 0.01
For testing data: 1.0

Problem 3 — 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.**

The workflow of this stepwise regression is:



Applying stepwise regression method, setting 0.1 as the significant level, the features I select are CO2, MEI, Aerosols, TSI and N20. The new model R2 s 0.7221. RMSE is 0.09519.

```
. stepwise, pe(0.1):regress Temp MEI CO2 CH4 N2O CFC11 CFC12 TSI Aerosols
                      begin with empty model
p = 0.0000 <  0.1000  adding  CO2
p = 0.0000 <  0.1000  adding  MEI
p = 0.0000 <  0.1000  adding  Aerosols
p = 0.0000 <  0.1000  adding  TSI
p = 0.0639 <  0.1000  adding  N2O
```

Source	SS	df	MS			
				Number of obs	=	308
				F(5, 302)	=	156.95
Model	7.11023826	5	1.42204765	Prob > F	=	0.0000
Residual	2.73621729	302	.009060322	R-squared	=	0.7221
				Adj R-squared	=	0.7175
Total	9.84645554	307	.032073145	Root MSE	=	.09519

Temp	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
CO2	.0057522	.0022553	2.55	0.011	.001314	.0101903
MEI	.0690259	.0062477	11.05	0.000	.0567314	.0813204
Aerosols	-1.745317	.2147222	-8.13	0.000	-2.167859	-1.322776
TSI	.0961337	.0138219	6.96	0.000	.0689344	.1233331
N2O	.0101027	.005433	1.86	0.064	-.0005887	.0207941
_cons	-136.304	18.80261	-7.25	0.000	-173.3047	-99.3033

Problem 4 — Gradient Descent

Gradient descent algorithm is an iterative process that takes us to the minimum of a function. Please write down the iterative expression for updating the solution of linear model and implement it using Python or Matlab in gradientDescent function.

In [33]:

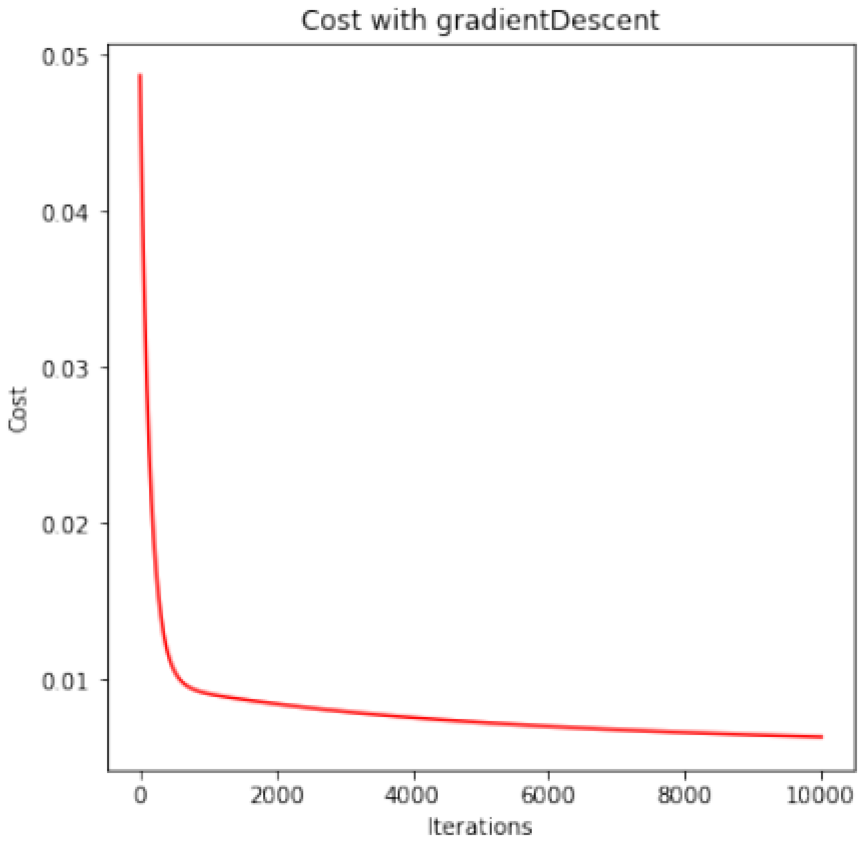
```
from numpy.linalg import inv
from numpy import dot
theta_n = dot(dot(inv(dot(X.T, X)), X.T), y)
print (theta_n)
def computeCost(X, y, theta):
inner = np.power(((X * theta.T) - y), 2)
return np.sum(inner) / (2 * len(X))
print(X.shape, theta_n.shape, y.shape)
lr_cost = computeCost(X, y, theta_n.T)
print(lr_cost)
```

```
[[-0.07643977]
 [ 0.30746786]
 [ 0.25167652]
 [ 0.01174117]
 [-0.31326271]
 [-0.58346414]
 [ 0.82748116]
 [ 0.18118894]
 [-0.23379557]]
(308, 9) (9, 1) (308, 1)
0.004092130064396185
```

In [34]:
```
def gradientDescent(X, y, theta, alpha, iters):
temp = np.matrix(np.zeros(theta.shape))
parameters = int(theta.ravel().shape[1])
cost = np.zeros(iters)
for i in range(iters):
error = (X * theta.T) - y
for j in range(parameters):
term = np.multiply(error, X[:,j])
temp[0,j] = theta[0,j] - ((alpha / len(X)) * np.sum(term))
theta = temp
cost[i] = computeCost(X, y, theta)
return theta, cost
```

In [36]:
```
alpha = 0.001
iters = 10000
theta = np.matrix(np.array([0,0,0,0,0,0,0,0,0]))
g, cost = gradientDescent(X, y, theta, alpha, iters)
print('thetas are',g)
print('final cost is',cost)
fig, bx = plt.subplots(figsize=(6,6))
bx.plot(np.arange(iters), cost, 'r')
bx.set_xlabel('Iterations')
bx.set_ylabel('Cost')
bx.set_title('Cost with gradientDescent')
plt.show()
```

```
thetas are [[-0.00898942  0.05557721  0.11781496  0.0806716   0.12249706  0.00789335
   0.09353225  0.03370607 -0.0737614 ]]
final cost is [0.04867689 0.04840425 0.0481335  ... 0.00630191 0.00630179 0.00630166]
```


Cost with gradientDescent
0.05
0.04
0.03
0.02
0.01
Cost
0
2000
4000
6000
8000
10000
Iterations