Ridge regression and LASSO for one variable

In this problem we'll try to understand the effect of regularization on model parameters by considering what happens when n=p=1, i.e. we have just one data point and just one predictor. In this case the equations are simple and by plotting the results we can build intuition.

In [1]:

```python
# Part A
import numpy as np
import matplotlib.pyplot as plt

def C_rigid(y, beta, lamda):
    return (y-beta)**2 + lamda*beta**2;

betas = np.arange(0.0,3.0, 0.1)
plt.plot(betas, [ C_rigid(2,beta,0) for beta in betas ],color="red",linewidth=
plt.title("the cost function of Ridge Regression")
plt.xlabel("beta")
plt.ylabel("C_rigid")
plt.show()
```

executed in 2.30s, finished 01:09:20 2018-11-14

```
<matplotlib.figure.Figure at 0x10b1dfcd0>
```

What is beta, the value of beta that minimizes Cridge= (2-beta)^2 ?

Since the differentail of Cridge is d(Cridge)/d(beta) = -2(2-beta)

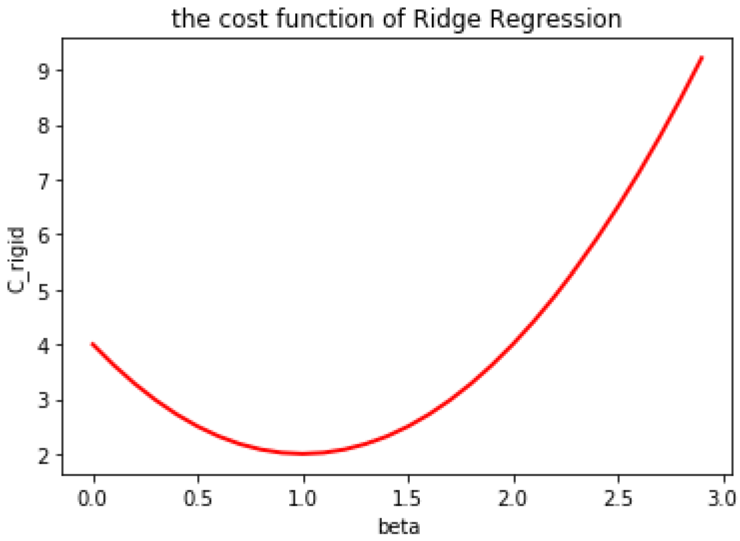
so this is minimized when beta is 2

In [2]:

```
#part B
# Now plot Cridge as a function of β for λ = 1

betas = np.arange(0.0,3.0, 0.1)
plt.plot(betas, [ C_rigid(2,beta,1) for beta in betas ],color="red",linewidth=
plt.title("the cost function of Ridge Regression")
plt.xlabel("beta")
plt.ylabel("C_rigid")
plt.show()
```

executed in 387ms, finished 01:09:21 2018-11-14



Show that the new beta is given by the formula in (6.14), . Describe, in words, how changes as β̂ /(1) /2 β̂ = y + λ = y β̂ λ increases -- this illustrates why ridge regression is a form of shrinkage .

The new beta that min the cost function is beta=1

since now Cridge = (2- beta)^2 + gama * beta^2

d(Cridge)/d(beta) = -2(2-beta) + 2* beta^2

Now if gama ==1 then beta = y/2

As gama increase the beta will decrease and shrink to 0

In [3]:

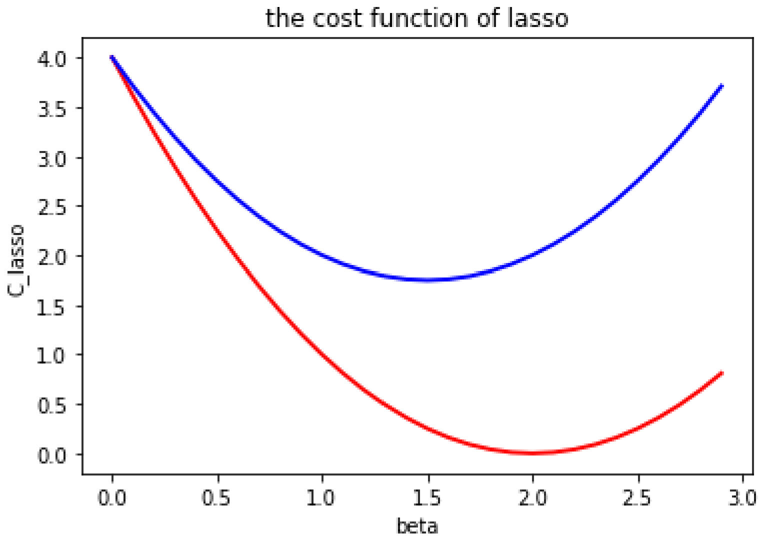
```
# part C
# Consider the cost function for LASSO, with p=1:
# Classo = (y − β)2 + λ |β|
# Make plots showing Classo as a function of β for λ = 0 and λ = 1 .


def C_lasso(y, beta, lamda):
    return (y-beta)**2 + lamda*abs(beta);

betas = np.arange(0.0,3.0, 0.1)
plt.plot(betas, [ C_lasso(2,beta,0) for beta in betas ],color="red",linewidth=
plt.plot(betas, [ C_lasso(2,beta,1) for beta in betas ],color="blue",linewidth
plt.title("the cost function of lasso")
plt.xlabel("beta")
plt.ylabel("C_lasso")
plt.show()

```

executed in 328ms, finished 01:09:21 2018-11-14



The min value of beta for lamda=0 is beta=2 when Lamda=0 it shows the cost func without regularization

The min value of beta for lamda=1 is beta=1.5

Notice that the two cost functions converge at beta=0, which is a result of weight shrinkage.

Forward stepwise model selection

In [4]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

#read data
df = pd.read_csv('./anesthesia.csv')
data = df.values
print data.shape
```

executed in 3.27s, finished 01:09:24 2018-11-14

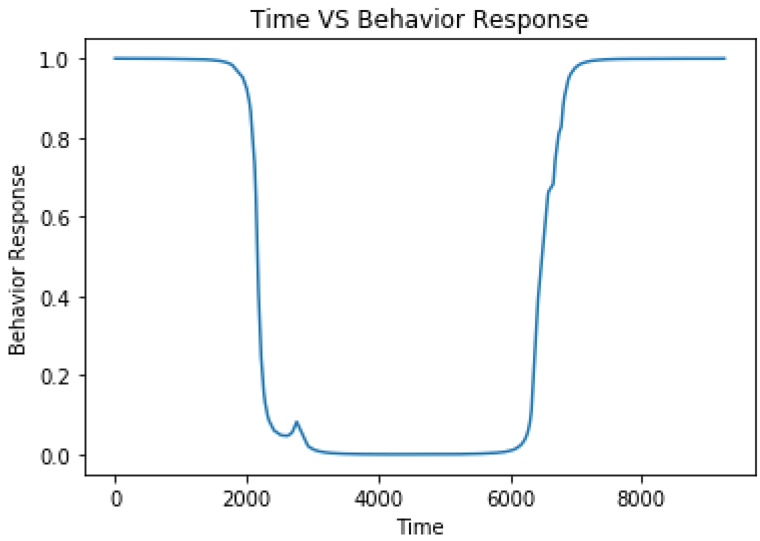
```
(926, 105)
```

In [20]:

```
# Part A
# Make a plot of Time Vs. BehaviorResponse for the anesthesia data.
time = data[:,0]
BehaviorResponse = data[:,104]
plt.plot(time, BehaviorResponse)
plt.title("Time VS Behavior Response")
plt.xlabel("Time")
plt.ylabel("Behavior Response")
plt.show()

```

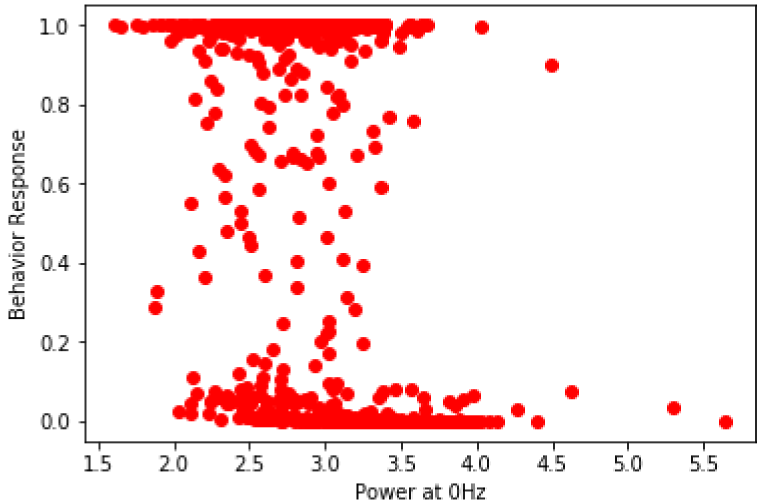
executed in 250ms, finished 01:23:39 2018-11-14



In [35]:

```
# Part b.Make a scatter plot of BehaviorResponse vs. EEG power at 0Hz
plt.plot(df['F0Hz_1'],df['BehaviorResponse'], 'ro');
plt.ylabel('Behavior Response');
plt.xlabel('Power at 0Hz');
```

executed in 287ms, finished 01:34:19 2018-11-14



Describe, in words, the relationship between these variables.

As EEG power increases at 0H, the prob of BehaviorResponse=1 decreases

In [41]:

```
## Part C
#What is the correlation coefficient between Behavior Response and EEG power a
import numpy as np

corr_coef = np.corrcoef(df['F0Hz_1'],df['BehaviorResponse']);
print 'The correlation coefficient is ', corr_coef[0][1]
```

executed in 9ms, finished 01:37:17 2018-11-14

```
The correlation coefficient is  -0.4632119330934692
```

In [52]:

```
# Part D
# Fit a simple linear regression model of the form, BehaviorResponse ~ 1 + F0H
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from patsy import dmatrices

### do linear regression
y, X = dmatrices('BehaviorResponse ~ 1 + F0Hz_1', data=df, return_type='datafr
X = sm.add_constant(X)
mod = sm.OLS(y, X)
# fit model
res = mod.fit()
# look at results
print(res.summary())
```

executed in 54ms, finished 01:55:28 2018-11-14

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:       BehaviorResponse   R-squared:
   0.215
Model:                            OLS   Adj. R-squared:
   0.214
Method:                 Least Squares   F-statistic:
   252.4
Date:                Wed, 14 Nov 2018   Prob (F-statistic):
1.96e-50
Time:                        01:55:28   Log-Likelihood:
-510.49
No. Observations:                 926   AIC:
   1025.
Df Residuals:                     924   BIC:
   1035.
Df Model:                           1

Covariance Type:            nonrobust

======================================================================
========
                 coef    std err          t      P>|t|      [0.025
  0.975]
----------------------------------------------------------------------
--------
Intercept      1.8503      0.084     22.000      0.000       1.685
   2.015
F0Hz_1        -0.4433      0.028    -15.888      0.000      -0.498
  -0.389
======================================================================
========
Omnibus:                      995.624   Durbin-Watson:
   0.198
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
  63.317
Skew:                          -0.152   Prob(JB):
1.78e-14
Kurtosis:                       1.755   Cond. No.
```

```
    20.3
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Is the slope parameter statistically significant?

Since the p-value is smaller than 0.05, there is a statistically significant relationship between BehaviorResponse and power at 0Hz

In [122]:

```
# part E
# Fit a multiple linear regression that uses all of the EEG power features

features = df.head(0)
features_str = '1 '
for feature in features:
    if feature != "Time" and feature != "BehaviorResponse":
        features_str += ' + ' + feature;
```

executed in 7ms, finished 03:23:01 2018-11-14

In [127]:

```
X = df[['F0Hz_1', 'F125Hz_103']]
y = df['BehaviorResponse']

### do linear regression
y, X = dmatrices('BehaviorResponse ~' + features_str, data=df, return_type='datafra
X = sm.add_constant(X)
mod = sm.OLS(y, X)
# fit model
res = mod.fit()
# look at results
print(res.summary())
```

executed in 697ms, finished 03:24:10 2018-11-14

```
Intercept      0.6519      0.131      4.994      0.000      0.396
   0.908
F0Hz_1        -0.0122      0.017     -0.698      0.485     -0.046
   0.022
F1Hz_2        -0.0394      0.025     -1.560      0.119     -0.089
   0.010
F3Hz_3         0.0006      0.028      0.020      0.984     -0.054
   0.055
F4Hz_4        -0.0060      0.027     -0.226      0.822     -0.058
   0.046
F5Hz_5         0.0008      0.025      0.032      0.974     -0.049
   0.050
F6Hz_6         0.1510      0.022      6.858      0.000      0.108
   0.194
F8Hz_7        -0.0003      0.024     -0.012      0.991     -0.047
   0.047
F9Hz_8         0.0066      0.027      0.247      0.805     -0.046
   0.059
F10Hz_9       -0.1492      0.024     -6.324      0.000     -0.195
  -0.103
```

What is the p-value of the slope for F0Hz_1?

0.485

Is it statistically significant?

The slope coefficient for power at F0Hz_1 is Not significant Now

In [206]:

```
# part F
# Fit a 103 models and find the single features with the lowest MSE
# for each fit use the MSE
# make a plot showing MSE VS feature number

mse_list = [0 for i in range(103)]
feature_list = []
for feature in features:
    if feature != "Time" and feature != "BehaviorResponse":
        feature_list.append(feature)

# 103 iterations
index = 0
for feature in feature_list:
    y, X = dmatrices('BehaviorResponse ~' + feature, data=df, return_type='dat
    X = sm.add_constant(X)
    mod = sm.OLS(y, X)
    # fit model
    res = mod.fit()
    #predict the data
    y_hat = res.predict(X);
    mse_list[index] = np.mean( (df['BehaviorResponse'] - y_hat)**2 )
    index += 1

```

executed in 1.19s, finished 04:19:36 2018-11-14

In [207]:

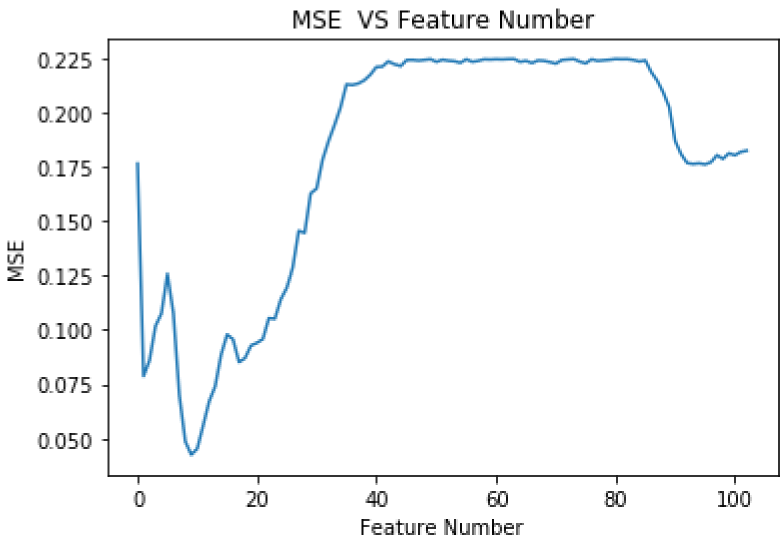
```
print min(mse_list)
print "So F11Hz_10 is winning feature"

x=range(103)
y= []
for index in range(103):
    y.append( mse_list[index] )

plt.plot(x,y)
plt.title("MSE  VS Feature Number")
plt.xlabel("Feature Number")
plt.ylabel("MSE")
plt.show()
```

executed in 281ms, finished 04:19:37 2018-11-14

```
0.0423969081374
So F11Hz_10 is winning feature
```



In [213]:

```
# Part G
# Now write a loop to fit 102 models of the form, BehaviorResponse ~ 1 + X1 +
# the best feature obtained from part (f) and X2 is one of the other features.
# Which combination of two features gives the best prediction?

best_feature = "F11Hz_10"

#build a new list
new_feature_list = []
for feature in features:
    if feature != "Time" and feature != "BehaviorResponse" and feature!="F11Hz
        new_feature_list.append(feature)

print len(new_feature_list)

# 102 iterations
index = 0
mse_list = [0 for i in range(103)]
for feature in new_feature_list:
    y, X = dmatrices('BehaviorResponse ~ 1 + ' + best_feature \
                     + '+' + feature, data=df, return_type='dataframe')
    X = sm.add_constant(X)
    mod = sm.OLS(y, X)
    # fit model
    res = mod.fit()
    #predict the data
    y_hat = res.predict(X);
    mse_list[index] = np.mean( (df['BehaviorResponse'] - y_hat)**2 )
    index += 1

print 'the winning combination is ', best_feature, \
' and ' ,new_feature_list[9]
```

executed in 1.25s, finished 04:24:05 2018-11-14

```
102
the winning combination is  F11Hz_10  and  F12Hz_11
```

In [241]:

```
# Part H
from sklearn.model_selection import KFold

best_comb = "F11Hz_10 + F12Hz_11";
Num_Splits = 10
Num_observation = len(df['BehaviorResponse'])
kf = KFold(n_splits=10)
X_Splits = kf.get_n_splits(Num_observation)
MSE_matrix = ones(10,2)

#iterate 10 times
for train_index, test_index in kf.split(X):
    train = (kf == index)
    test = test_ind
    y, X = dmatrices('BehaviorResponse ~ 1 + ' + best_comb, return_type='dataf
    X = sm.add_constant(X)
    mod = sm.OLS(y, X)
    res = mod.fit()
    y_hat = res.predict(X);
    mse_list[index] = np.mean( (df['BehaviorResponse'] - y_hat)**2 )

    model = fitlm(data(train_ind,:), ['BehaviorResponse ~ 1 + ' winning_comb])
    y_hat = predict(model, data);

    MSE_matrix[index][0] = np.mean( (df['BehaviorResponse'] - y_hat)**2 )
    MSE_matrix[index][1] = np.mean( (df['BehaviorResponse'] - y_hat)**2 )

print "So training and testing MSE for this model are printed below"
print "train    Test"
for item in MSE_matrix:
    print item
```

executed in 30ms, finished 04:57:36 2018-11-14

```
So training and testing MSE for this model are printed below
train    Test
[0.037099 0.024939]
[0.035453 0.039801]
[0.036064 0.034278]
[0.036726 0.028319]
[0.036026 0.034621]
[0.034259 0.050515]
[0.036244 0.032711]
[0.036189 0.033118]
[0.035931 0.035774]
[0.034742 0.046127]
```

In [ ]:

```

```