In [73]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from tqdm import tnrange,tqdm_notebook
import itertools
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from sklearn.linear_model import Ridge
from sklearn.datasets import make_regression
```

Conceptual exercises

1. Generate dataset

In [76]:

```
X,Y,beta = make_regression(1000,20,15,coef = True)
print(X.shape,Y.shape,beta.shape)
```

```
(1000, 20) (1000,) (20,)
```

2. Split dataset

In [77]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size = 100)
```

3. Perform best subset selection and plot training MSE

In [90]:

```
def my_linear_model(X,Y):
    model = linear_model.LinearRegression(fit_intercept = True)
    model.fit(X,Y)
    mse = mean_squared_error(Y,model.predict(X))
    return mse
```

In [91]:

```
mse_list,feature_list,num_feature = [],[],[]

for k in tnrange(1,X.shape[1]+1):
    for combination in itertools.combinations(list(range(20)),k):
        #print(combination)
        temp_result = my_linear_model(X_train[:,list(combination)],Y_train)
        mse_list.append(temp_result)
        feature_list.append(combination)
        num_feature.append(len(combination))
```

In [92]:

```
df = pd.DataFrame({'num_feature':num_feature,'mse':mse_list,'feature_list':featu
re_list})
```

In [93]:

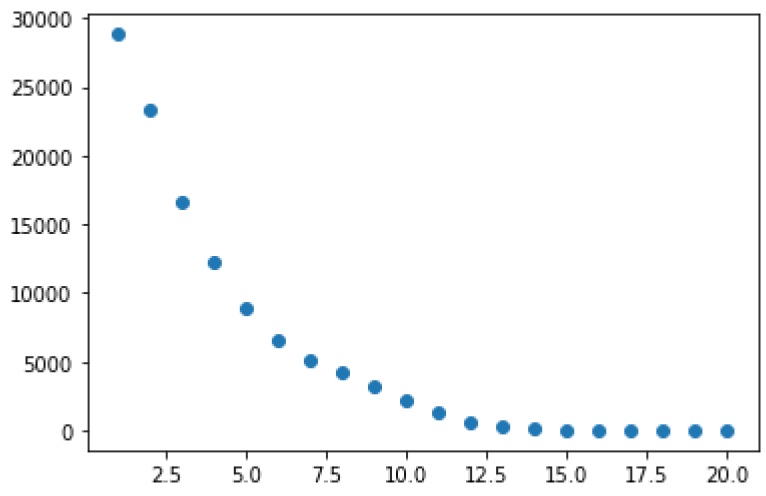
```
df_min = df[df.groupby('num_feature')['mse'].transform(min) == df['mse']]
```

In [94]:

```
plt.scatter(df_min.num_feature,df_min.mse)
```

Out[94]:

```
<matplotlib.collections.PathCollection at 0x1a2021a2e8>
```



4. Plot test set MSE

In [95]:

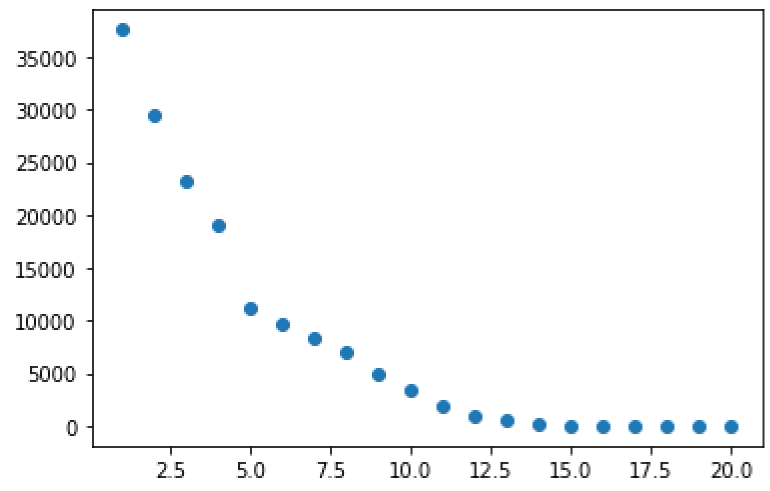
```
test_mse_list = []
for item in list(df_min.feature_list):
    mse = my_linear_model(X_test[:,list(item)],Y_test)
    test_mse_list.append(mse)
```

In [96]:

```
plt.scatter(df_min.num_feature,test_mse_list)
```

Out[96]:

```
<matplotlib.collections.PathCollection at 0x1a296240f0>
```



5. Which model size does the test set MSE take on its minimum value?

```
The model performs best when the size is 16. This is very close to the bet
a we used to generate the dataset. In our beta, we have five zeros, and in
our best model, we have 4 zeros coeficients and 1 coeficient very close to
zero.
```

6. How does the model at test test compare to true model?

In [97]:

```
def cal_coef(feature):
    model = linear_model.LinearRegression(fit_intercept = True).fit(X_train[:,li
st(feature)],Y_train)
    model_coef = []
    k = 0
    for i in range(20):
        if i not in feature:
            model_coef.append(0)
        else:
            model_coef.append(model.coef_[k])
            k+=1
    return model_coef
```

In [99]:

```
best_index = test_mse_list.index(min(test_mse_list))
best_feature = list(df_min.feature_list.iloc[best_index])
model_coef = cal_coef(best_feature)
print(model_coef)
print(beta)
stats.ttest_rel(beta,model_coef)
```

```
[0, 0, 0, 92.95593664253303, 61.56390975664588, 33.09612935007106, 1
7.816861976716734, 13.424527984131316, 18.97988511458644, 36.0137432
8775691, 81.65715262228957, 48.85778715431231, 83.5629258724932, 40.
45626643914372, 45.655819928753694, 0, 40.207604750886155, 38.336886
403851324, -5.577757517127241e-14, 93.1032789285086]
[ 0.          0.          0.         92.95593664 61.56390976 33.0961
2935
 17.81686198 13.42452798 18.97988511 36.01374329 81.65715262 48.8577
8715
 83.56292587 40.45626644 45.65581993  0.         40.20760475 38.3368
864
  0.         93.10327893]
```

Out[99]:

```
Ttest_relResult(statistic=-0.028590418214888964, pvalue=0.9774894284
386855)
```

The coefficient size of our model is much similar to the true model. From the T test, we could see that the coefficients of our best model don't differ significantly from the true model.

7. Plot coefficient estimate compare to true

In [105]:

```
coef_diff_list = []
for i in range(20):
    feature = list(df_min.feature_list.iloc[i])
    coef = cal_coef(feature)

    diff = np.sqrt(np.sum(np.square(beta-coef)))
    coef_diff_list.append(diff)
print(coef_diff_list)
```

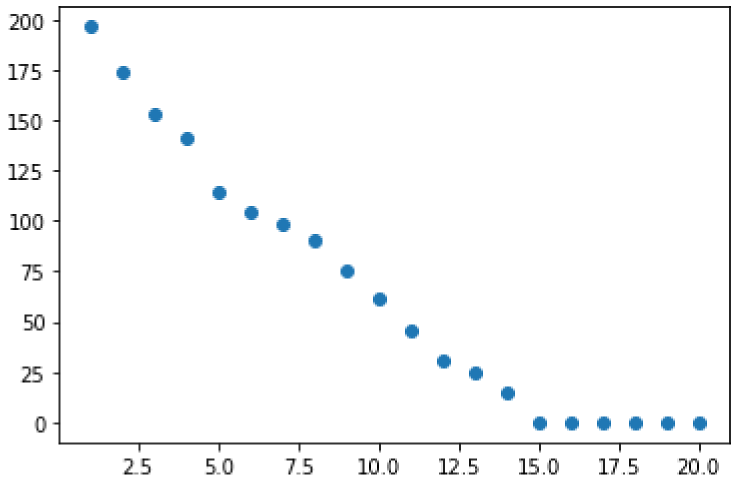
```
[196.54220939875586, 174.31587435845503, 152.8076659250246, 140.7914
3373755227, 114.74379597874575, 104.73822667010272, 98.4152625803238
1, 90.34437241410882, 75.36277856156407, 62.03320783288605, 46.12155
960728073, 30.546678819384116, 24.605018776900497, 14.43693739289573
2, 1.910306401484432e-13, 2.180144897550887e-13, 2.6665432922069574e
-13, 2.136879298797491e-13, 2.2650218791090567e-13, 4.48022008740490
9e-13]
```

In [109]:

```
plt.scatter(df_min.num_feature,coef_diff_list)
#plt.scatter(df_min.num_feature,test_mse_list,color = 'red',label = 'MSE')
```

Out[109]:

```
<matplotlib.collections.PathCollection at 0x1a285d6358>
```



Similar to the relationship between test set MSE and model size we have drawn on problem5, the calculated difference between true coefficient and coefficient estimate decreases as the number of coefficients increase. The figure reaches the lower point when the number of coefficients is 15, which is the number of true coefficients. The descending at the range of 1 to 15 coefficient size is sharper, to be compared with, the MSE plot decreases gradually from 1 to 13, and remains steadily with a slight decrease from 13 to 20 coefficients.

Application exercises

In [110]:

```
df_train = pd.read_csv('/Users/lijiaxuan/Downloads/problem-set-3-master/data/gss
_train.csv')
df_test = pd.read_csv('/Users/lijiaxuan/Downloads/problem-set-3-master/data/gss_
test.csv')
```

In [111]:

```
df_train.columns
```

Out[111]:

```
Index(['age', 'attend', 'authoritarianism', 'black', 'born', 'child
s',
       'colath', 'colrac', 'colcom', 'colmil', 'colhomo', 'colmslm',
       'con_govt', 'egalit_scale', 'evangelical', 'grass', 'happy',
       'hispanic_2', 'homosex', 'income06', 'mode', 'owngun', 'polvi
ews',
       'pornlaw2', 'pray', 'pres08', 'reborn_r', 'science_quiz', 'se
x', 'sibs',
       'social_connect', 'south', 'teensex', 'tolerance', 'tvhours',
       'vetyears', 'wordsum', 'degree_HS', 'degree_Junior.Coll',
       'degree_Bachelor.deg', 'degree_Graduate.deg', 'marital_Widowe
d',
       'marital_Divorced', 'marital_Separated', 'marital_Never.marri
ed',
       'news_FEW.TIMES.A.WEEK', 'news_ONCE.A.WEEK', 'news_LESS.THAN.
ONCE.WK',
       'news_NEVER', 'partyid_3_Ind', 'partyid_3_Rep', 'relig_CATHOL
IC',
       'relig_JEWISH', 'relig_NONE', 'relig_OTHER', 'relig_BUDDHIS
M',
       'relig_HINDUISM', 'relig_OTHER.EASTERN', 'relig_MOSLEM.ISLA
M',
       'relig_ORTHODOX.CHRISTIAN', 'relig_CHRISTIAN', 'relig_NATIVE.
AMERICAN',
       'relig_INTER.NONDENOMINATIONAL', 'social_cons3_Mod',
       'social_cons3_Conserv', 'spend3_Mod', 'spend3_Liberal', 'zodi
ac_TAURUS',
       'zodiac_GEMINI', 'zodiac_CANCER', 'zodiac_LEO', 'zodiac_VIRG
O',
       'zodiac_LIBRA', 'zodiac_SCORPIO', 'zodiac_SAGITTARIUS',
       'zodiac_CAPRICORN', 'zodiac_AQUARIUS', 'zodiac_PISCES'],
      dtype='object')
```

In [112]:

```
X_train, Y_train = df_train.loc[:, df_train.columns != 'egalit_scale'],df_train[
'egalit_scale']
X_test, Y_test = df_test.loc[:, df_test.columns != 'egalit_scale'],df_test['egal
it_scale']
print(X_train.shape,Y_train.shape,X_test.shape,Y_test.shape)
```

```
(1481, 77) (1481,) (493, 77) (493,)
```

1. Fit least squares linear model

In [167]:

```
#np.linalg.lstsq(X_train, Y_train)
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, Y_train)
mse = mean_squared_error(Y_test,model.predict(X_test))
print(mse)
```

```
63.213629623014995
```

2. Fit ridge regression model

In [178]:

```
from sklearn.linear_model import RidgeCV
rdg_clf = RidgeCV(alphas=np.linspace(1e-3,1,10)).fit(X_train, Y_train)
rdg_mse = mean_squared_error(Y_test,rdg_clf.predict(X_test))
print(rdg_mse)
```

```
63.05236636932556
```

3. Fit lasso regression model

In [183]:

```
from sklearn.linear_model import LassoCV
from sklearn.datasets import make_regression

lasso = LassoCV(cv=10, random_state=0).fit(X_train, Y_train)
lasso_mse = mean_squared_error(Y_test,lasso.predict(X_test))
non_zero_coef = [item for item in lasso.coef_ if item != 0]
print(len(non_zero_coef))
print(lasso_mse)
```

```
24
62.7780157899344
```

In [122]:

```
from sklearn.linear_model import ElasticNetCV
elas = ElasticNetCV(cv= 10,alphas=np.linspace(0.1,1,10),l1_ratio = np.linspace(
0.1,1,10)).fit(X_train,Y_train)
elas_mse = mean_squared_error(Y_test,elas.predict(X_test))
non_zero_coef = [item for item in elas.coef_ if item != 0]
print(elas_mse)
print(len(non_zero_coef))
print(elas.l1_ratio_)
print(elas.alpha_)
```

```
62.77841555477389
24
1.0
0.1
```

5. Comment on the results obtained

```
Overall, these models do not predict egalitarianism scales very well, with
MSE at around 63. There are not much difference between four models, with
 lasso regression outperformed slightly than others. In the elastic net mo
del, the l1_ratio equals to 1, which suggests that it does not differentia
te from lasso model. In most of the models, the coefficients are close to
 0, suggesting that most variables do not strong predicting power for the
 response variable.
```