In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from ipynb.fs.full.CoordinateDescent import coordinate_descent
from ipynb.fs.full.CoordinateDescent import maxLambda
```

In [2]:
```
df_train = pd.read_table("../data/crime-train.txt")
df_test = pd.read_table("../data/crime-test.txt")
```

In [3]:
```
df_train.head()
```

Out[3]:

	ViolentCrimesPerPop	population	householdsize	agePct12t21	agePct12t29	agePct16t24	a
0	0.67	-0.45	-1.85	-1.06	0.67	0.08	
1	0.43	-0.45	-0.27	-0.22	-0.17	-0.34	
2	0.12	-0.14	1.87	0.55	0.04	0.02	
3	0.03	-0.38	0.53	-0.28	-0.79	-0.64	
4	0.14	-0.30	-1.12	-0.74	-0.10	-0.40	

5 rows × 96 columns

In [4]:
```
df_test.head()
```

Out[4]:

	ViolentCrimesPerPop	population	householdsize	agePct12t21	agePct12t29	agePct16t24	a
0	0.08	-0.14	0.35	-0.41	-0.10	-0.46	
1	0.22	0.02	-0.45	-0.22	-0.24	-0.40	
2	0.06	-0.45	0.28	-0.16	0.18	-0.46	
3	0.16	0.02	-0.27	-0.67	-0.51	-0.58	
4	0.15	-0.22	1.20	3.71	3.53	3.99	

5 rows × 96 columns

In [5]:
```
################## setting up parameters ###################

##################### training ##############################
n,d = df_train.shape
y = df_train["ViolentCrimesPerPop"]
X = df_train.drop("ViolentCrimesPerPop",axis=1).values
Lam = maxLambda(X,y)
##################### testing ##############################
n,d = df_test.shape
y_test = df_test["ViolentCrimesPerPop"]
X_test = df_test.drop("ViolentCrimesPerPop",axis=1).values
#Lam_test = maxLambda(X_test,y_test)
```

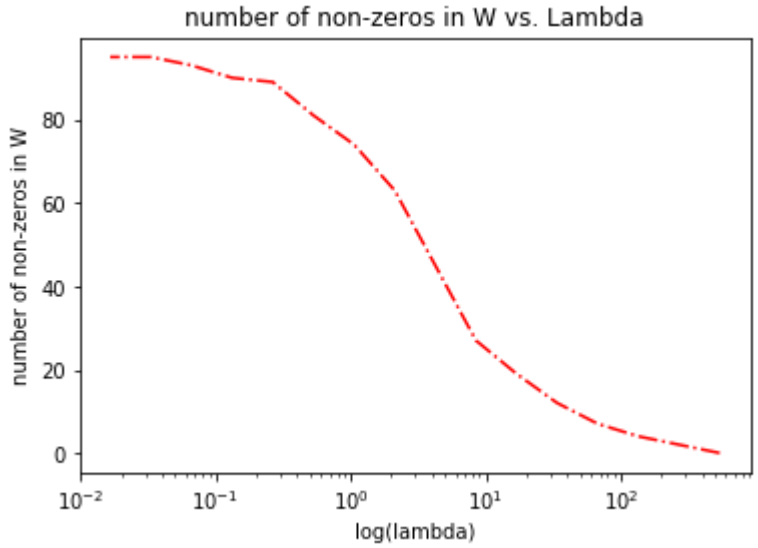
In [6]:
```
#####################################
w_yield=None
W = list()
LAM = list()
B = list()
while(Lam >= 0.01):
    b,w = coordinate_descent(y,X,Lam,tolerence=1e-5,w_init=w_yield)
    B.append(np.copy(b))
    W.append(w)
    w_yield = np.copy(w)
    LAM.append(Lam)
    Lam = Lam/2
```

```
++++++++++++++++++++++++++++Calculating for Lambda = 541.482602633228
9++++++++++++++++++++++++++
..................... it takes 1 iterations to converg
e...................
........error is 0.0...........
++++++++++++++++++++++++++++Calculating for Lambda = 270.741301316614
45++++++++++++++++++++++++++
..................... it takes 26 iterations to converg
e...................
........error is 7.842336644088566e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 135.370650658307
23++++++++++++++++++++++++++
..................... it takes 18 iterations to converg
e...................
........error is 8.576649476468856e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 67.6853253291536
1++++++++++++++++++++++++++
..................... it takes 53 iterations to converg
e...................
........error is 9.881997428924389e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 33.8426626645768
1++++++++++++++++++++++++++
..................... it takes 29 iterations to converg
e...................
........error is 8.907943691488507e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 16.9213313322884
03++++++++++++++++++++++++++
..................... it takes 31 iterations to converg
e...................
........error is 9.54418818094327e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 8.46066566614420
2++++++++++++++++++++++++++
..................... it takes 41 iterations to converg
e...................
........error is 9.38435787124943e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 4.23033283307210
1++++++++++++++++++++++++++
..................... it takes 103 iterations to converg
e...................
........error is 9.70686131387781e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 2.11516641653605
04++++++++++++++++++++++++++
..................... it takes 218 iterations to converg
e...................
........error is 9.938004736557635e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 1.05758320826802
52++++++++++++++++++++++++++
..................... it takes 517 iterations to converg
e...................
........error is 9.959325723844087e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 0.52879160413401
26++++++++++++++++++++++++++
..................... it takes 363 iterations to converg
e...................
........error is 9.99368273296719e-06...........
++++++++++++++++++++++++++++Calculating for Lambda = 0.26439580206700
63++++++++++++++++++++++++++
```

```
..................... it takes 356 iterations to converg
e...................
.......error is 9.980574927576612e-06............
++++++++++++++++++++++++++++Calculating for Lambda = 0.13219790103350
315++++++++++++++++++++++++++++
..................... it takes 1127 iterations to converg
e...................
.......error is 9.987981648924837e-06............
++++++++++++++++++++++++++++Calculating for Lambda = 0.06609895051675
158++++++++++++++++++++++++++++
..................... it takes 829 iterations to converg
e...................
.......error is 9.991786721341711e-06............
++++++++++++++++++++++++++++Calculating for Lambda = 0.03304947525837
579++++++++++++++++++++++++++++
..................... it takes 572 iterations to converg
e...................
.......error is 9.991548042612153e-06............
++++++++++++++++++++++++++++Calculating for Lambda = 0.01652473762918
7894++++++++++++++++++++++++++++
..................... it takes 347 iterations to converg
e...................
.......error is 9.992766062130731e-06............
```

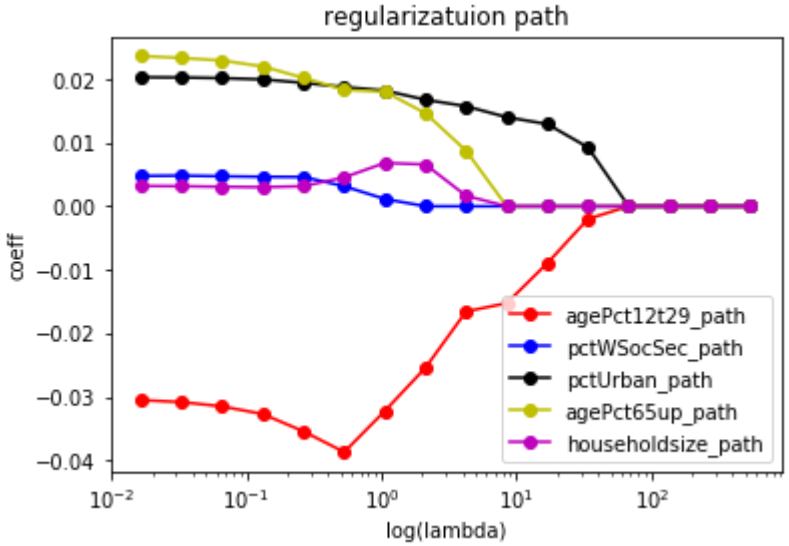
In [7]:

```
############################## A5 (a) ##################################
###################################
none_zeros = [np.count_nonzero(x) for x in W]
plt.plot(LAM, none_zeros, 'r-.')
plt.xscale('log')
plt.title("number of non-zeros in W vs. Lambda")
plt.xlabel('log(lambda)')
plt.ylabel('number of non-zeros in W')
plt.savefig('A5_a')
#plt.yscale('log')
```



In [8]:
```
# since at position 0 is 'ViolentCrimesPerPop', -1 to each one of th
e columns below
agePct12t29 = df_train.columns.get_loc("agePct12t29") -1
pctWSocSec=df_train.columns.get_loc("pctWSocSec") -1
pctUrban=df_train.columns.get_loc("pctUrban") -1
agePct65up=df_train.columns.get_loc("agePct65up") -1
householdsize=df_train.columns.get_loc("householdsize") -1

agePct12t29_path = [z[agePct12t29] for z in W ]
pctWSocSec_path = [z[pctWSocSec] for z in W ]
pctUrban_path = [z[pctUrban] for z in W ]
agePct65up_path = [z[agePct65up] for z in W ]
householdsize_path = [z[householdsize] for z in W ]
```

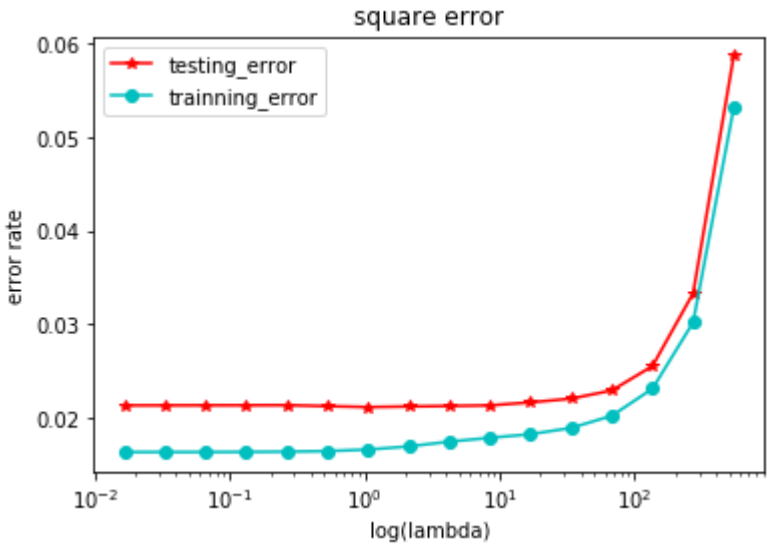
In [24]:
```
plt.plot(LAM, agePct12t29_path,'r-o')
plt.plot(LAM, pctWSocSec_path,'b-o')
plt.plot(LAM, pctUrban_path,'k-o')
plt.plot(LAM, agePct65up_path,'y-o')
plt.plot(LAM, householdsize_path,'m-o')
plt.xscale('log')
plt.yscale('linear')
plt.title("regularizatuion path")
plt.xlabel("log(lambda)")
plt.ylabel("coeff")
plt.legend(["agePct12t29_path","pctWSocSec_path","pctUrban_path",
            "agePct65up_path","householdsize_path"])
plt.savefig('A5_b')
#plt.figure(num=None, figsize=(14, 10), dpi=80, facecolor='g', edgec
olor='y')

#plt.show()

#plt.yscale('log')
```



In [13]:
```
def error(true_y, b,w,X):
    a=np.dot(X,w)
    c=a+ b
    error = np.square( c - true_y)
    return 1/X.shape[0] * np.sum(error)
```

In [14]:
```
################################## calculating errors ##############
#################3
trainning_error = [error(y,B[i],W[i],X) for i in range(len(B))]
testing_error = [error(y_test,B[i],W[i],X_test) for i in range(len(B
))]
```

In [26]:
```
plt.plot(LAM,testing_error,'r-*')
plt.plot(LAM,trainning_error,'c-o')
pos = [i for i in range(1,len(LAM))]
labelx = ["l{}".format(i) for i in range(1,len(LAM)) ]
plt.legend(["testing_error","trainning_error"])
plt.xscale('log')
plt.yscale('linear')
plt.xlabel('log(lambda)')
plt.ylabel('error rate')
plt.title("square error")
plt.savefig('A5_c')
#plt.xticks(pos,labelx)
```



In [29]:
```
Lambda30 = 30
w_yield =None
b,w = coordinate_descent(y,X,Lambda30,tolerence=1e-5,w_init=w_yield)
```

```
++++++++++++++++++++++++++++Calculating for Lambda = 30++++++++++++++
+++++++++++++++
..................... it takes 44 iterations to converg
e.....................
.......error is 9.260845594785683e-06............
```

In [30]:
```
max30= np.max(w)
indMax = w.argmax()
print("the most positive position value in w for LAMBDA=30  is{0}, t
he corresponding column name is {1}.".format(max30,df_test.columns[i
ndMax+1]))
min30 = np.min(w)
indMin=w.argmin()
print("the most negative position value in w for LAMBDA=30  is{0}, t
he corresponding column name is {1}.".format(min30,df_test.columns[i
ndMin+1]))
# -- PctIlleg: percentage of kids born to never married (numeric - d
ecimal)
#-- PctKids2Par: percentage of kids in family housing with two paren
ts (numeric - decimal)
```

```
the most positive position value in w for LAMBDA=30  is0.0689959261046
3331, the corresponding column name is PctIlleg.
the most negative position value in w for LAMBDA=30  is-0.068740519332
26402, the corresponding column name is PctKids2Par.
```