HW2

Plot of Accuracy of every 30 steps on 50 training samples(unseen)

Plot of Magnitude of coefficient vector

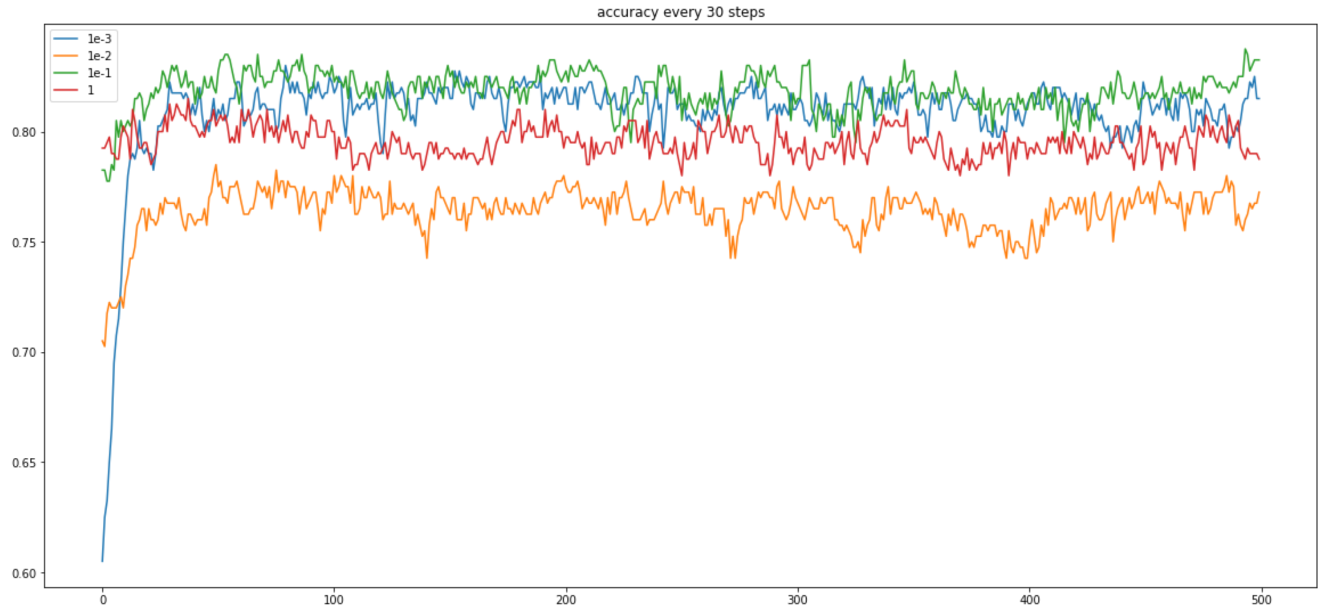
In [45]:

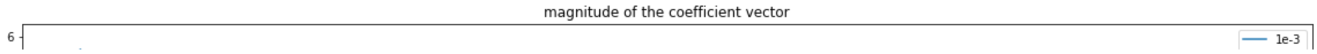
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import random
import matplotlib.pyplot as plt
%matplotlib inline

data1=pd.read_csv('adult.data.txt',header=None,usecols=[0,2,4,10,11,12,14],
                 names=['age','fnlwgt','education-num',
                        'capital-gain','capital-loss',
                        'hours-per-week','outcome'])
data2=pd.read_csv('adult.test.txt',header=None,usecols=[0,2,4,10,11,12,14],
                 names=['age','fnlwgt','education-num',
                        'capital-gain','capital-loss',
                        'hours-per-week','outcome'],skiprows=[0])
data = data1.append(data2)

# change outcome to 1,-1
data['outcome'] = np.where((data['outcome']==' <=50K.')|(data['outcome']==' <=50K'),
                     -1,1)
# 90% train, 10% validation, 10%test split
train, validate, test =np.split(data.sample(frac=1),[int(0.8*len(data)),
                                                    int(0.9*len(data))])

train = train.reset_index()
del train['index']
validate = validate.reset_index()
del validate['index']
test = test.reset_index()
del test['index']

# train set
X=train[[0,1,2,3,4,5]]
X=X.apply(lambda x: (x-np.mean(x))/np.std(x))
X=np.array(X)
y=np.array(train[[6]])

# Xval=validate[[0,1,2,3,4,5]]
# Xval=Xval.apply(lambda x: (x-np.mean(x))/np.std(x))
# yval=np.array(validate[[6]])
# Xval=np.array(Xval)
#train classifer

a_list=[]
b_list=[]
# initialization

#lambda
la=[0.001,0.01,0.1,1]

#ada=1/(0.01*1+50)
# number of epotch
n_epoch = 50
n_sample = int(len(train)/n_epoch)
n_step = 15000
n_compute =30

n_test=400
accuracy=[]
for u in range(len(la)):
    a=np.array([1,1,1,1,1,1])
    b=1
    for i in range(n_epoch):
```

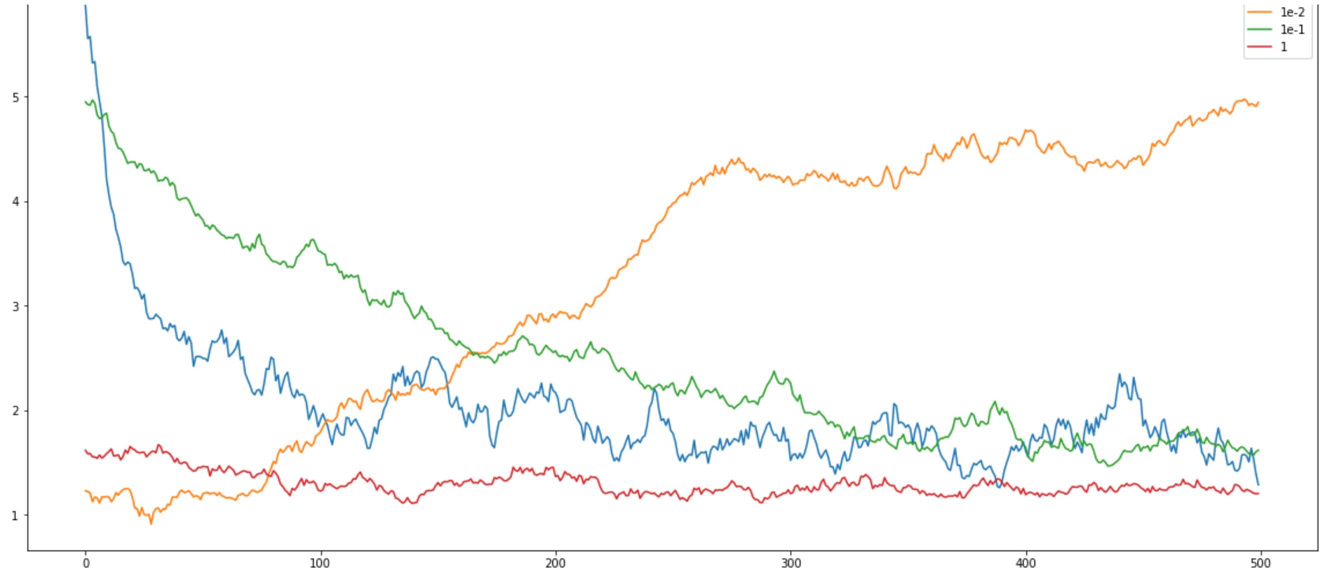
```
    for j in range(n_epoch):

        for k in range(int(n_step/n_compute)):
            for i in range(n_compute):
                c=random.randint(j*n_sample,(j+1)*n_sample-n_test-1)


                if (a.dot(X[c])+b)*y[c]>=1:
                    a=a-(1/(0.01*j+100))*la[u]*a

                else:
                    a=a-(1/(0.01*j+100))*(la[u]*a-y[c]*\
                             X[c])
                    b=b-(1/(0.01*j+100))*(-y[c])
            a_list.append(a)
            b_list.append(b)

            t=[]
            for h in range((j+1)*n_sample-n_test,(j+1)*n_sample):
                t_val=np.sign(sum(a_list[k+j*10+u*500]*X[h])+b_list[k+j*10+u*500])
                t.append(t_val)

            acc=sum(t==y[((j+1)*n_sample-n_test):(j+1)*n_sample])/n_test
            accuracy.append(acc)

#plot accuracy
plt.figure(figsize=(20,20))
plt.subplot(2,1,1)
plt.plot(accuracy[:500],label='1e-3')
plt.plot(accuracy[500:1000],label='1e-2')
plt.plot(accuracy[1000:1500],label='1e-1')
plt.plot(accuracy[1500:2000],label='1')
plt.legend()
plt.title('accuracy every 30 steps')
#plot magnitude of coefficient
mag=[]
for i in range(len(a_list)):
    mag_value = a_list[i].dot(a_list[i])
    mag.append(mag_value)

plt.figure(figsize=(20,20))
plt.subplot(2,1,2)
plt.plot(mag[:500],label='1e-3')
plt.plot(mag[500:1000],label='1e-2')
plt.plot(mag[1000:1500],label='1e-1')
plt.plot(mag[1500:2000],label='1')
plt.legend()
plt.title('magnitude of the coefficient vector')
```

Out[45]:

```
Text(0.5,1,'magnitude of the coefficient vector')
```







Compute the accuracy at the end of each epoch on the validation set

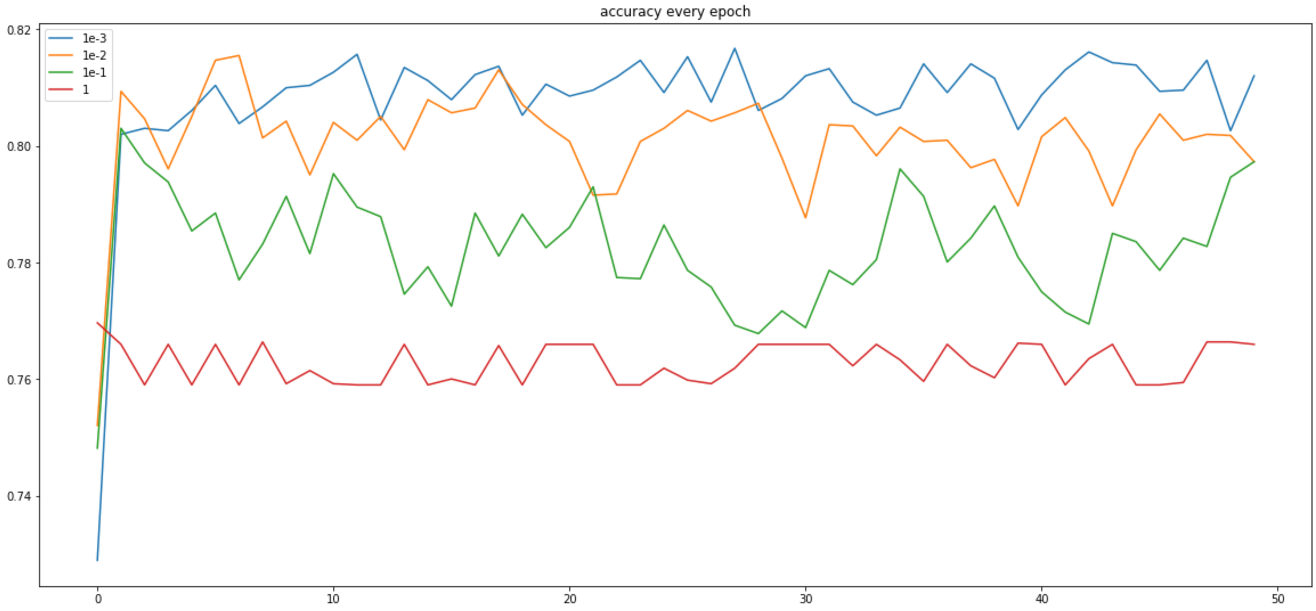
In [43]:

```
data1=pd.read_csv('adult.data.txt',header=None,usecols=[0,2,4,10,11,12,14],
                 names=['age','fnlwgt','education-num',
                        'capital-gain','capital-loss',
                        'hours-per-week','outcome'])
data2=pd.read_csv('adult.test.txt',header=None,usecols=[0,2,4,10,11,12,14],
                 names=['age','fnlwgt','education-num',
                        'capital-gain','capital-loss',
                        'hours-per-week','outcome'],skiprows=[0])
data = data1.append(data2)

# change outcome to 1,-1
data['outcome'] = np.where((data['outcome']==' <=50K.')|(data['outcome']==' <=50K'),
                     -1,1)
# 90% train, 10% validation, 10%test split
train, validate, test =np.split(data.sample(frac=1),[int(0.8*len(data)),
                                                    int(0.9*len(data))])

train = train.reset_index()
del train['index']
validate = validate.reset_index()
del validate['index']
test = test.reset_index()
del test['index']

# train set
X=train[[0,1,2,3,4,5]]
X=X.apply(lambda x: (x-np.mean(x))/np.std(x))
X=np.array(X)
y=np.array(train[[6]])

Xval=validate[[0,1,2,3,4,5]]
Xval=Xval.apply(lambda x: (x-np.mean(x))/np.std(x))
yval=np.array(validate[[6]])
Xval=np.array(Xval)
#train classifer

a_list=[]
b_list=[]
# initialization

#lambda
la=[0.001,0.01,0.1,1]

#ada=1/(0.01*1+50)
# number of epotch
n_epoch = 50
n_sample = int(len(train)/n_epoch)
n_step = 300
n_compute =30
```

```
accuracy=[]

for u in range(len(la)):
    a=np.array([1,1,1,1,1,1])
    b=1
    for j in range(n_epoch):

        for k in range(n_step): #training epoch
            c=random.randint(j*n_sample,(j+1)*n_sample-1)


            if (a.dot(X[c])+b)*y[c]>=1:
                a=a-(1/(0.01*j+100))*la[u]*a

            else:
                a=a-(1/(0.01*j+100))*(la[u]*a-y[c]*\
                         X[c])
                b=b-(1/(0.01*j+100))*(-y[c])
        a_list.append(a)
        b_list.append(b)

for i in range(200):
    t=[]
    for k in range(len(Xval)):
        t_val=np.sign(sum(a_list[i]*Xval[k])+b_list[i])
        t.append(t_val)

    acc=sum(t==yval)/len(yval)
    accuracy.append(acc)

#plot accuracy
plt.figure(figsize=(20,20))
plt.subplot(2,1,1)
plt.plot(accuracy[:50],label='1e-3')
plt.plot(accuracy[50:100],label='1e-2')
plt.plot(accuracy[100:150],label='1e-1')
plt.plot(accuracy[150:200],label='1')
plt.legend()
plt.title('accuracy every epoch')
```

Out[43]:

```
Text(0.5,1,'accuracy every epoch')
```



It shows lambda equals 1e-3 has a higher accuracy on validation set, so it will be chosen for the final model

I will combine training and validation set, and compute the accuracy on the test set

In [56]:

```
#choose lambda=0.001
```

```
train_b=train.append(validate)

X_b = train_b[[0,1,2,3,4,5]]
y_b = np.array(train_b[[6]])
X_b= X_b.apply(lambda x: (x-np.mean(x))/np.std(x))
X_b = np.array(X_b)


X_test = test[[0,1,2,3,4,5]]
y_test = np.array(test[[6]])
X_test= X_test.apply(lambda x: (x-np.mean(x))/np.std(x))
X_test = np.array(X_test)


la=0.001
#initialization
a=np.array([1,1,1,1,1,1])
b=1
for i in range(len(X_b)):

    if (sum(a.T*X_b[i])+b)*y_b[i]>=1:
        a=a-(1/(0.01+50))*la*a

    else:
        a=a-(1/(0.01+50))*(la*a-y_b[i]*\
                 X_b[i])
        b=b-(1/(0.01+50))*(-y_b[i])

# compute accuracy
t=[]
for i in range(len(X_test)):
    t_val = np.sign(sum(a*X_test[i])+b)
    t.append(t_val)

acc=sum(t==y_test)/len(y_test)
print('the estimate of the accuracy',acc)
```

```
the estimate of the accuracy [ 0.80122825]
```

In [ ]: