# SSVS

April 27, 2018

```
In [1]: import numpy as np
        import pandas as pd
        import scipy
        #import seaborn as sns
        from numpy.linalg import inv
        from scipy.stats import invgamma
        from numpy import linalg as La
        import matplotlib.pyplot as plt
        from scipy.stats import itemfreq
        from scipy.stats import multivariate_normal
        import pdb as db
```

## 0.1 Experiment 1.1

```
In [20]: mu, sigma = 0, 1 # mean and standard deviation
         m,n=5,60
         data=np.zeros((n,m))

         for i in range(m):
             s = np.random.normal(mu, sigma, n)
             data[:,i]=s

         df=pd.DataFrame(data)
         df=df.rename(columns={0:'x1',1:'x2',2:'x3',3:'x4',4:'x5'})

         mu, sigma = 0, 2.5
         eps = np.random.normal(mu, sigma, n)
         target=df['x4']+1.2*df['x5']+eps
```

```
In [7]: df.head()
```

```
Out[7]:          x1        x2        x3        x4        x5
        0  0.317640  1.240063  0.157526  0.790140 -1.043167
        1  1.471331 -0.042229  0.415386 -0.732802 -0.784271
        2  0.453375 -0.003970  0.726393  0.384540  0.720648
        3  0.340917 -0.508111  0.502497 -0.967315 -0.253593
        4  0.086766 -0.188309 -0.212977 -0.701916  2.125630
```

```
In [21]: maxiters=5000
         data=df.values
         y=target.values
         pi,c,lambda_gamma,v=1/2, 10 ,1,0#data.shape[0]
         R=df.corr().values
         #R=np.zeros((5,5))
         #np.fill_diagonal(R,1)

         beta=np.zeros((maxiters,m))
         sigma=np.zeros((maxiters,1))
         r=np.zeros((maxiters,m))

         r[0]=np.ones(m)
         a=inv(np.matmul(data.T,data))
         b=np.matmul(data.T,y)
         beta[0]=np.matmul(a,b)

         sigma[0]=np.sqrt((y-(beta[0]*data).sum(1)).var())
         #db.set_trace()

         a=np.zeros((maxiters,m))
         a[np.where(r==0)[0],np.where(r==0)[1]]=1
         a[np.where(r==1)[0],np.where(r==1)[1]]=c

         ssxx=((data-data.mean(0))**2).sum(0)
         tau=sigma/np.sqrt(ssxx)

         temp=a*tau
         D=[]
         for i in range(temp.shape[0]):
             D.append(np.diag(temp[0]))
         D=np.array(D)
```

```
In [58]: def get_A(sigma_1,x,D_1,R):
             try:
                 A = sigma_1**(-2)*np.matmul(x.T,x)+np.matmul(np.matmul(inv(D_1).T,
             except:
                 db.set_trace()
             return inv(A)

         def get_beta(sigma_1,x,D_1,R,beta_ls):

             beta_ls=beta_ls.reshape((beta_ls.shape[0],1))
             A=get_A(sigma_1,x,D_1,R)

             xx=np.matmul(x.T,x)
             temp=np.matmul(A,xx)  #get ride of transpose!!!!
             temp=np.matmul(temp,beta_ls)# 5*1
```

```
    mean=(sigma_1**(-2)*temp).reshape(temp.shape[0])
    cov=A#5*5
    #db.set_trace()
    beta=np.random.multivariate_normal(mean, cov, 1)[0]
    #db.set_trace()
    return beta

def get_sigma(n,y,beta,r,lambda_gamma,v):


    err=((y-(beta*data).sum(1))**2).sum()
    a=(n+v)/2
    scale=(err+v*lambda_gamma)/2
    sig=invgamma.rvs(a=a,loc=0,scale=scale,size=1)
    #db.set_trace()
    return sig

def get_gamma(idx,beta):

    r[idx]=r[idx-1]
    sig=np.sqrt((y-(beta*data).sum(1)).var())
    tau[idx]=sig/np.sqrt(ssxx)

    a1=np.zeros((m))
    a1[r[idx]==0]=1
    a1[r[idx]==1]=c
    d=np.diag(a1*tau)
    for i in range(0,len(beta)):

        a1[i]=c
        d1=np.diag(a1*tau[idx])
        mean1,sigma1=0,np.matmul(np.matmul(d1.T,R),d1)
        aa=multivariate_normal.pdf(beta, mean=np.zeros(sigma1.shape[0]), c
        #aa=(1/np.sqrt(La.norm(sigma1)))*np.exp(-0.5*np.matmul(np.matmul(b
        aa*=pi

        a2=a1.copy()
        a2[i]=1
        d2=np.diag(a2*tau[idx])
        mean2,sigma2=0,np.matmul(np.matmul(d2.T,R),d2)
        bb=multivariate_normal.pdf(beta, mean=np.zeros(sigma2.shape[0]), c
#        bb=(1/np.sqrt(La.norm(sigma2)))*np.exp(-0.5*np.matmul(np.matmul(b
        bb*=(1-pi)

        if (aa+bb)!=0:
            p=aa/(aa+bb)
        else:
```

```
                    db.set_trace()

                #db.set_trace()

                if p<0.5:
                    r[idx,i]=0
                else:
                    r[idx,i]=1

            return r[idx]
```

```
In [23]: for i in range(1,len(r)):#len(r)):
             #db.set_trace()
             beta[i]=get_beta(sigma[i-1],data,D[i-1],R,beta[0])
             sigma[i]= get_sigma(n,y,beta[i],r[i-1],lambda_gamma,v)
             r[i]=get_gamma(i,beta[i])
             #db.set_trace()
```

```
In [24]: unique_elements, counts_elements = np.unique(r[2500:],axis=0, return_count
         rank=list(zip(unique_elements, counts_elements))
         rank=sorted(rank, key=lambda rank: rank[1],reverse=True)
         rank[:5]
```

```
Out[24]: [(array([1., 0., 0., 1., 1.]), 113),
          (array([1., 1., 0., 1., 0.]), 108),
          (array([0., 0., 1., 1., 1.]), 107),
          (array([1., 0., 1., 0., 1.]), 102),
          (array([0., 1., 1., 1., 0.]), 100)]
```

## 0.2 Experiment 1.2

```
In [25]: mu, sigma = 0, 1 # mean and standard deviation
         m,n=5,60
         data=np.zeros((n,m))

         for i in range(m):
             s = np.random.normal(mu, sigma, n)
             data[:,i]=s

         df=pd.DataFrame(data)
         df=df.rename(columns={0:'x1',1:'x2',2:'x3',3:'x4',4:'x5'})
         df['x3']=df['x5']+0.15*np.random.normal(0,1,60)
         mu, sigma = 0, 2.5
         eps = np.random.normal(mu, sigma, n)
         target=df['x4']+1.2*df['x5']+eps
```

```
In [26]: maxiters=5000
         data=df.values
         y=target.values
         pi,c,lambda_gamma,v=1/2, 10 ,1,0#data.shape[0]
         R=df.corr().values
         #R=np.zeros((5,5))
         #np.fill_diagonal(R,1)

         beta=np.zeros((maxiters,m))
         sigma=np.zeros((maxiters,1))
         r=np.zeros((maxiters,m))

         r[0]=np.ones(m)
         a=inv(np.matmul(data.T,data))
         b=np.matmul(data.T,y)
         beta[0]=np.matmul(a,b)

         sigma[0]=np.sqrt((y-(beta[0]*data).sum(1)).var())
         #db.set_trace()

         a=np.zeros((maxiters,m))
         a[np.where(r==0)[0],np.where(r==0)[1]]=1
         a[np.where(r==1)[0],np.where(r==1)[1]]=c

         ssxx=((data-data.mean(0))**2).sum(0)
         tau=sigma/np.sqrt(ssxx)

         temp=a*tau
         D=[]
         for i in range(temp.shape[0]):
             D.append(np.diag(temp[0]))
         D=np.array(D)
```

```
In [44]: for i in range(len(R)):
             print(R[i])
```

```
[ 1.         -0.16164442 -0.06060787 -0.16244333 -0.06743702]
[-0.16164442  1.          0.0779122   0.11722447  0.06918207]
[-6.06078718e-02  7.79121963e-02  1.00000000e+00  3.08230900e-04
  9.82596966e-01]
[-1.62443335e-01  1.17224472e-01  3.08230900e-04  1.00000000e+00
  6.48455304e-02]
[-0.06743702  0.06918207  0.98259697  0.06484553  1.        ]
```

```
In [27]: for i in range(1,len(r)):#len(r)):
             #db.set_trace()
             beta[i]=get_beta(sigma[i-1],data,D[i-1],R,beta[0])
             sigma[i]= get_sigma(n,y,beta[i],r[i-1],lambda_gamma,v)
             r[i]=get_gamma(i,beta[i])
```

```
            #db.set_trace()
```

```
In [28]: unique_elements, counts_elements = np.unique(r[2500:],axis=0, return_count
         rank=list(zip(unique_elements, counts_elements))
         rank=sorted(rank, key=lambda rank: rank[1],reverse=True)
         rank[:5]
```

```
Out[28]: [(array([0., 1., 1., 1., 1.]), 322),
          (array([0., 0., 1., 1., 1.]), 277),
          (array([1., 0., 1., 1., 1.]), 276),
          (array([0., 1., 1., 0., 1.]), 230),
          (array([1., 1., 1., 1., 1.]), 224)]
```

### 0.3 Experiment 2.1

```
In [89]: def normalize(x):
             for fea in list(x):
                 #db.set_trace()
                 if x[fea].dtype in ['float32','int64','float64','int32'] and \
                                   len(x[fea].value_counts().values)>45:
                     interval=x[fea].quantile([0.001,0.999]).values
                     x[fea]=(x[fea]-interval[0])/(interval[1]-interval[0])
             return x
```

```
In [90]: import pickle

         path='/Users/yanxinzhou/course/review/2016_NYC_Yellow_Cab_trip_record_data
         pkl_file = open(path+'2018-03-31_00_38_train.pkl', 'rb')
         df=pickle.load(pkl_file)

         fea=[
          'trip_duration',
          'distance',
          't_sin_hour',
          't_cos_hour',
          't_sin_day',
          't_cos_day',
          'holiday',
          'number_of_steps',
          'total_distance',
          'minimum temperature'
         ]

         train=df[fea]
         train=normalize(train)
         target=train['trip_duration']
         del train['trip_duration']
         train=train[:5000]
         target=target[:5000]
```

```
/Users/yanxinzhou/.pyenv/versions/anaconda3-4.2.0/lib/python3.5/site-packages/ipyke
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
```

```
In [106]: maxiters=5000
          data=train.values
          y=target.values
          m,n=len(list(train)),len(data)
          pi,c,lambda_gamma,v=0.5, 50 ,1,0#data.shape[0]
          R=train.cov().values
          #R=np.zeros((5,5))
          #np.fill_diagonal(R,1)

          beta=np.zeros((maxiters,m))
          sigma=np.zeros((maxiters,1))
          r=np.zeros((maxiters,m))

          r[0]=np.ones(m)
          a=inv(np.matmul(data.T,data))
          b=np.matmul(data.T,y)
          beta[0]=np.matmul(a,b)

          sigma[0]=np.sqrt((y-(beta[0]*data).sum(1)).var())
          #db.set_trace()

          a=np.zeros((maxiters,m))
          a[np.where(r==0)[0],np.where(r==0)[1]]=1
          a[np.where(r==1)[0],np.where(r==1)[1]]=c

          ssxx=((data-data.mean(0))**2).sum(0)
          tau=sigma/np.sqrt(ssxx)

          temp=a*tau
          D=[]
          for i in range(temp.shape[0]):
              D.append(np.diag(temp[0]))
          D=np.array(D)
```

```
In [107]: for i in range(1,len(r)):#len(r)):
              #db.set_trace()
              beta[i]=get_beta(sigma[i-1],data,D[i-1],R,beta[0])
              sigma[i]= get_sigma(n,y,beta[i],r[i-1],lambda_gamma,v)
```

```
            r[i]=get_gamma(i,beta[i])
            #db.set_trace()
```

```
In [108]: unique_elements, counts_elements = np.unique(r[2500:],axis=0, return_cour
          rank=list(zip(unique_elements, counts_elements))
          rank=sorted(rank, key=lambda rank: rank[1],reverse=True)
          rank[:5]
```

```
Out[108]: [(array([1., 0., 1., 0., 0., 1., 0., 1., 1.]), 2463),
           (array([1., 0., 1., 1., 0., 1., 0., 1., 1.]), 37)]
```

```
In [109]: beta[:-5]
```

```
Out[109]: array([[ 0.01330566, -0.00016276, -0.00195076, ...,  0.00065043,
                   0.01001904,  0.00535774],
                 [ 0.00601061, -0.00097346, -0.00231858, ...,  0.00051156,
                   0.01158838,  0.00591838],
                 [ 0.01383481, -0.00013601, -0.00191049, ...,  0.00066141,
                   0.00934152,  0.00534373],
                 ...,
                 [ 0.01297934, -0.0001991 , -0.00198616, ...,  0.00065216,
                   0.01034965,  0.00532368],
                 [ 0.01327226, -0.00016446, -0.0019728 , ...,  0.00064695,
                   0.01010219,  0.0054205 ],
                 [ 0.01189063, -0.00017657, -0.00197751, ...,  0.00064067,
                   0.01149391,  0.00541989]])
```

```
In [ ]:
```