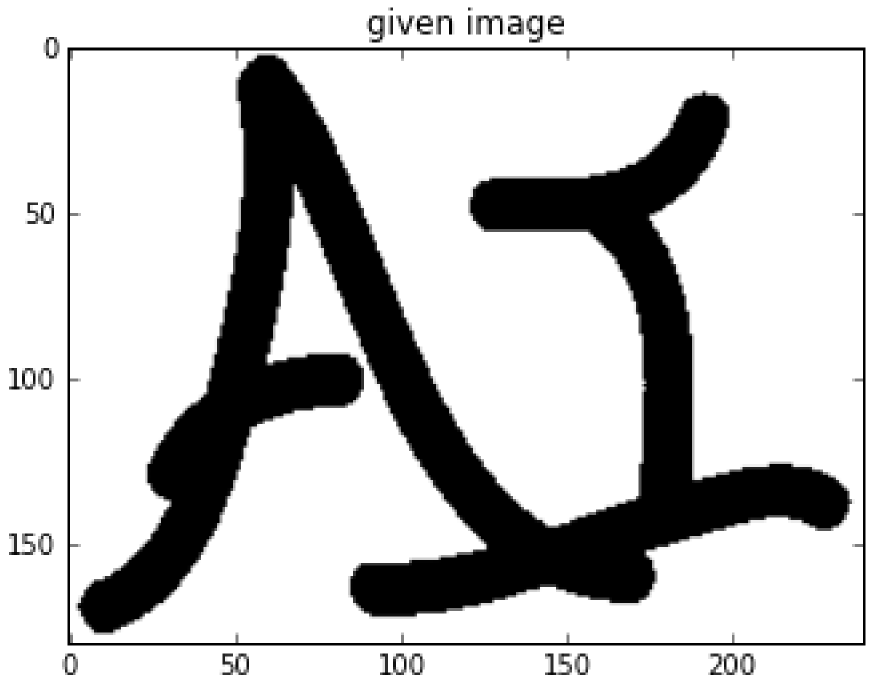
simulated_annealing

June 9, 2016

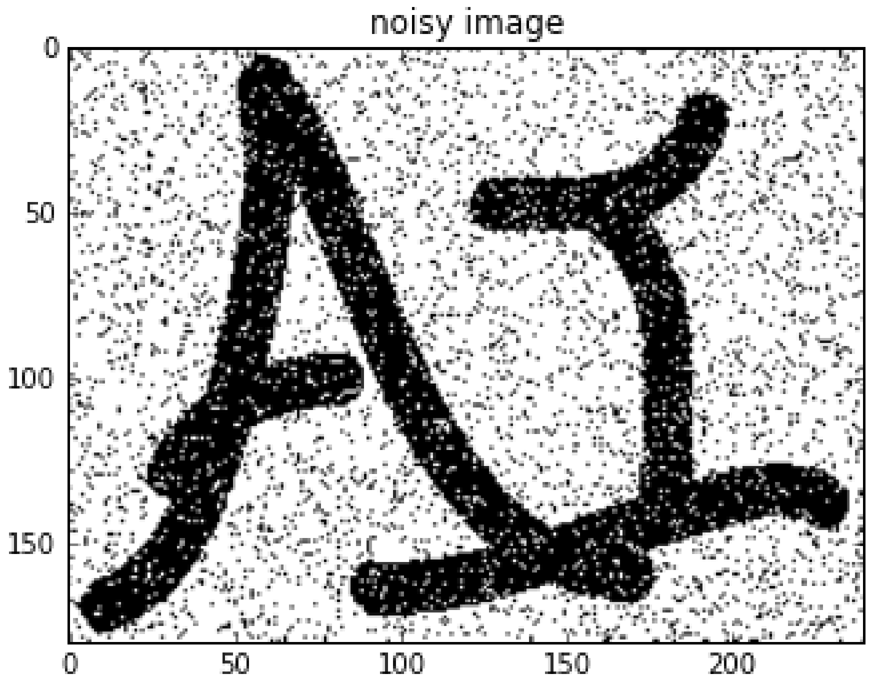
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from PIL import Image
        %matplotlib inline

In [2]: I=Image.open('in.png')
        plt.imshow(I,cmap=plt.cm.gray)
        plt.title('given image')
        plt.show()
```



```
In [3]: def change(y,translate):
            data=np.array(y)
            return np.vectorize(lambda x: translate[x])(data)
        def sign(y,density):
```

```
        a,b=len(y),np.array(y)
        for i in range(a):
            p=np.random.rand()
            if p<density:
                b[i]=-1*b[i]
        return b
```

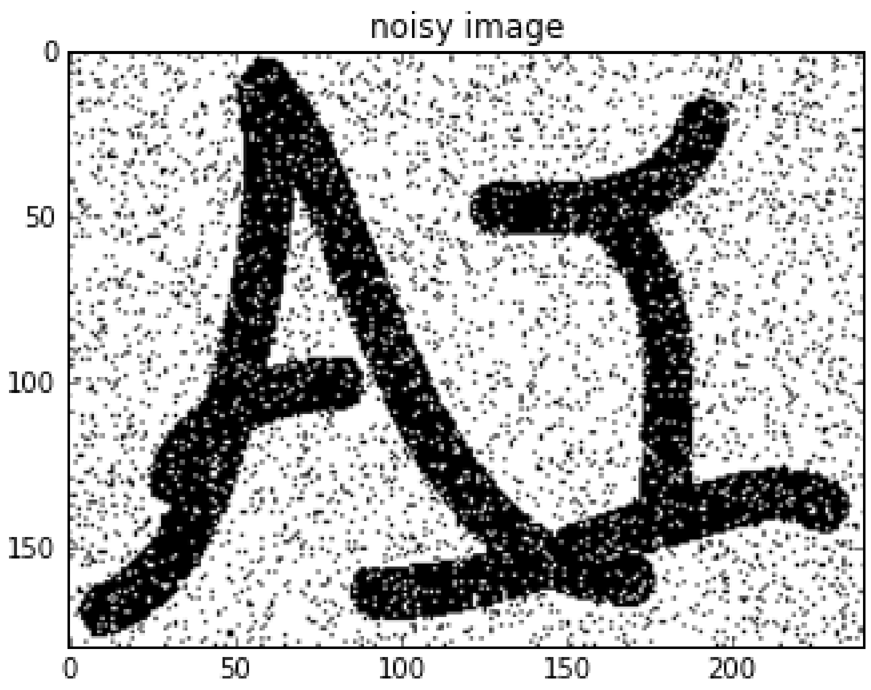
```
In [4]: mydata=change(I.getdata(),{0:-1,255:1})
        mysigndata=sign(mydata,0.1)
        mysigndata=mysigndata.reshape(np.array(I).shape)
        plt.imshow(mysigndata,cmap=plt.cm.gray)
        plt.title('noisy image')
        plt.show()
```

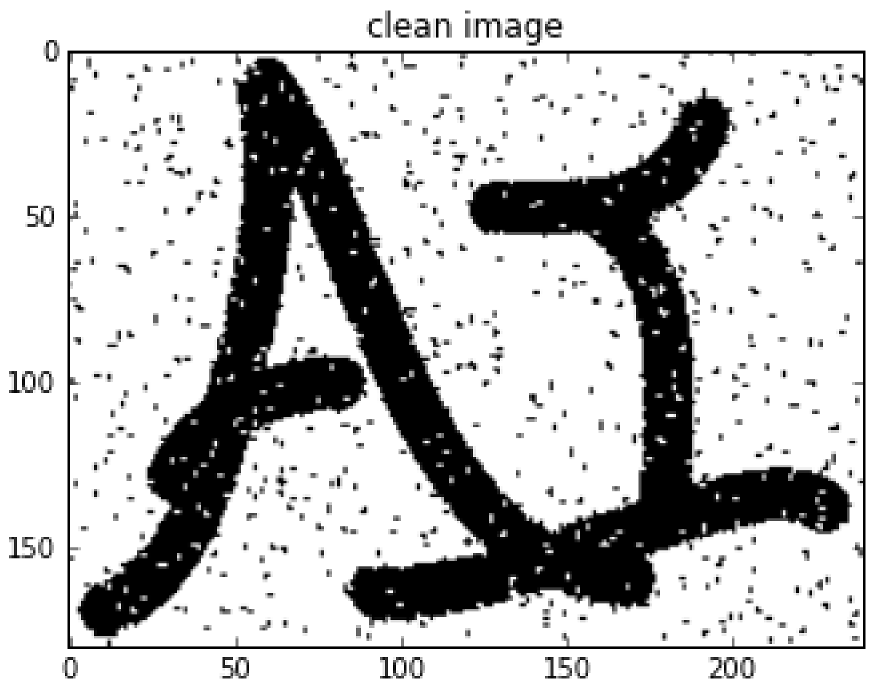


```
In [5]: def Energy(beta,eta,h):

            def possible(i,j,shape):
                return i>=0 and j>=0 and i<shape[0] and j<shape[1]

            def E_old(i,j,x,y):
                old_value=x[i,j]
                E1=h*old_value-eta*old_value*y[i,j]
                fours=[(-1,0),(1,0),(0,-1),(0,1)]
                neighbor=[x[i+di,j+dj] for di, dj in fours if possible(i+di,j+dj,x.
                E1=E1-beta*sum(value*old_value for value in neighbor)
                return E1

            def E_new(i,j,x,y):
                old_value=x[i,j]
                new_value=-1*old_value
                E2=h*new_value-eta*new_value*y[i,j]
                fours=[(-1,0),(1,0),(0,-1),(0,1)]
                neighbor=[x[i+di,j+dj] for di, dj in fours if possible(i+di,j+dj,x.
                E2=E2-beta*sum(value*new_value for value in neighbor)
                return old_value,new_value,E2
            return E_old,E_new
```

```
In [6]: E_old,E_new=Energy(1e-3,2.1e-3,0)
        mydata=change(I.getdata(),{0:-1,255:1})
        mysigndata=sign(mydata,0.1)
        mysigndata=mysigndata.reshape(np.array(I).shape)
        x=mysigndata
        y=mysigndata

        for idx in np.ndindex(y.shape):
            E1=E_old(idx[0],idx[1],x,y)

            old_value,new_value,E2=E_new(idx[0],idx[1],x,y)

            if E2<E1:
                E1,x[idx]=E2,new_value
            else:
                 E1,x[idx]=E2,old_value
```

```
In [7]: mydata=change(I.getdata(),{0:-1,255:1})
        mysigndata=sign(mydata,0.1)
        mysigndata=mysigndata.reshape(np.array(I).shape)
        plt.imshow(mysigndata,cmap=plt.cm.gray)
        plt.title('noisy image')
        plt.show()
        plt.imshow(x,cmap=plt.cm.gray)
        plt.title('clean image')
        plt.show()
```





```
In [8]: def prob(E1,E2,t):
            return 1 if E1>E2 else np.exp((E1-E2)/t)
```

```
        def temperature(k,kmax):
            return 1.0/500*(1.0/k-1.0/kmax)

In [23]: kmax=5
         E_old,E_new=Energy(1e-3,2.1e-3,0)
         mydata=change(I.getdata(),{0:-1,255:1})
         mysigndata=sign(mydata,0.1)
         mysigndata=mysigndata.reshape(np.array(I).shape)
         x=mysigndata
         y=mysigndata

         E_list=[]
         for k in range(1,kmax):
             t=temperature(k,kmax)
             accept,reject=0,0
             E_total=[]
             for idx in np.ndindex(y.shape):
                 E1=E_old(idx[0],idx[1],x,y)
                 Ebest=E1
                 old_value,new_value,E2=E_new(idx[0],idx[1],x,y)


                 p,q=prob(E1,E2,t),np.random.rand()
                 if p>q:
                     accept+=1
                     E1,x[idx]=E2,new_value

                     if (E2<Ebest):
                         Ebest=E2
                 else:
                     reject+=1
                     E1,x[idx]=E1,old_value


                     E_total.append(Ebest)


             E_list.append(np.sum(E_total))
             print 'iteration=%d,temp =%f,accept=%d,reject=%d' %(k,t,accept,reject)
```

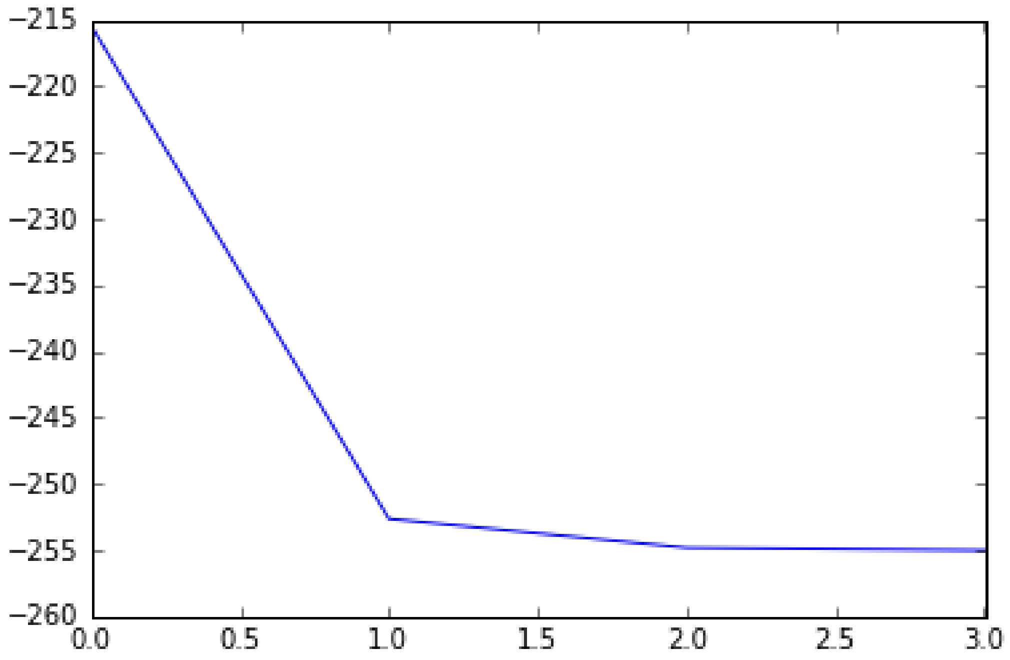
```
iteration=1,temp =0.001600,accept=4090,reject=39110
iteration=2,temp =0.000600,accept=237,reject=42963
iteration=3,temp =0.000267,accept=24,reject=43176
iteration=4,temp =0.000100,accept=0,reject=43200
```

```
In [24]: print np.array(E_total).shape
```

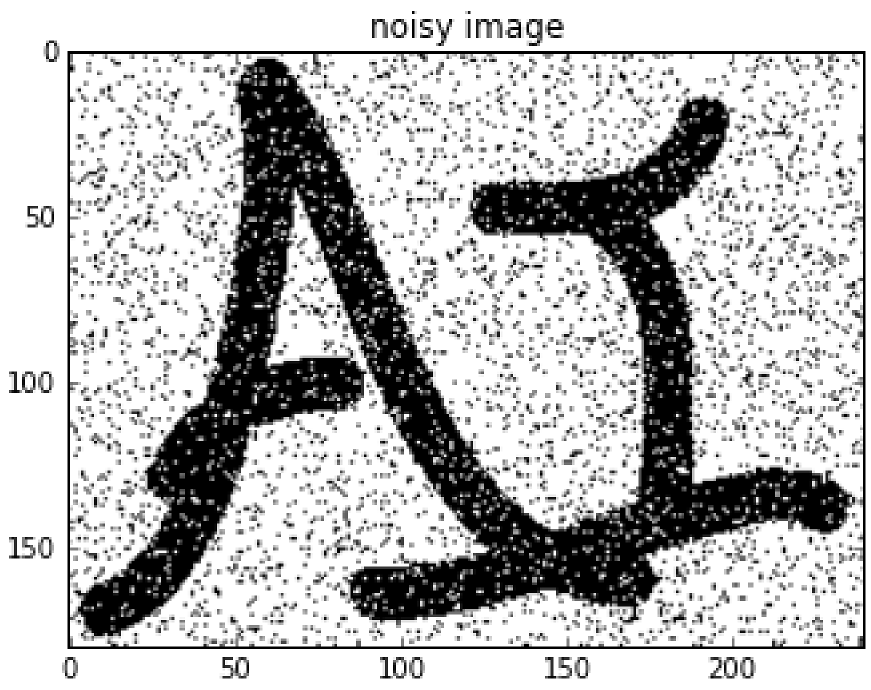
```
        plt.plot(E_list)
```

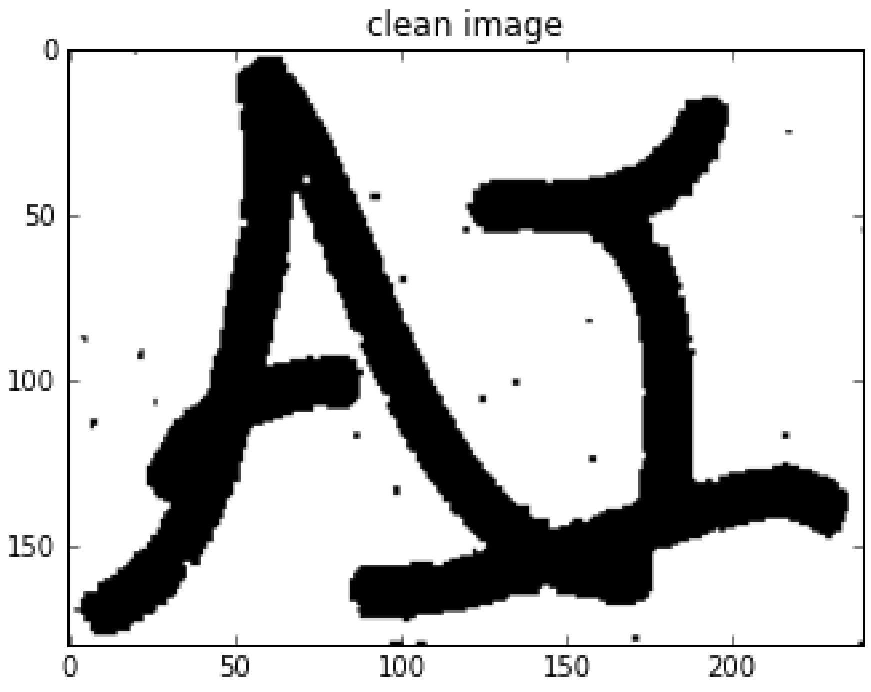
```
(43200L,)
```

```
Out[24]: [<matplotlib.lines.Line2D at 0x8e4f048>]
```



```
In [25]: mydata=change(I.getdata(),{0:-1,255:1})
         mysigndata=sign(mydata,0.1)
         mysigndata=mysigndata.reshape(np.array(I).shape)
         plt.imshow(mysigndata,cmap=plt.cm.gray)
         plt.title('noisy image')
         plt.show()
         plt.imshow(x,cmap=plt.cm.gray)
         plt.title('clean image')
         plt.show()
```


noisy image
0
50
100
150
0
50
100
150
200


clean image
0
50
100
150
0
50
100
150
200

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```