Machine Intelligence 2

Exercise 08

Stochastic Optimization

Group Members:
Xugang Zhou
Fangzhou Yang

Tutor:
Timm Lochmann

June 19, 2013

1 8.1 Simulated Annealing

```
from numpy import *
from math import *
import random

def E(w,s):
    tmp = 0.
    for i in range (len(s)):
        for j in range (len(s)):
            tmp += w[i,j] * s[i] * s[j]
    tmp = -tmp/2
    return tmp
def Esi(w,s,i):
    tmp = 0
    for j in range (len(s)):
        tmp += w[i,j] * s[i] * s[j]
    tmp = -tmp/2
    return tmp
def flip(s,deltaE,i,beta):
    if(beta*deltaE>100):
        p = 0
    else:
        p = (1. + math.exp(beta*deltaE) )**(-1)
    #print p, beta
    if (random.random()) < p:
        s[i] = -s[i]
        #print ’changed’
```

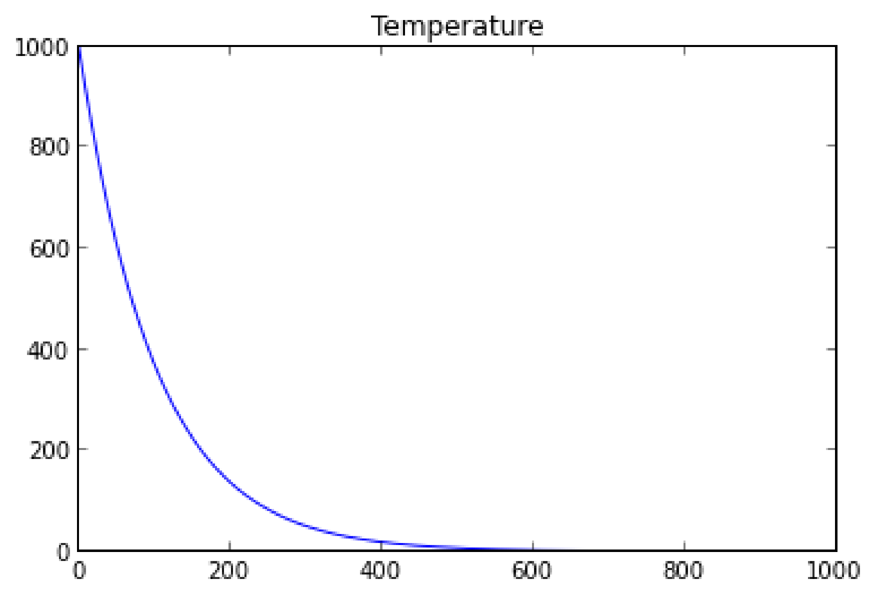
```
#Initialization
beta0 = 0.001
tao =1.01
tmax = 1000
random.seed(100)
N = 6
Et = [ 0. for i in range(tmax)]
beta = [0. for i in range(tmax)]
beta[0] = beta0

S = [0 for i in range (N)]
for i in range(N):
    if(random.random()>0.5):
        S[i] = 1
    else:
        S[i] = -1
W = matrix([[0. for i in range(N)]for j in range (N)])
for i in range(N):
    for j in range(N):
        if(i==j): W[i,j] =0
        if(i<j): W[i,j] = random.random()
        if(i>j): W[i,j] = W[j,i]
```

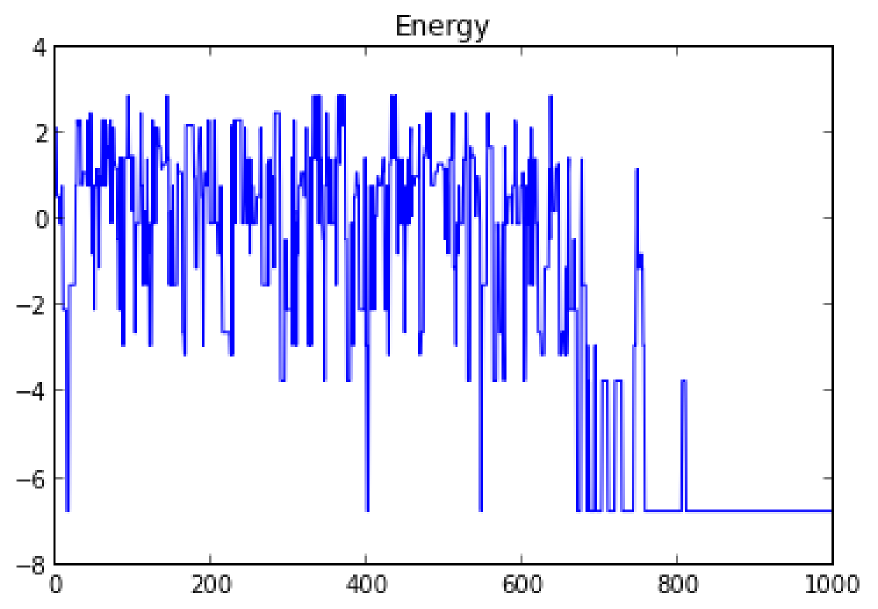
```
print 'beta0:', beta0
print 'tao:',tao
print 'tmax:',tmax
print 'W:'
print W
print 'S', S
```

```
beta0: 0.001
tao: 1.01
tmax: 1000
W:
[[ 0.          0.80002046  0.53290141  0.08015371  0.45594588  0.04788752]
 [ 0.80002046  0.          0.9329624   0.94707801  0.33535078  0.30940593]
 [ 0.53290141  0.9329624   0.          0.76801815  0.20386953  0.17846076]
 [ 0.08015371  0.94707801  0.76801815  0.          0.18859491  0.34700445]
 [ 0.45594588  0.33535078  0.20386953  0.18859491  0.          0.62632164]
 [ 0.04788752  0.30940593  0.17846076  0.34700445  0.62632164  0.        ]]
S [-1, -1, 1, 1, 1, -1]
```

```
#optimization
for t in range(tmax):
    i = random.randint(0,5)
    Et[t] = E(W,S)
    localE = Esi(W,S,i)
    deltaE = -2 * localE
    flip(S,deltaE,i,beta[t])
    if((t+1) < tmax):
        beta[t+1] = tao*beta[t]
```

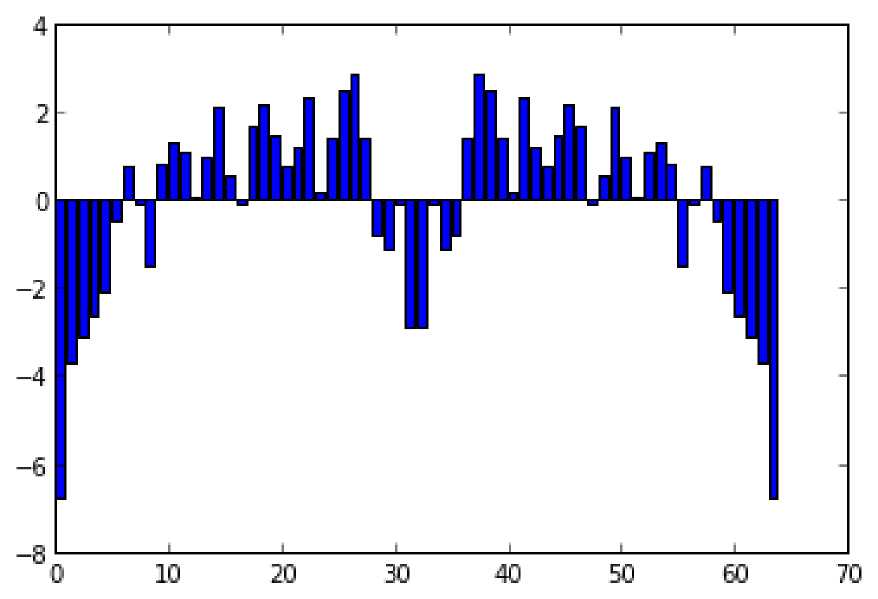
```
#plotting
import matplotlib
import matplotlib.pyplot as plt


Tt = [ 1/beta[t] for t in range (tmax)]

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(Tt)
ax.set_title("Temperature")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(Et)
ax.set_title("Energy")
plt.show()

print S
```



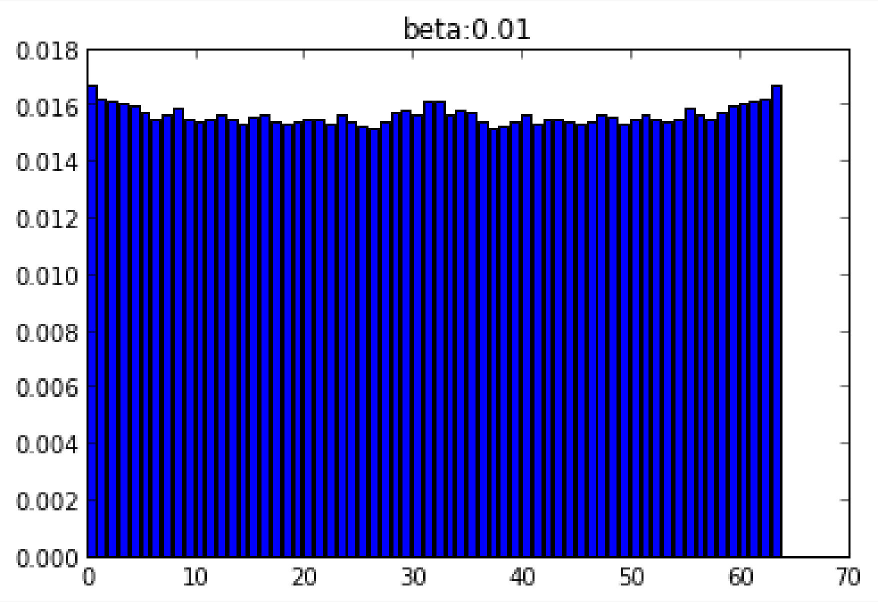


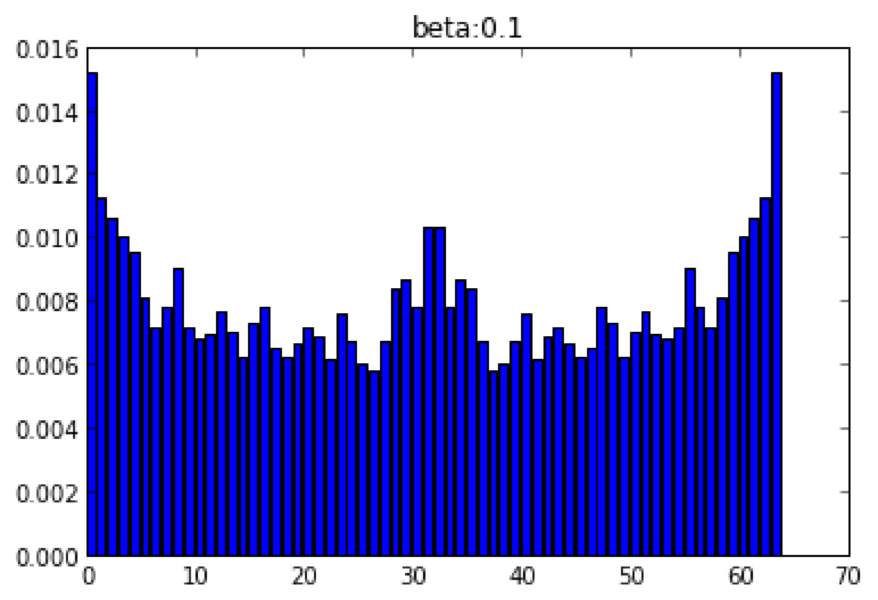
```
[-1, -1, -1, -1, -1, -1]
```

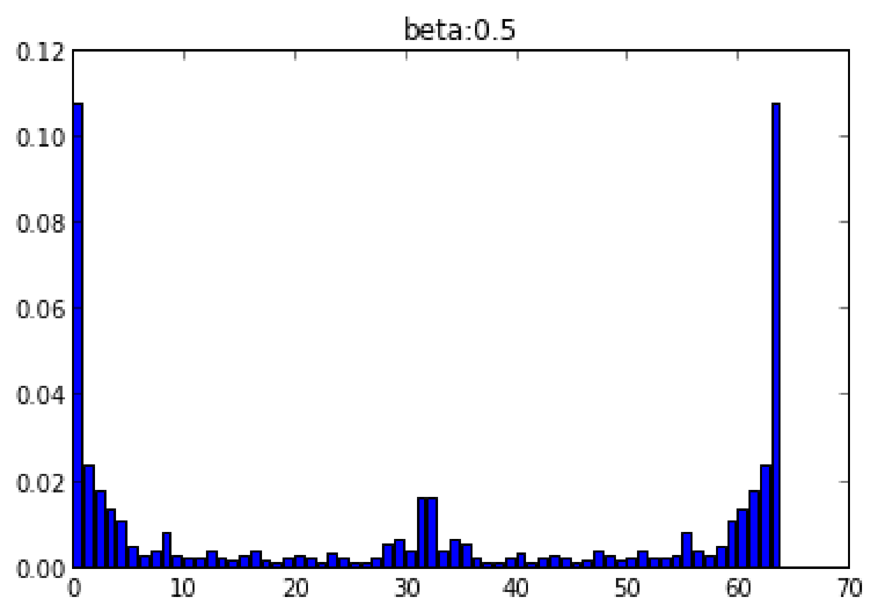
```
m = [0 for i in range(6)]
S_all = [[0 for p in range(6)] for q in range(64)]
tmp = 0
E_all = [0 for i in range(64)]
for m[0] in range(2):
    for m[1] in range(2):
        for m[2] in range(2):
            for m[3] in range(2):
                for m[4] in range(2):
                    for m[5] in range(2):
                        for i in range(6):
                            if(m[i] == 0):
                                S_all[tmp][i] = -1
                            else:
                                S_all[tmp][i] = 1
                        E_all[tmp] = E(W,S_all[tmp])
                        tmp = tmp +1

ind = np.arange(64)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.bar(ind, E_all)
plt.show()
```

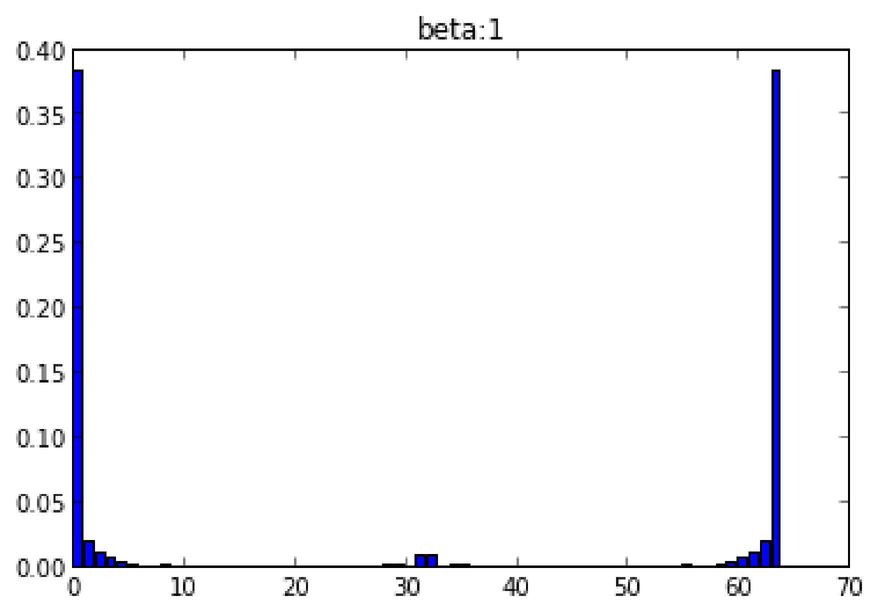


```
Z = 0

beta = [0.01, 0.1, 0.5, 1]

P = [[0 for i in range(64)] for j in range(len(beta))]

for m in range (len(beta)):
    for i in range (64):
        Z += math.exp(-beta[m]*E(W,S_all[i]) )
    for i in range (64):
        P[m][i] = math.exp(-beta[m]*E(W,S_all[i]))/Z

    ind = np.arange(64)
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.set_title(’beta:’+ (str)(beta[m]))
    ax.bar(ind, P[m])
    plt.show()
```









2 8.2 Mean-Field Annealing

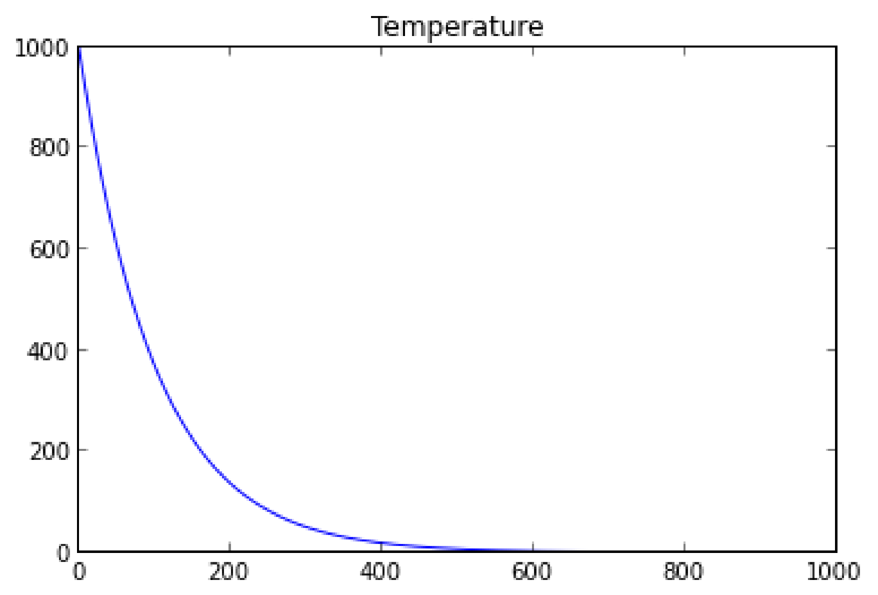
```
def e(w,s,i):
    tmp = 0.
    for j in range(len(s)):
        tmp = w[i,j] * s[j]
    return tmp

def et(w,s):
    tmp = 0.
    for i in range(len(s)):
       for j in range(len(s)):
          tmp += w[i,j] * s[i] * s[j]
    tmp = -0.5 * tmp
    return tmp
```

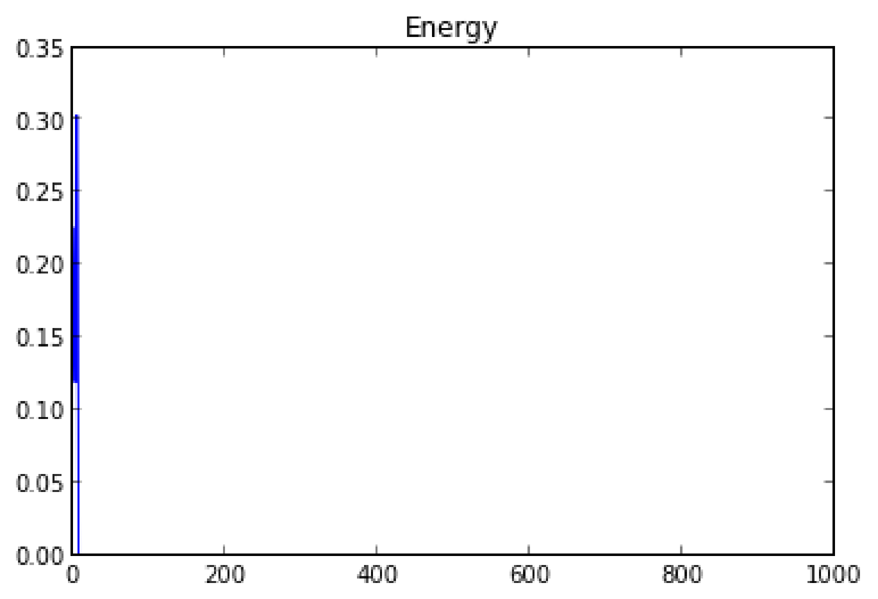
```
#Initialization
beta0 = 0.001
tao =1.01
tmax = 1000
random.seed(100)
N = 6
Et = [ 0. for i in range(tmax)]
beta = [0. for i in range(tmax)]
beta[0] = beta0

S = [0 for i in range (N)]
for i in range(N):
    S[i] = random.random()*2-1
    W = matrix([[0. for i in range(N)]for j in range (N)])
for i in range(N):
    for j in range(N):
       if(i==j): W[i,j] =0
       if(i<j): W[i,j] = random.random()
       if(i>j): W[i,j] = W[j,i]
print 'beta0:', beta0
print 'tao:',tao
print 'tmax:',tmax
print 'W:'
print W
print 'S', S
```

```
beta0: 0.001
tao: 1.01
tmax: 1000
W:
[[ 0.          0.80002046  0.53290141  0.08015371  0.45594588  0.04788752]
 [ 0.80002046  0.          0.9329624   0.94707801  0.33535078  0.30940593]
 [ 0.53290141  0.9329624   0.          0.76801815  0.20386953  0.17846076]
 [ 0.08015371  0.94707801  0.76801815  0.          0.18859491  0.34700445]
 [ 0.45594588  0.33535078  0.20386953  0.18859491  0.          0.62632164]
 [ 0.04788752  0.30940593  0.17846076  0.34700445  0.62632164  0.        ]]
S [-0.7086614897917394, -0.0901459909719573, 0.5415676113180443, 0.4110264538680559, 0.4639179460665115,
```

```
#optimization
for t in range(tmax):
    i = random.randint(0,5)
    ei = e(W,S,i)
    Et[t]= et(W,S)
    S[i] = math.tanh(beta[t]*ei)
    if((t+1)<tmax):
        beta[t+1] = tao * beta[t]
```

```
#plotting

Tt = [ 1/beta[t] for t in range (tmax)]

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(Tt)
ax.set_title("Temperature")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(Et)
ax.set_title("Energy")
plt.show()

print S
```





```
[-0.0, -0.0, -0.0, -0.0, -0.0, -0.0]
```