Exercise 1:

$x_0, x_1, x_2 \in \{0,1\}$

$$E(x_0,x_1,x_2) = \phi_0(x_0) + \phi_1(x_1) + \phi_2(x_2) + \phi_p(x_0,x_1) + \phi_p(x_1,x_2) + \phi_p(x_0,x_2)$$

a) Evaluate $E(x_0,x_1,x_2)$ by hand for all possible configurations of x_0,x_1,x_2

i) all down-state $(0,0,0)$

$$E(0,0,0) = 0{,}1 + 0{,}8 + 0{,}9 + 0 = 1{,}8$$

ii) all up-state $(1,1,1)$

$$E(1,1,1) = 0{,}9 + 0{,}1 + 0{,}1 + 0 = 1{,}1$$

iii) x_1 up-state, x_2, x_3 ~~different~~ arbitrary

$$E(1,0,1) = 0{,}9 + 0{,}8 + 0{,}1 + 2 = 3{,}8$$
$$E(1,1,0) = 0{,}9 + 0{,}1 + 0{,}9 + 2 = 3{,}9$$
$$E(1,0,0) = 0{,}9 + 0{,}8 + 0{,}9 + 2 = 4{,}6$$

iv) x_1 down state

$$E(0,0,1) = 0{,}1 + 0{,}8 + 0{,}9 + 2 = 3{,}8$$
$$E(0,1,0) = 0{,}1 + 0{,}1 + 0{,}9 + 2 = 3{,}1$$
$$E(0,1,1) = 0{,}1 + 0{,}1 + 0{,}1 + 2 = 2{,}3$$

b) Which configuration of x_0, x_1, x_2 minimizes $E(x_0,x_1,x_2)$?

$E(1,1,1)$ and $E(0,0,0)$ are the 2 lowest energies. They minimize $E_{pott} = \sum \phi_p$. Because the upper state is more likely on 2 particles $E(1,1,1)$ minimizes E.

Exercise 2:

$$\psi_{ij}(z_i, z_j) = \begin{cases} \alpha & \text{if } z_i = z_j \\ \beta & \text{if } z_i \neq z_j \end{cases}$$

$$E(z) = \sum_{i,j} \psi_{ij}(z_i, z_j)$$

Find the configuration which minimizes E for:

a) $\alpha = 0$, $\beta = 1$:

for minimal E, ψ needs to be 0.

ψ is 0 for $z_i = z_j$

↳ neighbors need to be in the same state

$E = 0$ for
$$\begin{array}{ccccc} \downarrow & -0- & \downarrow & -0- & \downarrow \\ \downarrow & & \downarrow & & \downarrow \\ \downarrow & & \downarrow & & \downarrow \end{array}$$

all particles in down state (0)

$E = 0$ for
$$\begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \uparrow & \uparrow & \uparrow \\ \uparrow & \uparrow & \uparrow \end{array}$$

all particles in up state (1)

b) $\alpha = 1$, $\beta = 0$:

$\psi = 0$ for $z_i \neq z_j$

↳ neighbors need to be in different states

configuration 1

$E = 0$ for
$$\begin{array}{cccccc} \uparrow & -0- & \downarrow & -0- & \uparrow & \downarrow \\ \downarrow & & \uparrow & & \downarrow & \uparrow \\ \uparrow & & \downarrow & & \uparrow & \downarrow \\ \downarrow & & \uparrow & & \downarrow & \uparrow \end{array}$$

configuration 2

BRUNNEN

Exercise 3: $\psi_{ij}(z_i, z_j) = \begin{cases} \alpha & \text{if } z_i = z_j \\ \beta & \text{if } z_i \neq z_j \end{cases}$

Is it possible to find z such that $E(z) = 0$ for ~~regar~~

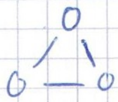
a) $\alpha = 0$, $\beta = 1$ regardless of the structure.
It is possible, because if all particles are in the same state ψ for any neighbor is 0.

z all up-state

or

z all down-state fullfills ~~minimizes~~ $E(z) = 0$

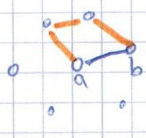
b) $\beta = 0$, $\alpha = 1$ regardless of structure.
This is not possible ~~e.g.~~ i.e. for the configuration of exercise 1



we can not achieve that all neighbors are in different states.

The „chain" which connects particle a to the neighbor b needs to be odd:

"chain": 3



Maybe the consequence is, that a particle needs to have an even number of neighbors

Untitled

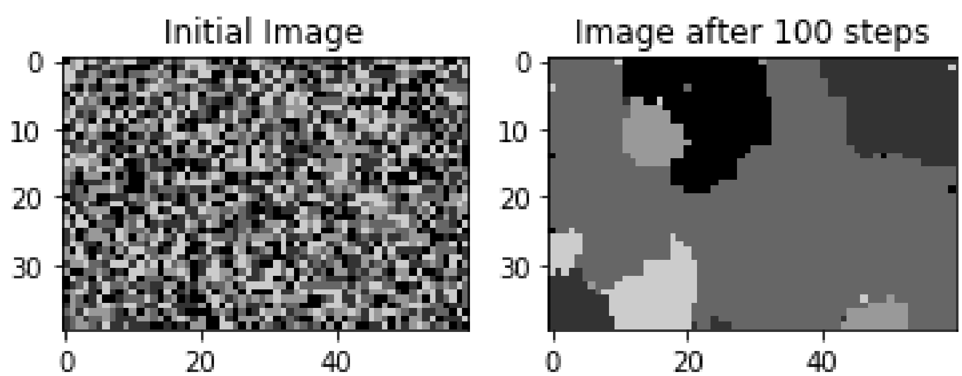
November 8, 2017

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import fabs as abs

def prior_l0(n_states, alpha):
    prior = np.ones((n_states, n_states))*alpha
    np.fill_diagonal(prior,0)
    return prior

def prior_l1(n_states, alpha):
    prior = np.zeros((n_states, n_states))
    for n in range(n_states):
        for m in range(n_states):
            prior[n,m] = alpha*abs(n-m)
    return prior

def calc_proba(lattice, x, y, prior):
    (xMax, yMax) = lattice.shape
    (horiz_neighbors, vertic_neighbors) = np.array([x-1,x+1]),np.array([y-1,y+1])
    states = []
    neighbors = []
    for xN in horiz_neighbors[np.in1d(horiz_neighbors,np.arange(xMax))]:
        neighbors.append(lattice[xN,y])
    for yN in vertic_neighbors[np.in1d(vertic_neighbors,np.arange(yMax))]:
        neighbors.append(lattice[x,yN])
    energies = np.exp(-np.sum(prior[:,neighbors], axis= 1))
    for i in range(len(prior)):
        states.append( energies[i]/np.sum(energies))
    return states

def gibbs_update(lattice, x, y, prior):
    probabilities = calc_proba(lattice, x, y, prior)
    new_state = np.random.choice(len(prior), 1, p=probabilities)
    lattice[x,y] = new_state

def sweep_scanlines(lattice, prior):
    (x_len,y_len) = lattice.shape
    for x in range(x_len):
```

```
            for y in range(y_len):
                gibbs_update(lattice, x, y, prior)
        return 0
```

```
In [2]: print('Prior 1, 100 steps, alpha = 2')
        n_iter = 100
        n_states = 5
        n_x = 40
        n_y = 60
        prior= prior_l0(n_states,2)
        lattice = np.random.randint(n_states, size = (n_x, n_y))
        plt.figure()
        plt.subplot(121)
        plt.imshow(lattice, cmap='gray', vmax=n_states)
        plt.title('Initial Image')
        for i in range(n_iter):
            sweep_scanlines(lattice, prior)

        plt.subplot(122)
        plt.imshow(lattice, cmap='gray', vmax=n_states)
        plt.title('Image after %d steps'%n_iter)
        plt.show()
```

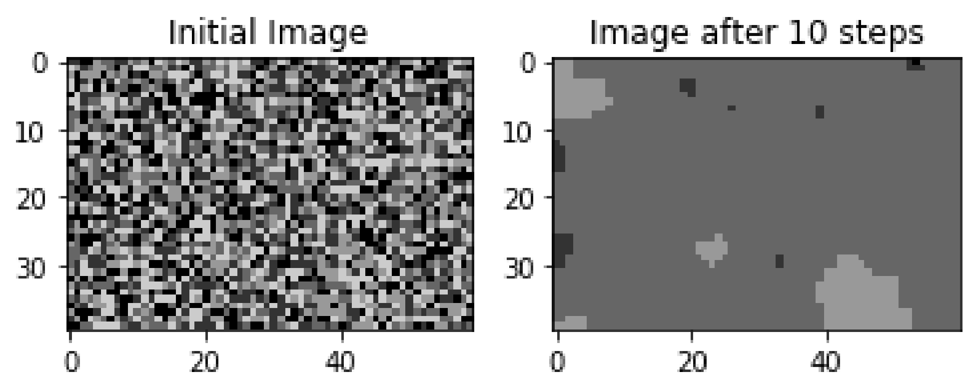
```
Prior 1, 100 steps, alpha = 2
```




```
In [3]: print('Prior 1, 10 steps, alpha = 2')
        n_iter = 10
        n_states = 5
        n_x = 40
        n_y = 60
        prior= prior_l1(n_states,2)
        lattice = np.random.randint(n_states, size = (n_x, n_y))
        plt.figure()
```

```
plt.subplot(121)
plt.imshow(lattice, cmap='gray', vmax=n_states)
plt.title('Initial Image')
for i in range(n_iter):
    sweep_scanlines(lattice, prior)

plt.subplot(122)
plt.imshow(lattice, cmap='gray', vmax=n_states)
plt.title('Image after %d steps'%n_iter)
plt.show()
```

```
Prior 1, 10 steps, alpha = 2
```



Prior 2 converges much faster