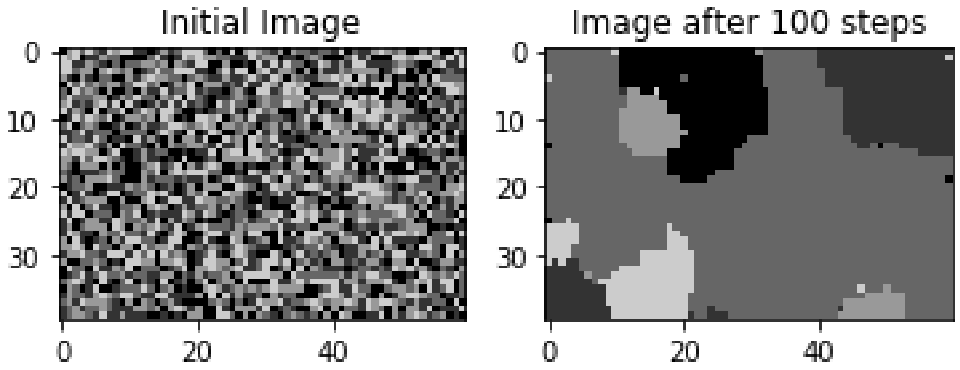
# Untitled

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
```

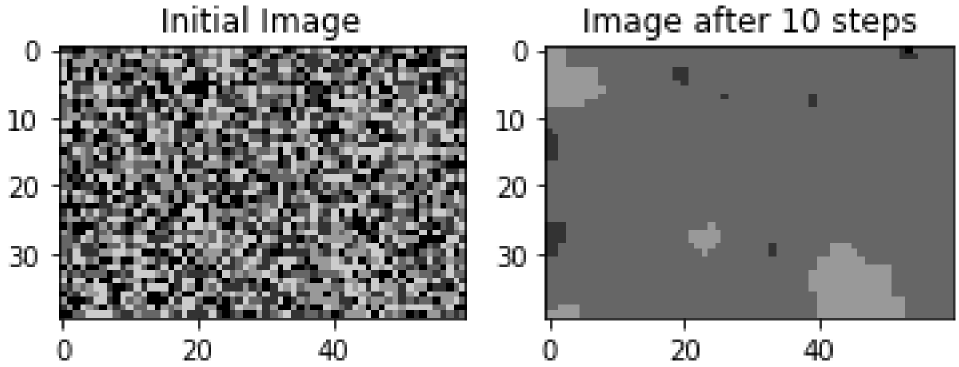

Initial Image

Image after 10 steps

Prior 2 converges much faster