In [1]:
```
import sys
sys.path.insert(0, '../')
```

In [2]:
```
from minisom_new import MiniSom

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
```

In [3]:
```
%matplotlib inline
```

In [4]:
```
data = np.genfromtxt('iris.csv', delimiter=',', usecols=(0, 1, 2, 3))
data
```

```
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
```

In [5]:
```
# data normalization
data = np.apply_along_axis(lambda x: x/np.linalg.norm(x), 1, data)
data
```

Out[5]:
```
array([[0.80377277, 0.55160877, 0.22064351, 0.0315205 ],
       [0.82813287, 0.50702013, 0.23660939, 0.03380134],
       [0.80533308, 0.54831188, 0.2227517 , 0.03426949],
       [0.80003025, 0.53915082, 0.26087943, 0.03478392],
       [0.790965  , 0.5694948 , 0.2214702 , 0.0316386 ],
       [0.78417499, 0.5663486 , 0.2468699 , 0.05808704],
       [0.78010936, 0.57660257, 0.23742459, 0.0508767 ],
       [0.80218492, 0.54548574, 0.24065548, 0.0320874 ],
       [0.80642366, 0.5315065 , 0.25658935, 0.03665562],
       [0.81803119, 0.51752994, 0.25041771, 0.01669451],
       [0.80373519, 0.55070744, 0.22325977, 0.02976797],
       [0.786991  , 0.55745196, 0.26233033, 0.03279129],
       [0.82307218, 0.51442011, 0.24006272, 0.01714734],
       [0.8025126 , 0.55989251, 0.20529392, 0.01866308],
       [0.81120865, 0.55945424, 0.16783627, 0.02797271],
       [0.77381111, 0.59732787, 0.2036345 , 0.05430253],
       [0.79428944, 0.57365349, 0.19121783, 0.05883625],
       [0.80327412, 0.55126656, 0.22050662, 0.04725142],
       [0.8068282 , 0.53788547, 0.24063297, 0.04246464],
```

In [6]:
```
# Initialization and training
som = MiniSom(7, 7, 4, sigma=3, learning_rate=0.5, neighborhood_function='triang
#som = MiniSom(x= 7, y = 7, input_len= 4, sigma=3, learning_rate=0.5, random_see
#som.random_weights_init(data)
som.pca_weights_init(data)
print("Training...")
som.train_random(data, 4000)  # random training
print("\n...ready!")
```

```
Training...

...ready!
```

In [7]:
```
som
```

Out[7]:
```
<minisom_new.MiniSom at 0x28c18430128>
```

In [8]:
```
#som.random_weights_init(data)
som.pca_weights_init(data)
print("Training...")
som.train_random(data, 4000)  # random training
print("\n...ready!")
```

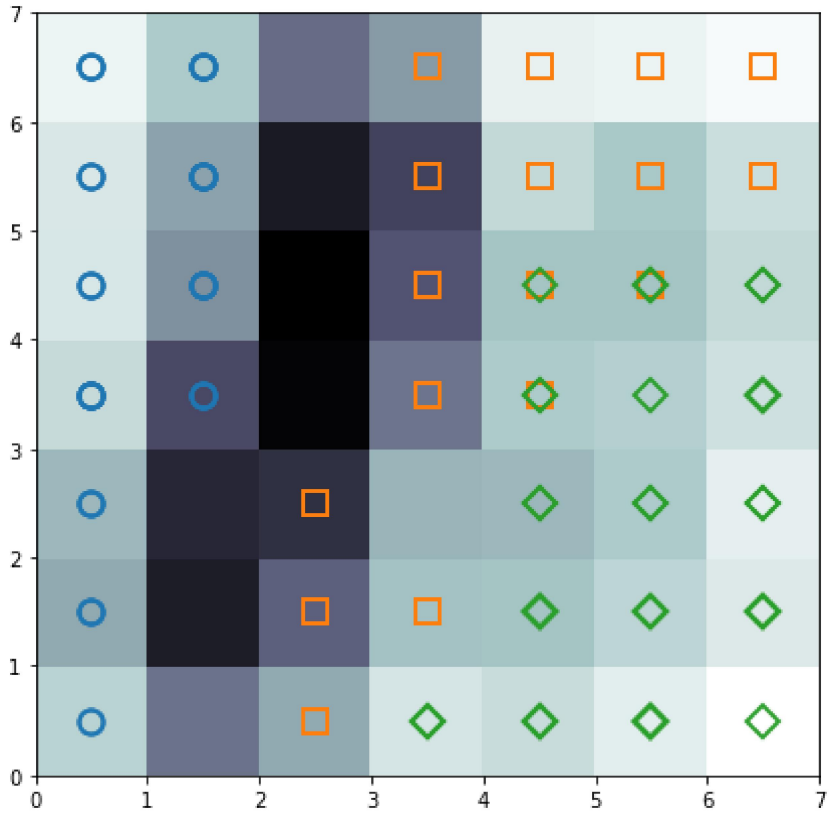
```
Training...

...ready!
```

In [9]:
```
plt.figure(figsize=(7, 7))
# Plotting the response for each pattern in the iris dataset
plt.pcolor(som.distance_map().T, cmap='bone_r')  # plotting the distance map as l
#plt.colorbar()

target = np.genfromtxt('iris.csv', delimiter=',', usecols=(4), dtype=str)
t = np.zeros(len(target), dtype=int)
t[target == 'setosa'] = 0
t[target == 'versicolor'] = 1
t[target == 'virginica'] = 2

# use different colors and markers for each label
markers = ['o', 's', 'D']
colors = ['C0', 'C1', 'C2']
for cnt, xx in enumerate(data):
    w = som.winner(xx)  # getting the winner
    # palce a marker on the winning position for the sample xx
    plt.plot(w[0]+.5, w[1]+.5, markers[t[cnt]], markerfacecolor='None',
             markeredgecolor=colors[t[cnt]], markersize=12, markeredgewidth=2)
plt.axis([0, 7, 0, 7])
plt.savefig('som_iris.png')
plt.show()
```



In [11]:
```
labels_maps = som.labels_map(data, t)
"""Returns a dictionary wm where wm[(i,j)] is a dictionary
       that contains the number of samples from a given label
       that have been mapped in position i,j.

       Parameters
       ----------
       data : data matrix

       label : list or array that contains the label of each sample in data.
       """
```

Out[11]:
```
'Returns a dictionary wm where wm[(i,j)] is a dictionary\n       that contains
the number of samples from a given label\n       that have been mapped in posi
tion i,j.\n\n       Parameters\n       ----------\n       data : data matrix
\n\n       label : list or array that contains the label of each sample in dat
a.\n       '
```

In [12]:
```
labels_maps
```

Out[12]:
```
defaultdict(list,
            {(1, 5): Counter({0: 7}),
             (0, 3): Counter({0: 7}),
             (1, 3): Counter({0: 5}),
             (1, 6): Counter({0: 6}),
             (1, 4): Counter({0: 7}),
             (0, 5): Counter({0: 3}),
             (0, 6): Counter({0: 4}),
             (0, 1): Counter({0: 3}),
             (0, 0): Counter({0: 2}),
             (0, 4): Counter({0: 3}),
             (0, 2): Counter({0: 3}),
             (3, 5): Counter({1: 7}),
             (2, 2): Counter({1: 6}),
             (4, 5): Counter({1: 4}),
             (5, 5): Counter({1: 3}),
             (4, 4): Counter({1: 4, 2: 1}),
             (2, 1): Counter({1: 2}),
             (3, 4): Counter({1: 1}),
             (5, 6): Counter({1: 3}),
             (2, 0): Counter({1: 1}),
             (6, 6): Counter({1: 4}),
             (3, 6): Counter({1: 3}),
             (3, 1): Counter({1: 3}),
             (4, 6): Counter({1: 2}),
             (6, 5): Counter({1: 2}),
             (4, 3): Counter({1: 1, 2: 2}),
             (3, 3): Counter({1: 3}),
             (5, 4): Counter({1: 1, 2: 3}),
             (4, 0): Counter({2: 4}),
             (5, 0): Counter({2: 7}),
             (6, 2): Counter({2: 2}),
             (6, 0): Counter({2: 1}),
             (6, 3): Counter({2: 5}),
             (6, 4): Counter({2: 2}),
             (4, 1): Counter({2: 6}),
             (3, 0): Counter({2: 3}),
             (5, 1): Counter({2: 4}),
             (6, 1): Counter({2: 5}),
             (5, 3): Counter({2: 1}),
             (5, 2): Counter({2: 2}),
             (4, 2): Counter({2: 2})})
```

In [ ]:
```

```