```
import numpy as np


proto_nodes = np.array([
                        [0,0],
                        [0.5,1],
                        [1,0.5]
])

def manhattan_distance(x,y):
  return np.absolute(x[0] - y[0]) + np.absolute(x[1] - y[1])

def rb_function(x, p_node):
  return max(0, 1 - manhattan_distance(x, p_node))

datapoint = np.array([0.6,0.8])
o_nodes = 3
weights = np.zeros([proto_nodes.shape[0], o_nodes])
weights[:,0] = 0.6
weights[:,1] = -0.4
weights[:,2] = 0
weights
```

```
    array([[ 0.6, -0.4,  0. ],
           [ 0.6, -0.4,  0. ],
           [ 0.6, -0.4,  0. ]])
```

```
rbf_results = []
for n in proto_nodes:
  rbf_results.append(rb_function(datapoint, n))

rbf_results = np.array(rbf_results)
outputs = np.dot(rbf_results, weights)
outputs
```

```
    array([ 0.6, -0.4,  0. ])
```

Class A is the winner!

```
target = np.array([0,1,0])
lr = 0.1
weight_changes = np.zeros_like(weights)
for i, a in enumerate(rbf_results):
  weight_changes[i] = lr*(target - outputs)*a
weight_changes
```

```
array([[-0.   ,  0.   ,  0.   ],
       [-0.042,  0.098,  0.   ],
       [-0.018,  0.042,  0.   ]])
```

```
new_weights = weights+weight_changes
new_weights
```

```
array([[ 0.6  , -0.4  ,  0.   ],
       [ 0.558, -0.302,  0.   ],
       [ 0.582, -0.358,  0.   ]])
```