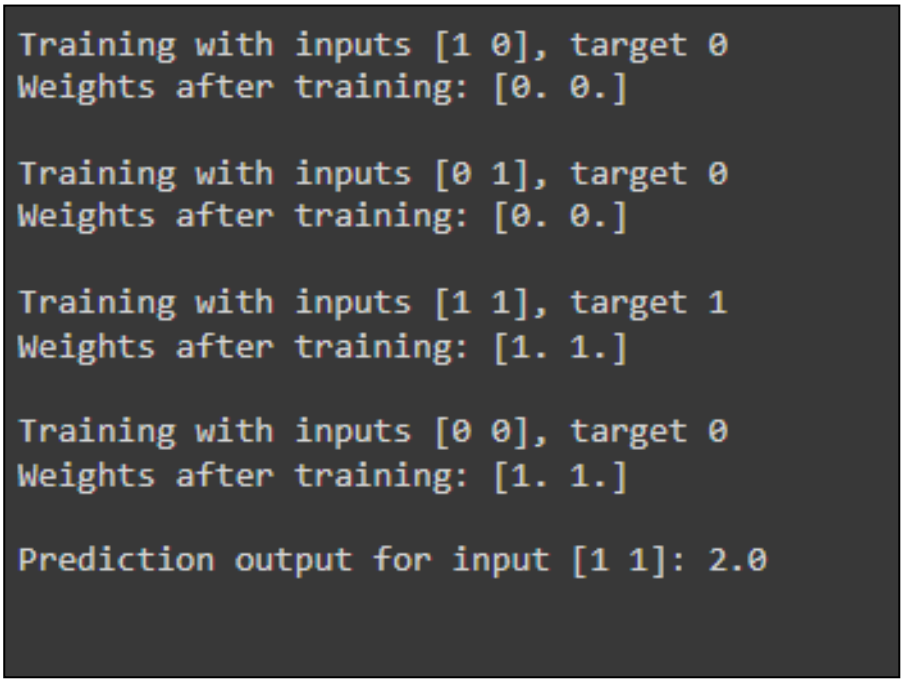
Code:

```
import numpy as np
class HebbianNetwork:
    def __init__(self, input_size):
        self.weights = np.zeros(input_size)
    def predict(self, inputs):
        return np.dot(inputs, self.weights)
    def train(self, inputs, target):
        self.weights += inputs * target
if __name__ == "__main__":
    network = HebbianNetwork(input_size=2)
    training_data = [
        (np.array([1, 0]), 0),
        (np.array([0, 1]), 0),
        (np.array([1, 1]), 1),
        (np.array([0, 0]), 0) ]
    for inputs, target in training_data:
        print(f"Training with inputs {inputs}, target {target}")
        network.train(inputs, target)
        print(f"Weights after training: {network.weights}\n")
    test_input = np.array([1, 1])
    test_input = np.array([0, 0])
    output = network.predict(test_input)
    print(f"Predicted Output for input {test_input} is {output}")
```

Output:



Code:

```
clc;
clear all;
close all;
M=readmatrix('/MATLAB Drive/iris7.csv')
x=M(:,1:4);
t=M(:,5);
net=perceptron;
net.trainParam.epochs=50;
net=train(net,x',t');
out=sim(net,x');
Y=round(out);
[C,O]=confusionmat(t',Y);
[m,n]=size(C);
S=0;
for i=1:1:m
   for j=1:1:n
      S=S+C(i,j);
   end
end
D=sum(diag(C));
Acc=(D*100)/S;
Acc
```

Output:

```
Acc =

    4.6667
```

Code:

```
clc;
clear all;
close all;
M=readmatrix('/MATLAB Drive/iris7.csv')
x=M(:,1:4);
t=M(:,5);
net=feedforwardnet(10);
net=train(net,x',t');
out=sim(net,x');
Y=round(out);
[C,O]=confusionmat(t',Y);
[m,n]=size(C);
S=0;
for i=1:1:m
  for j=1:1:n
     S=S+C(i,j);
  end
end
D=sum(diag(C));
Acc=(D/S)*100;
Acc
```

Output:

```
Acc =

    8.6667
```

Code:

```
clc;
clear all;
close all;
M=readmatrix('/MATLAB Drive/iris7.csv');
x=M(:,1:4);
t=M(:,5);
t=grp2idx(categorical(t));
T=ind2vec(t');
net=newrbe(x',T,1.2);
out=sim(net,x');
Y=vec2ind(out);
[C,O]=confusionmat(t',Y);
[m,n]=size(C);
S=0;
for i=1:1:m
  for j=1:1:n
     S=S+C(i,j);
  end
end
D=sum(diag(C));
Acc=(D/S)*100;
Acc
```

Output:

```
Acc =

   100
```

Code:

```
clc;
clear all;
close all;
M = readmatrix("/MATLAB Drive/iris2.csv");
x = M(:,1:4);
t = M(:,5);
net = lvqnet(10,0.9);
net.trainParam.epochs=50;
net = train(net,x',t');
out = sim(net,x');
[C,O] = confusionmat(t',out);
%disp(C);
[m,n] = size(C);
S = 0;
for i=1:1:m
  for j=1:1:n
     S = S+C(i,j);
  end
end
D = sum(diag(C));
Acc = (D/S)*100;
Acc
```

Output:

```
Acc =

   40.5405

>>
```

Code:

```
clc;
clear all;
close all;
M = readmatrix("/MATLAB Drive/Iris.csv");
x = M(:,1:4);
net = competlayer(4);
net.trainParam.epochs=100;
net = train(net,x');
output = net(x');
Y = vec2ind(output);
Y
```

Output:

Command Window

```
output =

  Columns 1 through 22

    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0
    1    0    0    0    1    1    0    1    0    0    1    1    0    0    1    0    0    1    0    0    1    0
    0    1    1    1    0    0    1    0    1    1    0    0    1    1    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0    1    1    0    1

  Columns 23 through 37

    0    0    0    1    0    1    1    0    1    1    1    1    1    1    1
    1    0    1    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    1    0    0    1    0    0    1    0    0    0    0    0    0    0


Y =

  Columns 1 through 22

    2    3    3    3    2    2    3    2    3    3    2    2    3    3    2    1    4    2    4    4    2    4

  Columns 23 through 37

    2    4    2    1    4    1    1    4    1    1    1    1    1    1    1
```

Code:

```
clc;
clear all;
close all;
M=dlmread('/MATLAB Drive/iris2.csv');
x=M(:,1:4);
t=M(:,5);
T=ind2vec(t');
net=newpnn(x',T,1.2); out=sim(net,x');
Y=vec2ind(out);
[C,O]=confusionmat(t',Y);
[m,n]=size(C);
S=0;
for i=1:1:m
for j=1:1:n
   S=S+C(i,j);
end
end
D=sum(diag(C));
acc=(D/S)*100;
acc
```

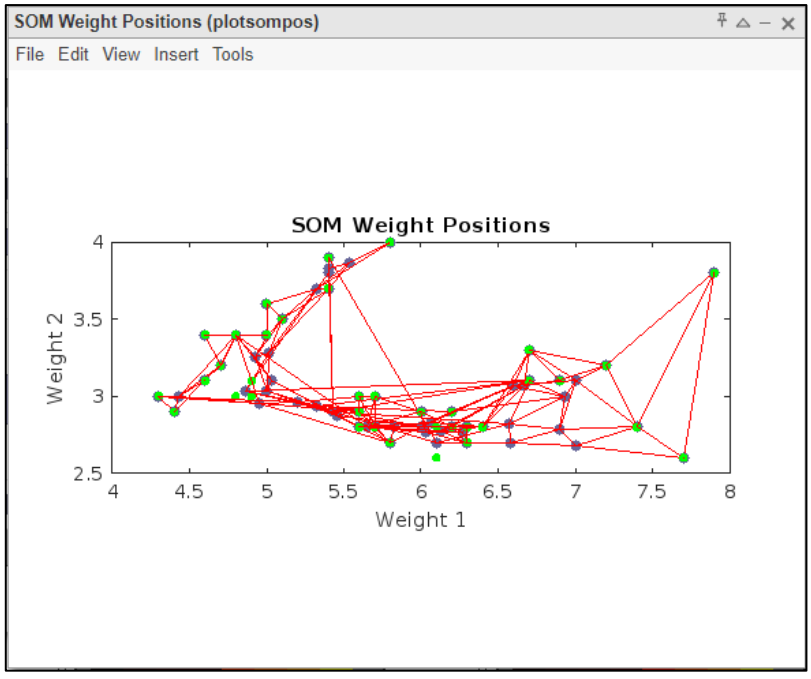
Output:

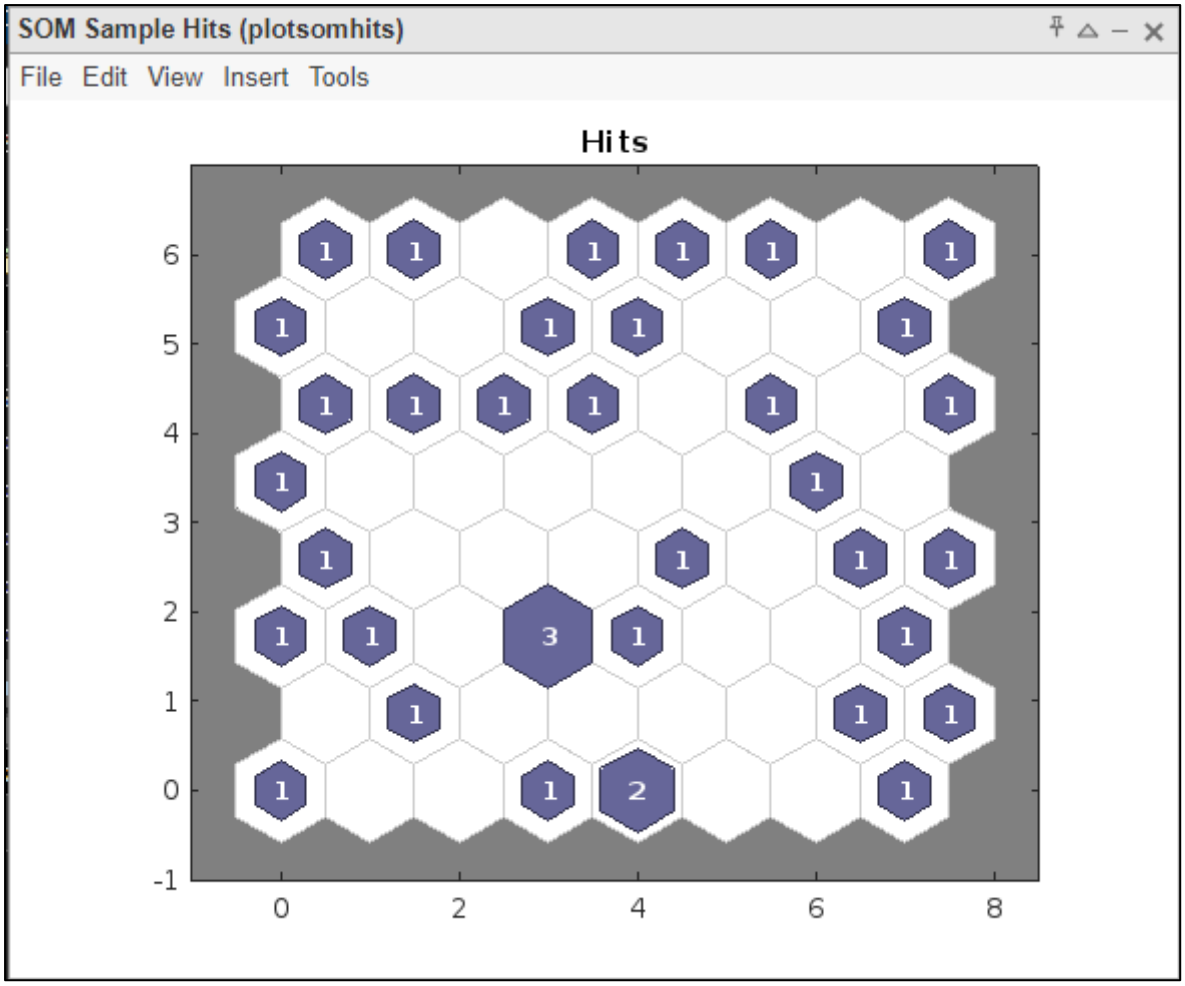
```
acc =

   91.8919
```

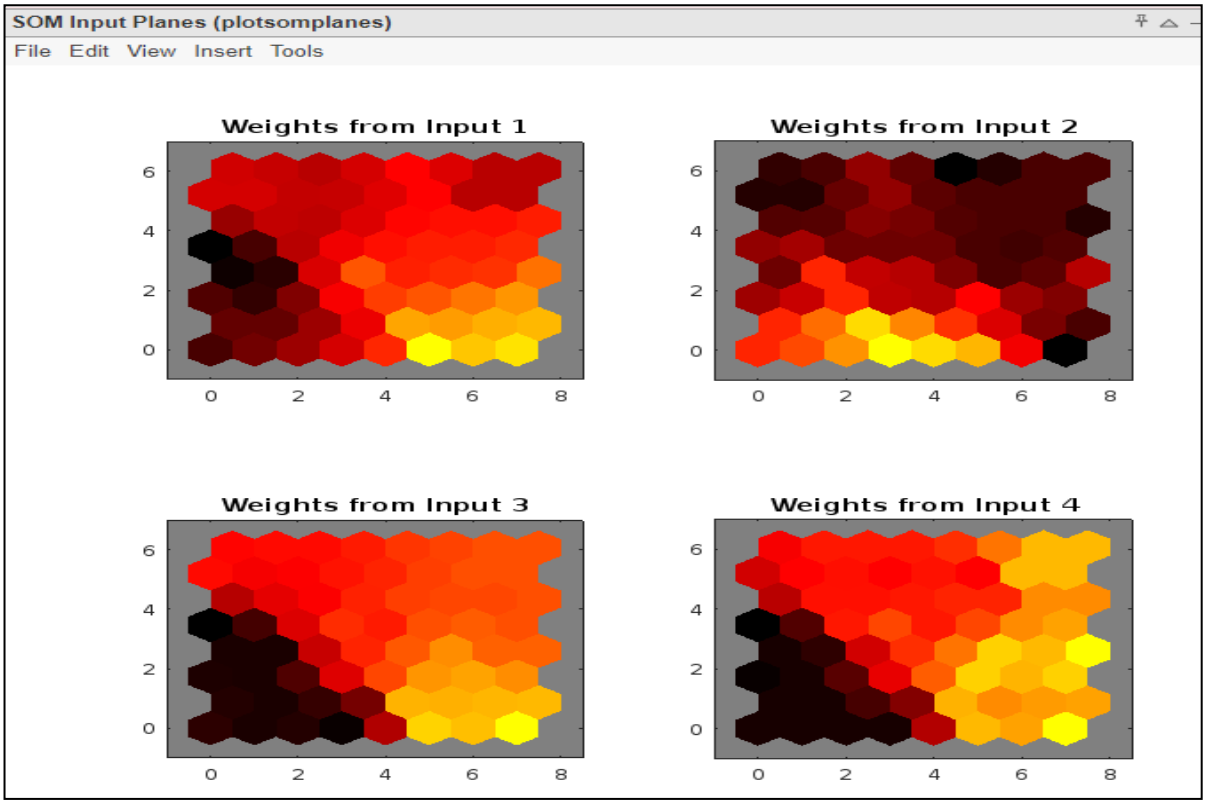
Code:

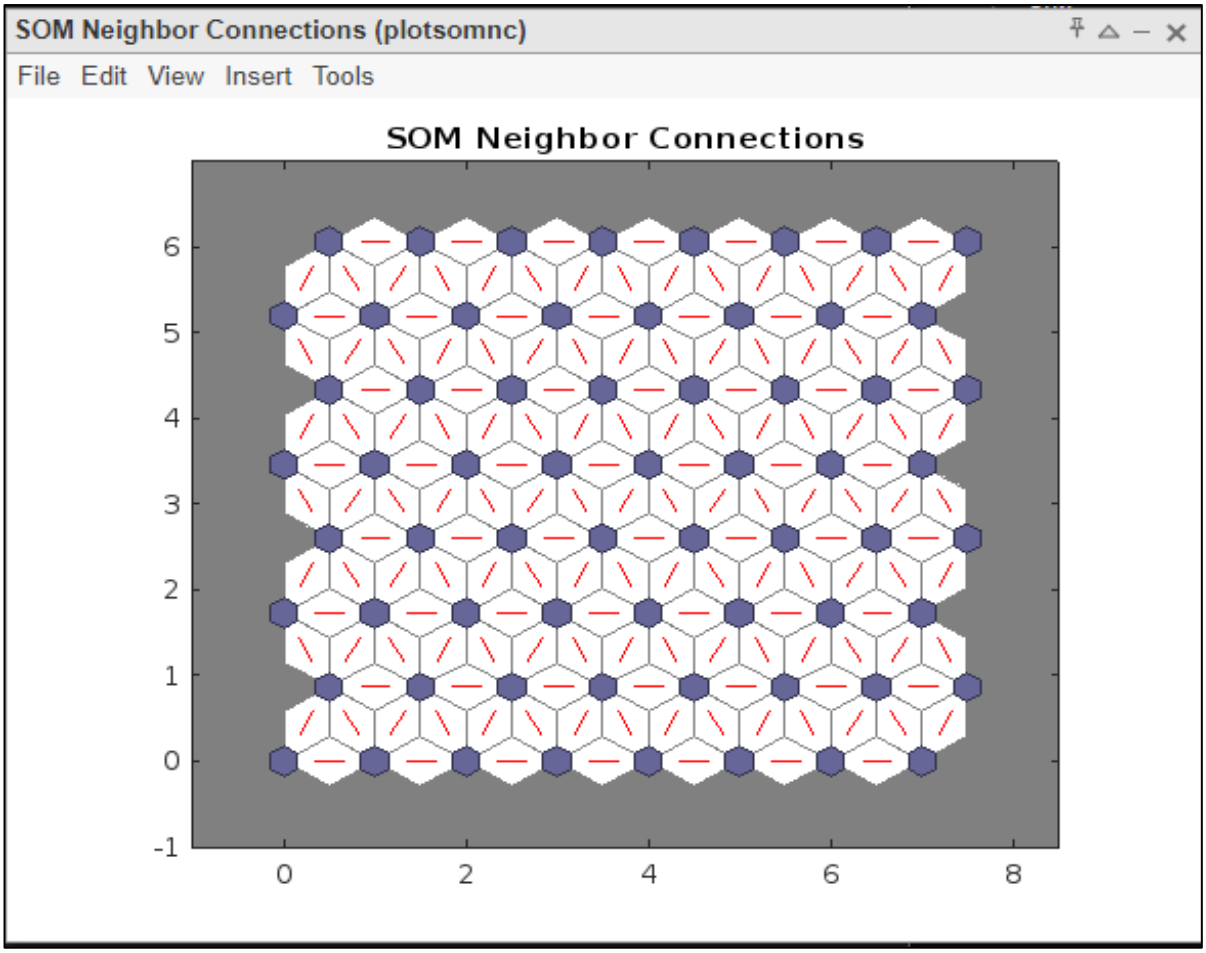
```
clc;
clear all;
close all;
M = readmatrix("/MATLAB Drive/iris2.csv");
x = M(:,1:4);
net = selforgmap([8,8]);
net = train(net,x');
view(net);
y = net(x');
Y = vec2ind(y);
%Y
figure(1),plotsomnc(net);
figure(2),plotsomnd(net);
figure(3),plotsomplanes(net);
figure(4),plotsomhits(net,x');
figure(5),plotsompos(net,x');
```

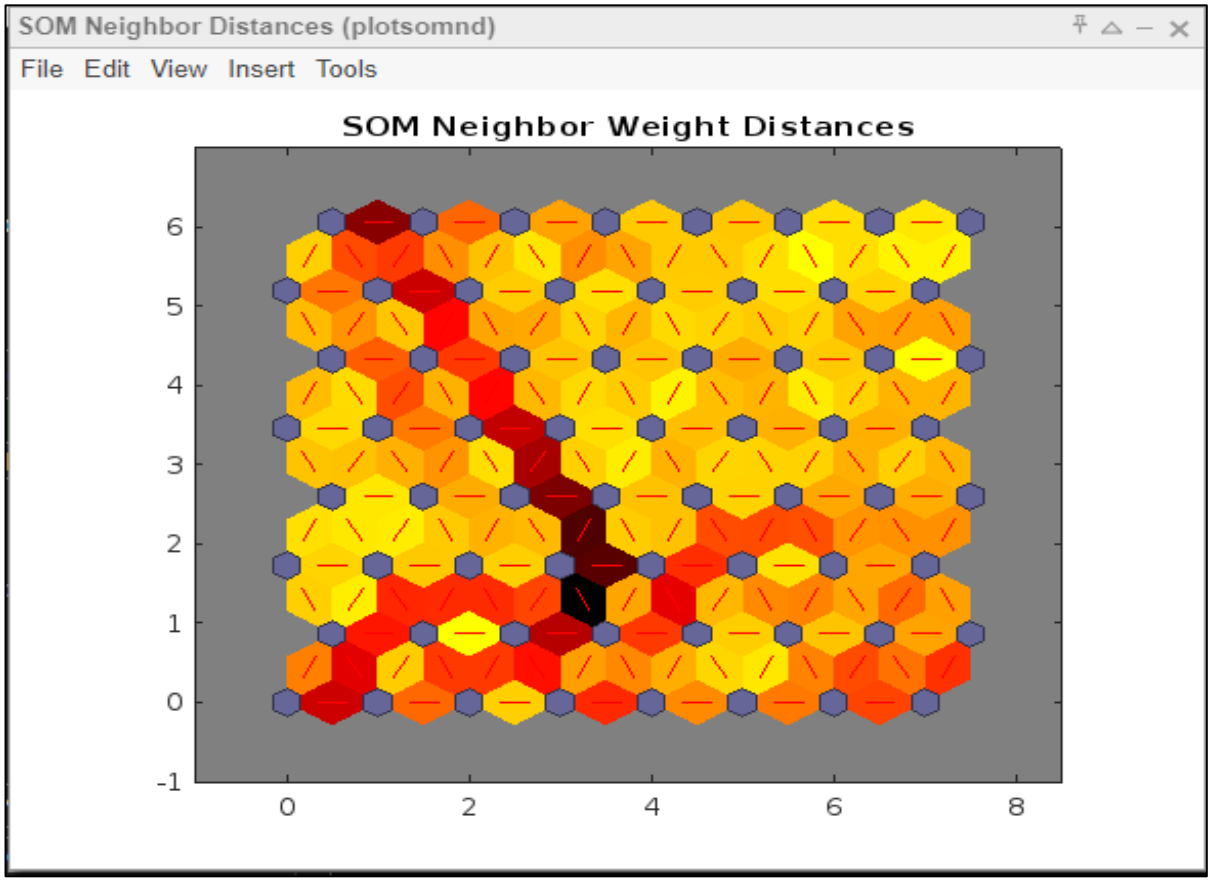
Output:

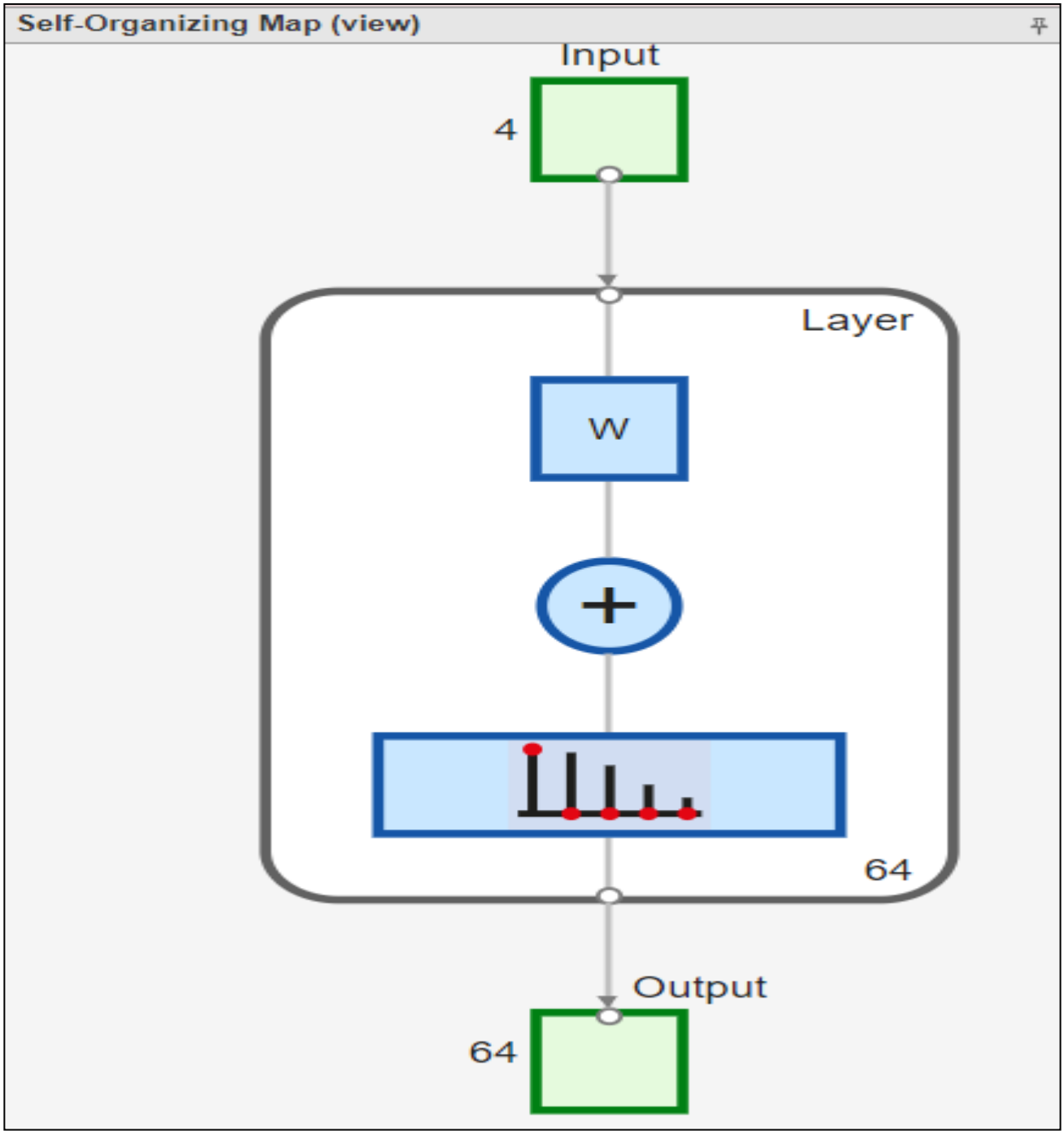



SOM Sample Hits (plotsomhits)
File Edit View Insert Tools
Hits
6
5
4
3
2
1
0
-1
0
2
4
6
8


SOM Input Planes (plotsomplanes)
File Edit View Insert Tools
Weights from Input 1
Weights from Input 2
Weights from Input 3
Weights from Input 4


SOM Neighbor Connections (plotsomnc)
File Edit View Insert Tools
SOM Neighbor Connections
6
5
4
3
2
1
0
-1
0
2
4
6
8


SOM Neighbor Distances (plotsomnd)
File Edit View Insert Tools
SOM Neighbor Weight Distances
6
5
4
3
2
1
0
-1
0
2
4
6
8


Self-Organizing Map (view)
Input
4
Layer
W
+
64
Output
64

Code:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
# Define fuzzy variables
# Input variables
TestPerformance = ctrl.Antecedent(np.arange(0, 11, 1), 'TestPerformance')
ClassPerformance = ctrl.Antecedent(np.arange(0, 11, 1), 'ClassPerformance')
# Output variable
OverallPerformance = ctrl.Consequent(np.arange(0, 11, 1), 'OverallPerformance')
# Define membership functions for TestPerformance
TestPerformance['Poor'] = fuzz.gaussmf(TestPerformance.universe, 0, 1.5)
TestPerformance['Good'] = fuzz.gaussmf(TestPerformance.universe, 5, 1.5)
TestPerformance['Excellent'] = fuzz.gaussmf(TestPerformance.universe, 10, 1.5)
# Define membership functions for ClassPerformance
ClassPerformance['Bad'] = fuzz.gaussmf(ClassPerformance.universe, 0, 1.5)
ClassPerformance['Decent'] = fuzz.gaussmf(ClassPerformance.universe, 5, 1.5)
ClassPerformance['Great'] = fuzz.gaussmf(ClassPerformance.universe, 10, 1.5)
# Define membership functions for OverallPerformance
OverallPerformance['Low'] = fuzz.gaussmf(OverallPerformance.universe, 0, 1.5)
OverallPerformance['Medium'] = fuzz.gaussmf(OverallPerformance.universe, 5,
1.5)
OverallPerformance['High'] = fuzz.gaussmf(OverallPerformance.universe, 10, 1.5)
# Define fuzzy rules
rule1 = ctrl.Rule(TestPerformance['Excellent'] & ClassPerformance['Great'],
OverallPerformance['High'])
rule2 = ctrl.Rule(TestPerformance['Good'] & ClassPerformance['Decent'],
OverallPerformance['Medium'])
rule3 = ctrl.Rule(TestPerformance['Poor'] & ClassPerformance['Bad'],
OverallPerformance['Low'])
# Create a control system and simulation
performance_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
performance_sim = ctrl.ControlSystemSimulation(performance_ctrl)
# Get user inputs
A = float(input("Enter your Test Performance rate (0 to 10): "))
C = float(input("Enter your Class Performance rate (0 to 10): "))

# Pass inputs to the simulation
performance_sim.input['TestPerformance'] = A
performance_sim.input['ClassPerformance'] = C
```

```
# Compute the result
performance_sim.compute()
# Display the result
result = performance_sim.output['OverallPerformance']
print(f"Overall Performance: {result:.2f}")
```

Output: