Self Organizing Map

Graduate Program in Software
SEIS 763: Machine Learning
Dr. Chih Lai

SOM Links

- SOM plot explains
  - http://www.mathworks.com/help/nnet/gs/cluster-data-with-a-self-organizing-map.html

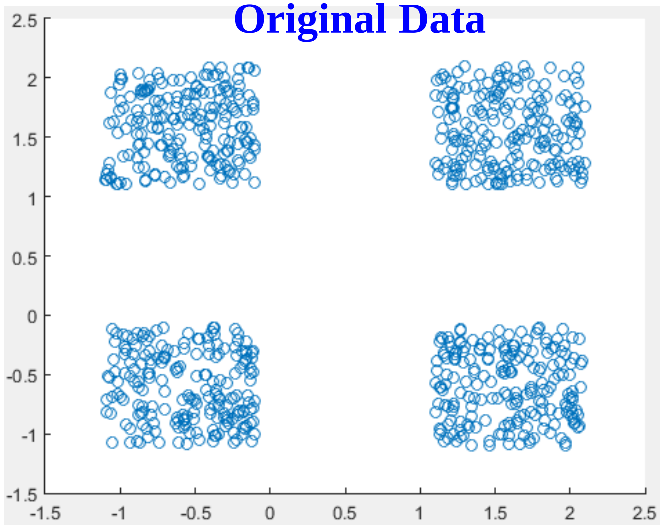
- Cluster with Self-Organizing Map Neural Network (theory & how it works)
  - http://www.mathworks.com/help/nnet/ug/cluster-with-self-organizing-map-neural-network.html

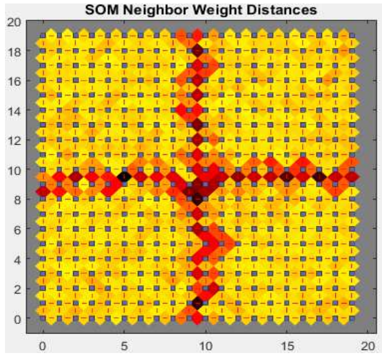
- selforgmap()
  - http://www.mathworks.com/help/nnet/ref/selforgmap.html

- learnsomb() ➔ Batch self-organizing map weight learning function
  - http://www.mathworks.com/help/nnet/ref/learnsomb.html

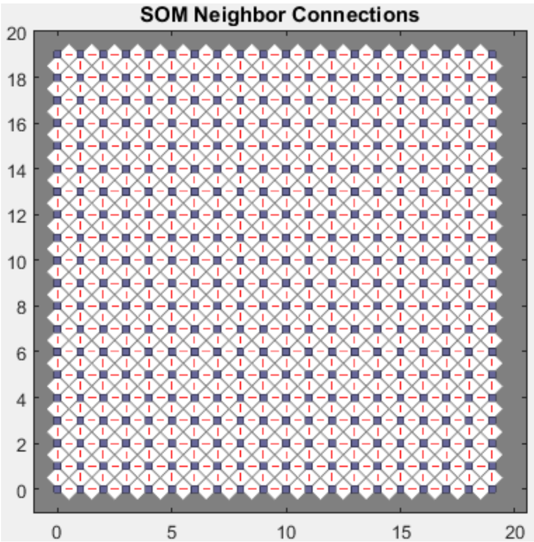
Self Organizing Map (SOM)

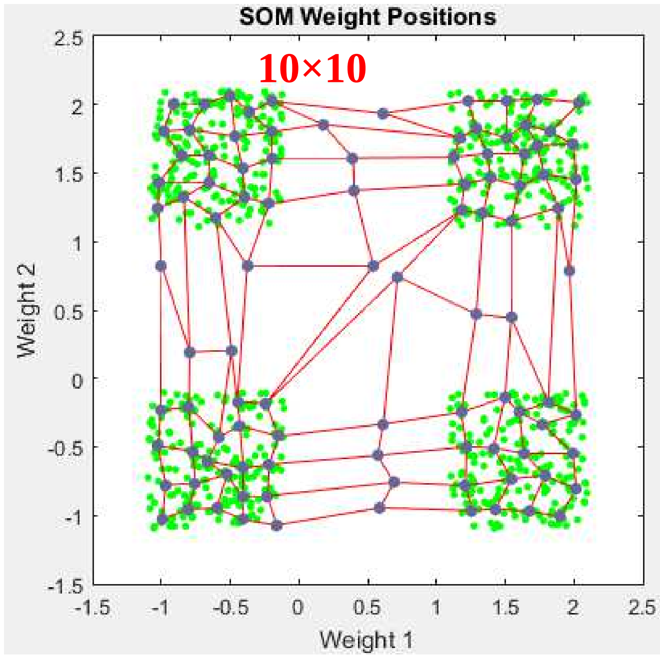
- Find clusters using SOM.
  - Neurons clustered into groups.

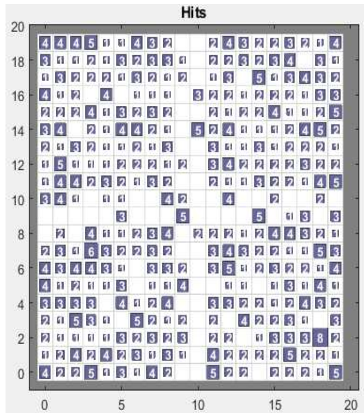




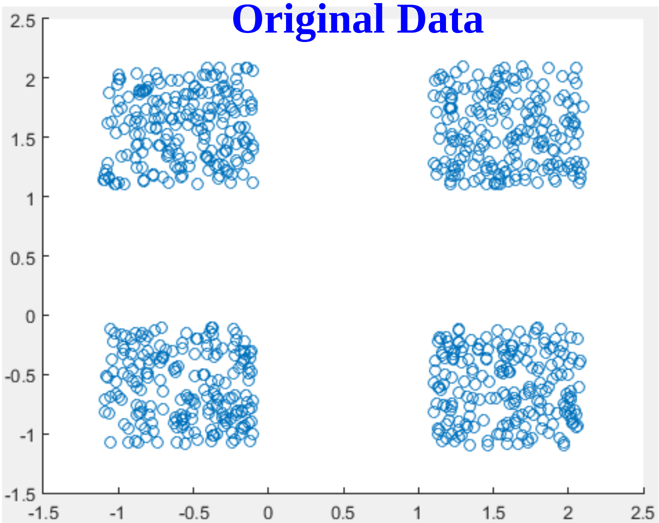
Grid of *x* × *x* neurons

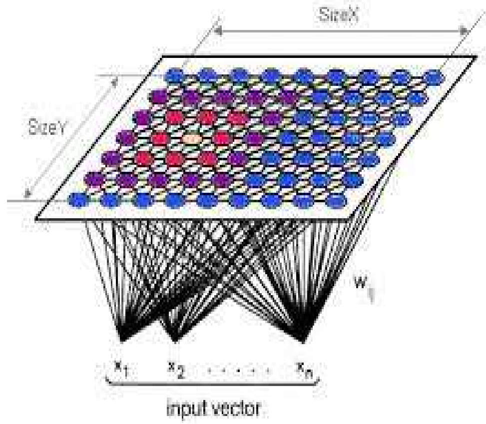




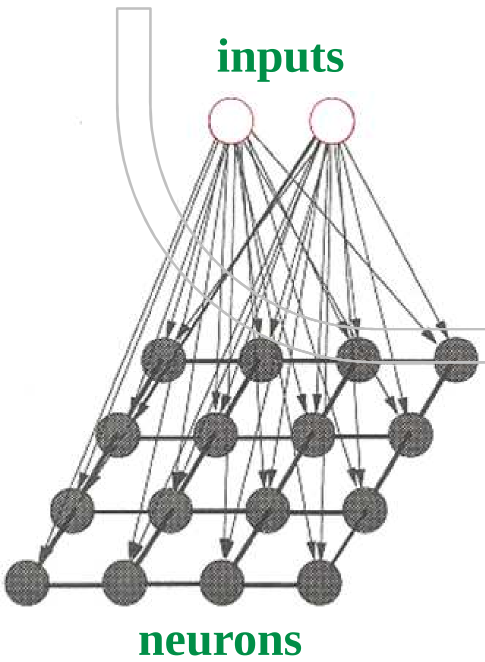


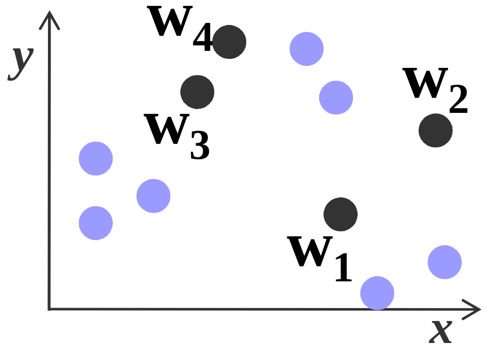
SOM, Another View

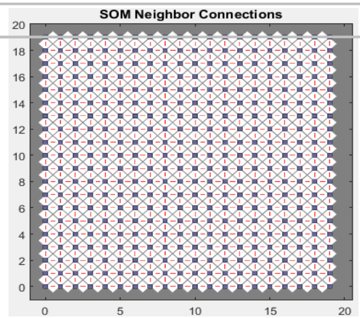




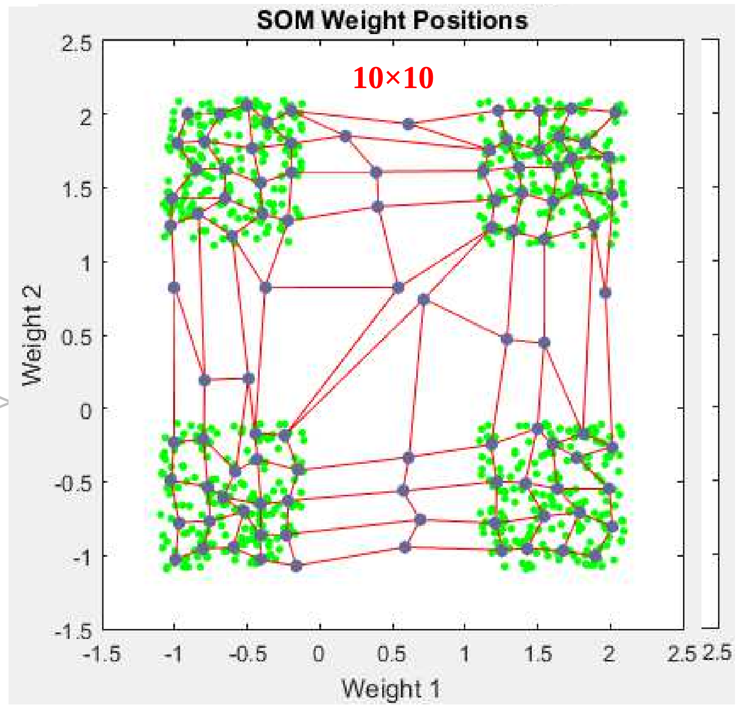
http://www.pitt.edu/~is2470pb/Spring05/FinalProjects/Group1a/tutorial/som.html





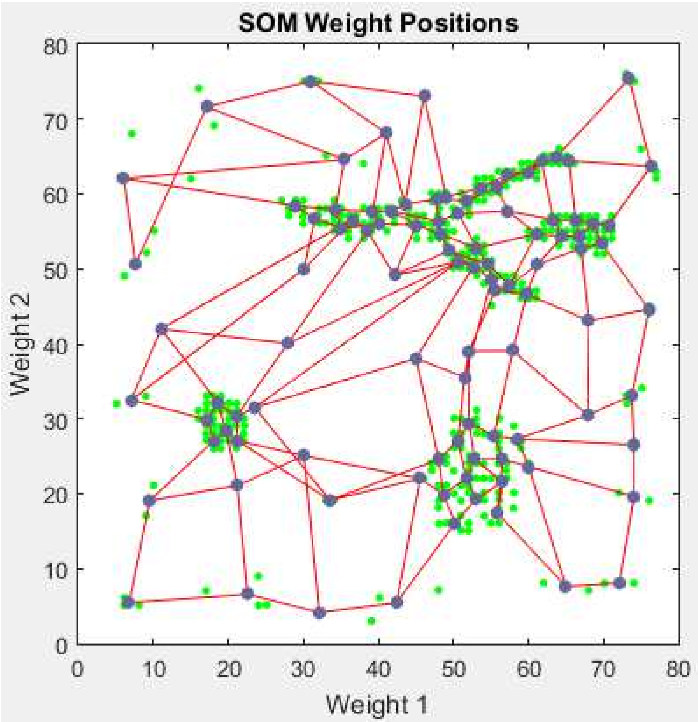


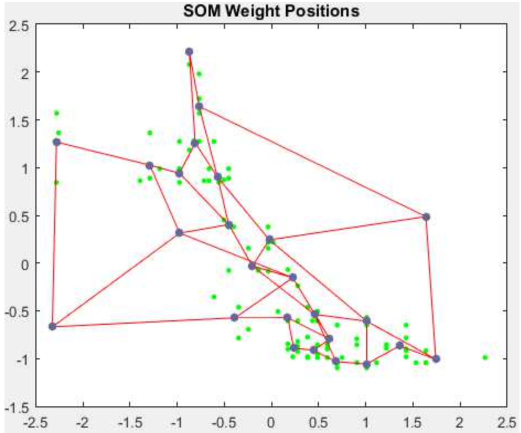
Grid of 20×20 neurons



Another SOM Application

- Summarizing data before further processing,   like classification.





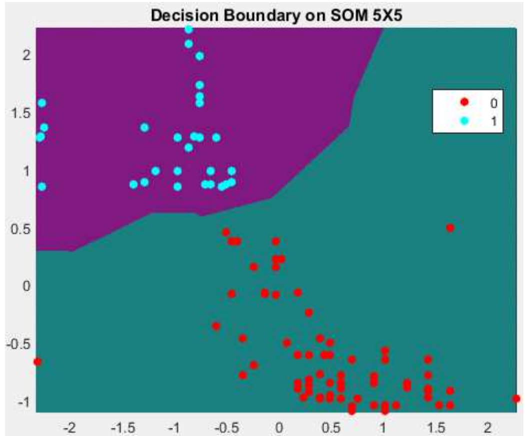
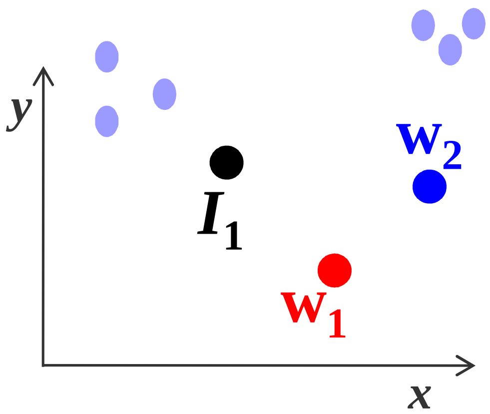
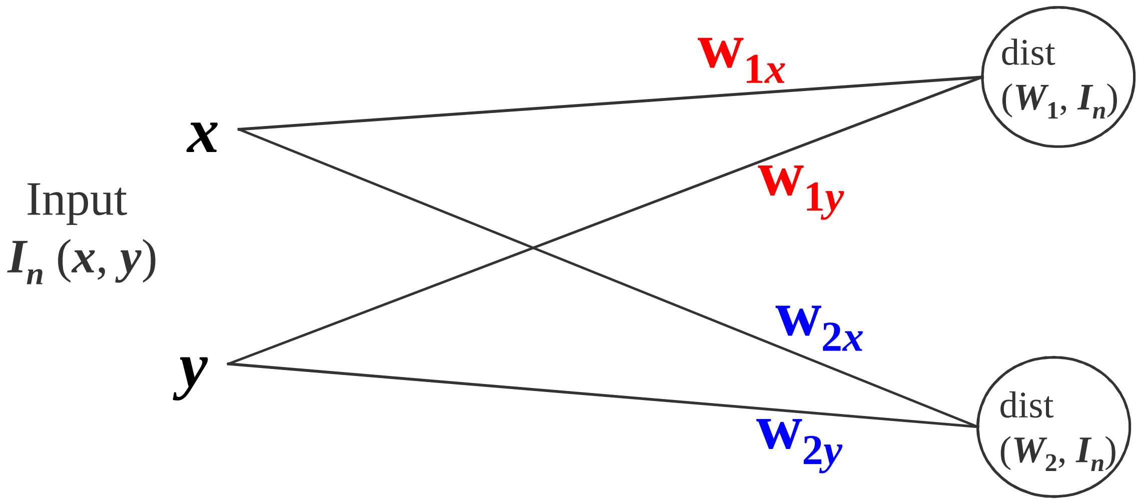


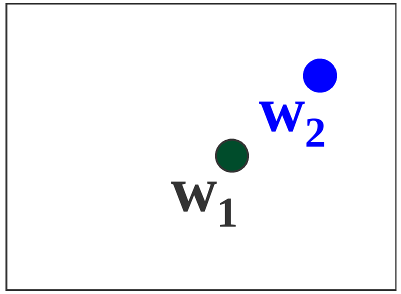
Illustration of How SOM Works

- Randomly initialize weights of the neurons.
  - Or, set them to two largest **PCA** eigenvectors (learning is much faster).
- Training records are fed to the SOM one at a time.
- SOM learning makes neighboring neurons respond similarly to similar inputs.

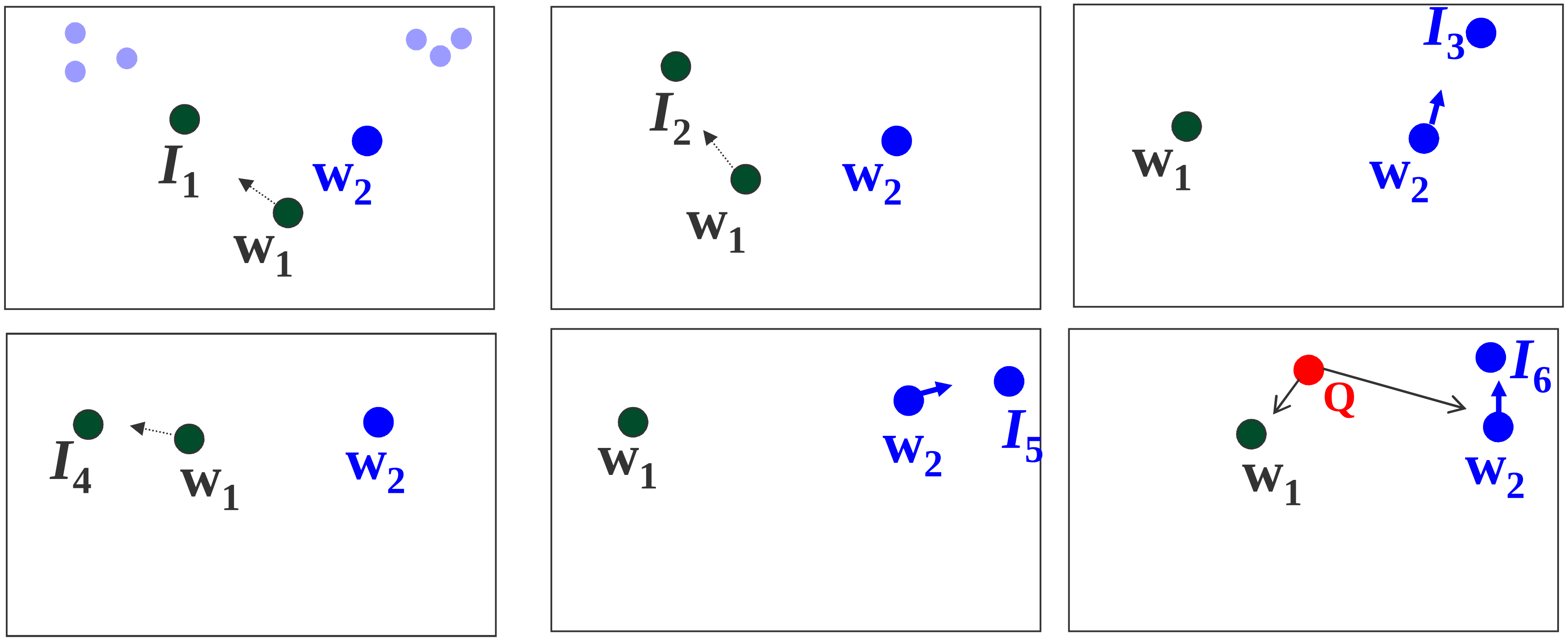




Learning in SOM



- Determine the winner in the learning process.
  - Pick one neuron w/ weights **<u>closest</u>** to the input vector.
  - Move the weight vector ***w*** of the winning neuron towards the input ***I***.
- After learning, neurons w/ similar weights tend to cluster on map.
- Query similar to instance-based learning, except on summary SOM.
  - **Eager**



SOM Reward and Penalty

- Not only move the weight vector $\boldsymbol{w_a}$ of the winning neuron towards the input $\boldsymbol{I}$.
- But also move the weight vector $\boldsymbol{w_b}$ of the most different neuron away from $\boldsymbol{I}$.
  - Penalty (negative impact) on the most different node.

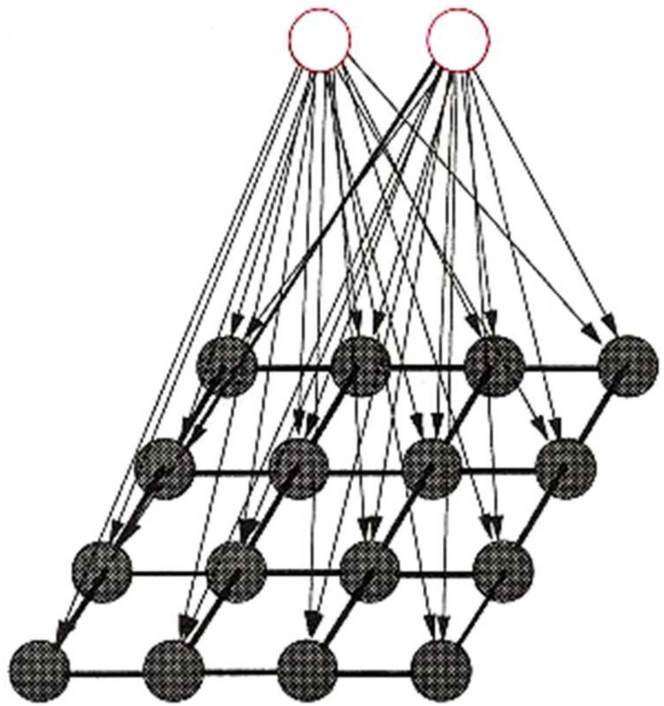
selected

Inputs

rejected

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	-1	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

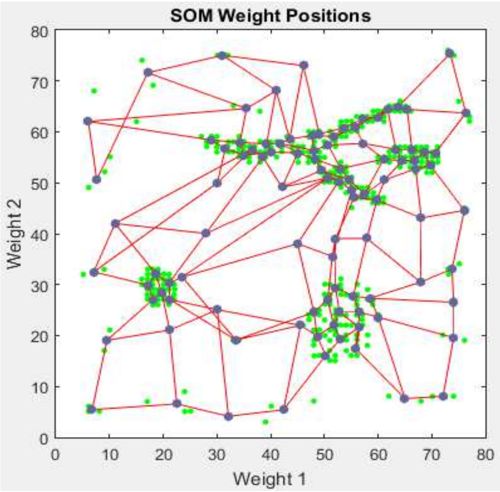
$+ \varepsilon$

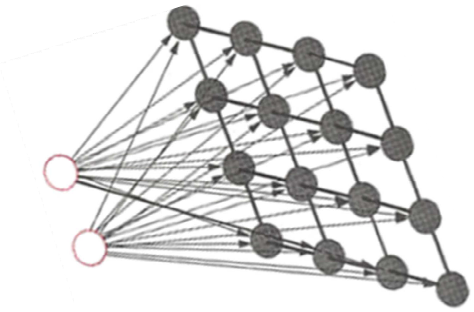


Self Organizing Map

Visualizing SOM Neighborhood

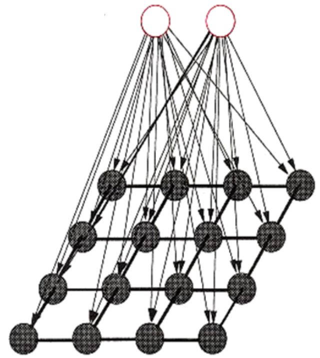
1	**0.5**	**0**	**0**	**0**	**0**
0.5	**0.5**	**0**	**-0.5**	**-0.5**	**-0.5**
0.5	**0.5**	**0.5**	**-0.5**	**-1**	**-0.5**
0.5	**1**	**0.5**	**-0.5**	**-0.5**	**-0.5**
0.5	**0.5**	**0.5**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**





- Since vectors of the similar neighborhood are moved in the same direction, similar records tend to excite adjacent neurons.
- Therefore, SOM forms a semantic map where similar neurons are mapped close together and dissimilar apart.

Detailed SOM Algorithm

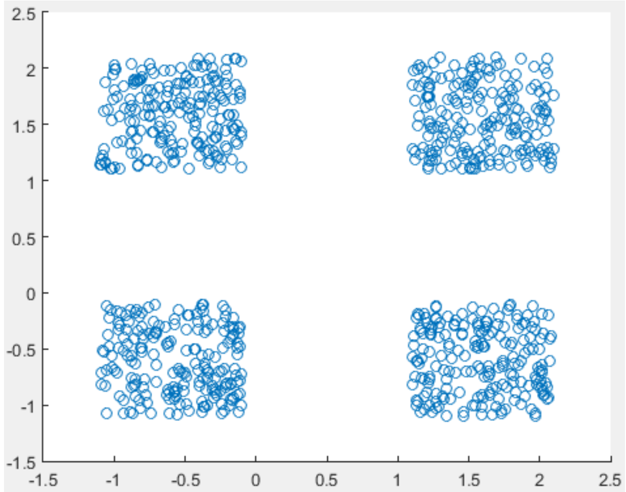


1. Randomize nodes' weight vectors in SOM
2. Take an input vector X_i
3. Check each node in the map
   - Use Euclidean distance to find all vectors in SOM that are within distance r from X_i
   - Denote the node that has the smallest distance as ***B**est **M**atching **U**nit*, BMU
4. Update nodes in neighbourhood of BMU by pulling them closer to input vector
   - All neighbors m_k of BMU move toward X_i as $m_k = m_k + \alpha (X_i - m_k)$
5. Decrease α and r by c%, then repeat from 2
   - SOM performance depends on **learning rate** α and **neighborhood threshold** r

1	**0.5**	**0**	**0**	**0**	**0**
0.5	**0.5**	**0**	**-0.5**	**-0.5**	**-0.5**
0.5	**0.5**	**0.5**	**-0.5**	**-1**	**-0.5**
0.5	**1**	**0.5**	**-0.5**	**-0.5**	**-0.5**
0.5	**0.5**	**0.5**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

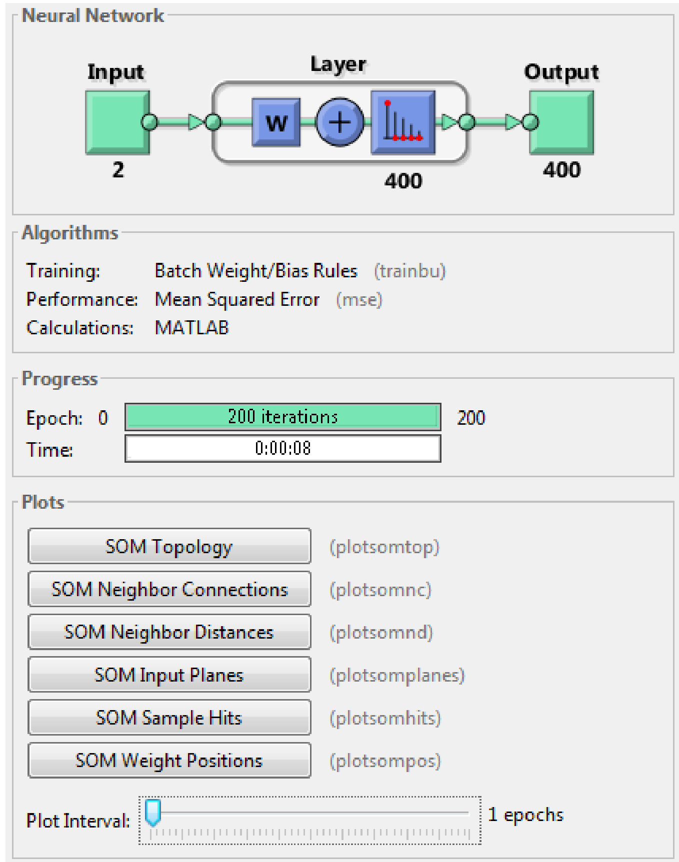
SOM, First Example

```
rng(3)        % set random seed to get same result
K = 200;      % number of samples of each cluster
q = 1.1;      % offset of groups
% define 4 clusters of input data
X = [rand(1,K)-q rand(1,K)+q rand(1,K)+q rand(1,K)-q;
    rand(1,K)+q rand(1,K)+q rand(1,K)-q rand(1,K)-q]';
```



% define net, http://www.mathworks.com/help/nnet/ug/cluster-with-self-organizing-map-neural-network.html

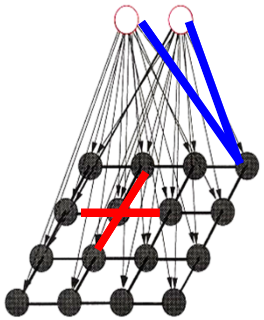
```
dimensions   = [20 20];
topologyFcn  = 'gridtop';    % Topologies (gridtop, hextop, randtop)
distanceFcn  = 'linkdist';   % Distance Functions (dist, linkdist, mandist, boxdist)
coverSteps   = 100;          initNeighbor = 10;

net = selforgmap(dimensions, coverSteps, initNeighbor, ...
                 topologyFcn, distanceFcn);
[net, Y] = train(net, X');
```

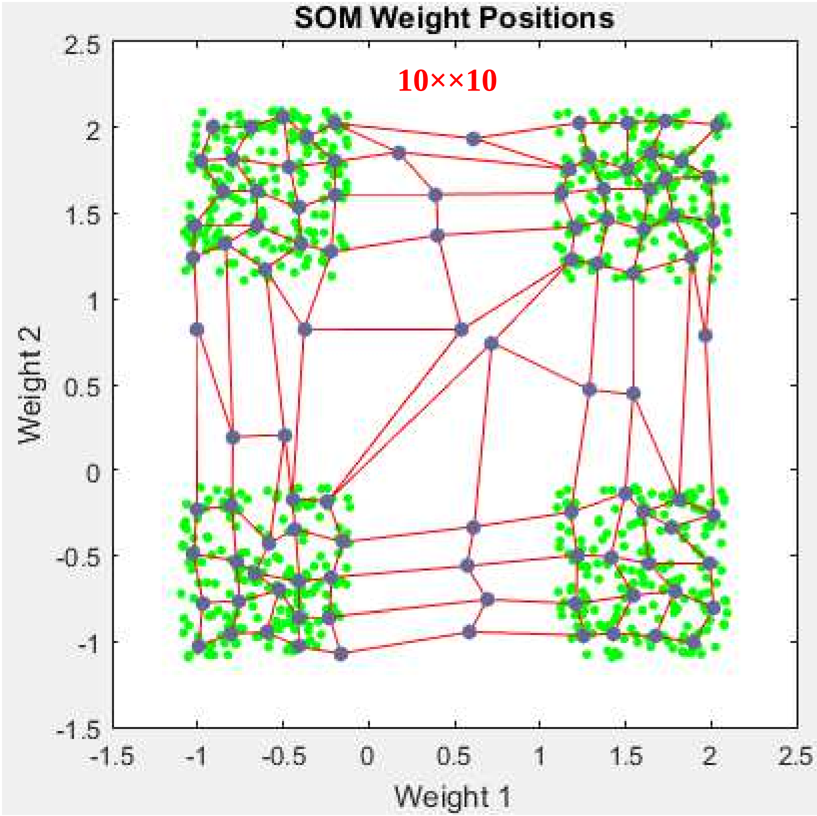


- You can watch training interactively.

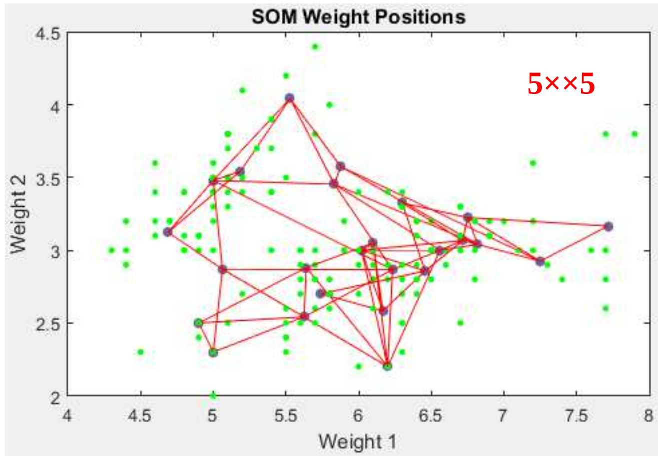
Plot Weight Positions➔ plotsompos()

- Show **input data as green dots**.
- Plot **weights to each neuron** as a dot (i.e. **blue-gray dots**).
- Plot **each neuron's neighboring (connecting) neurons**.

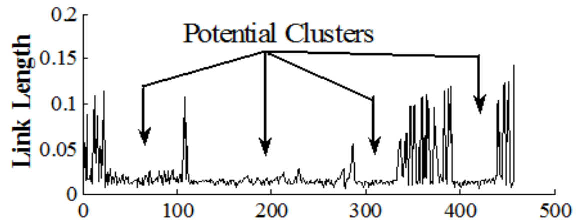




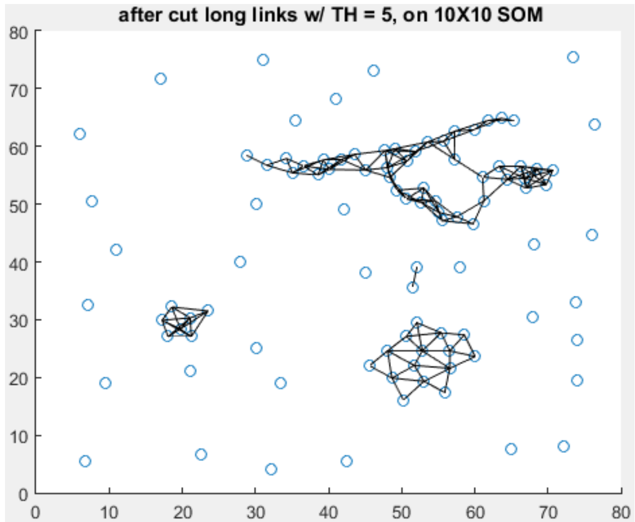

```
x = iris_dataset;
net = selforgmap([10 10]);
net = train(net, x);
plotsompos(net, x)
```

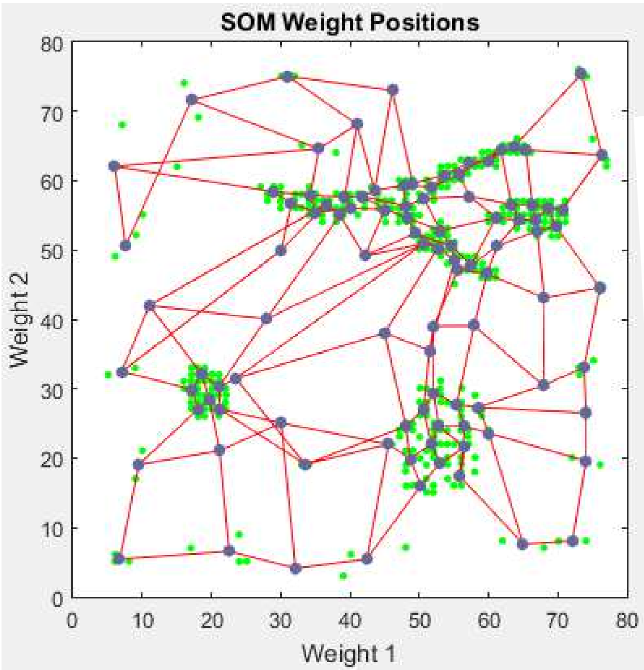


Weight Positions for Y-Shape Clustering

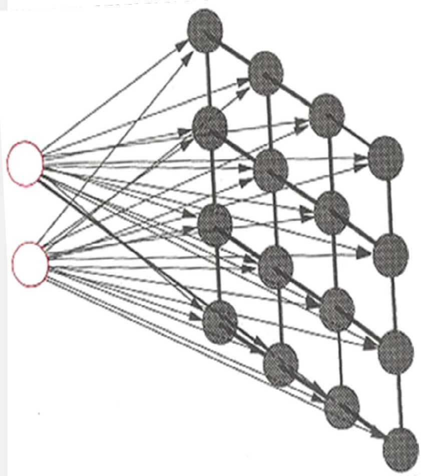


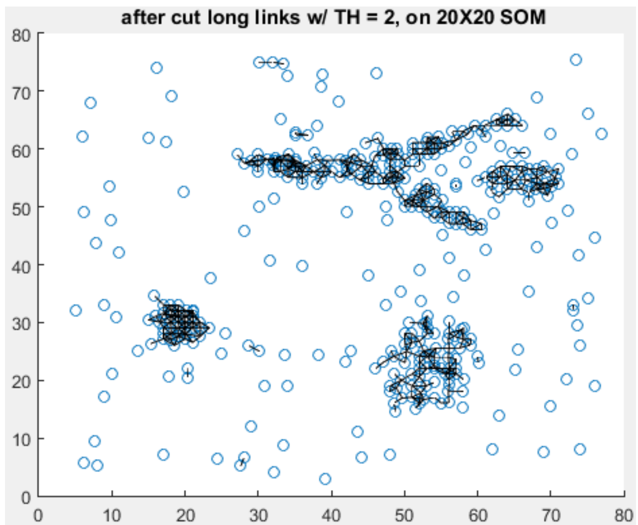
- **Cut** connections w/ too long distance ➔
  - Still connected neurons become clusters.





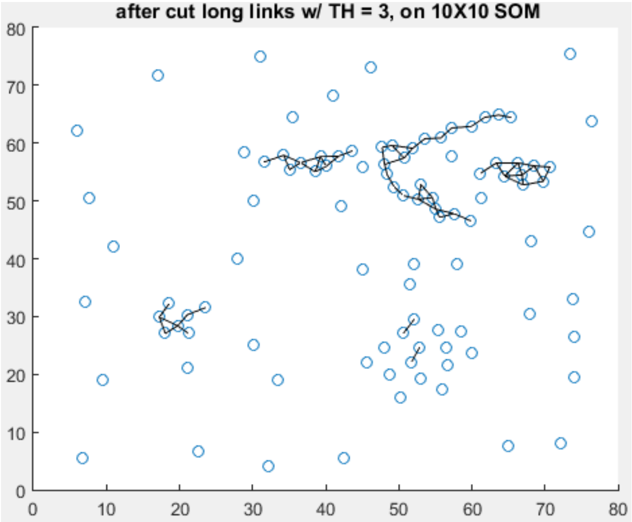
10×10, grid

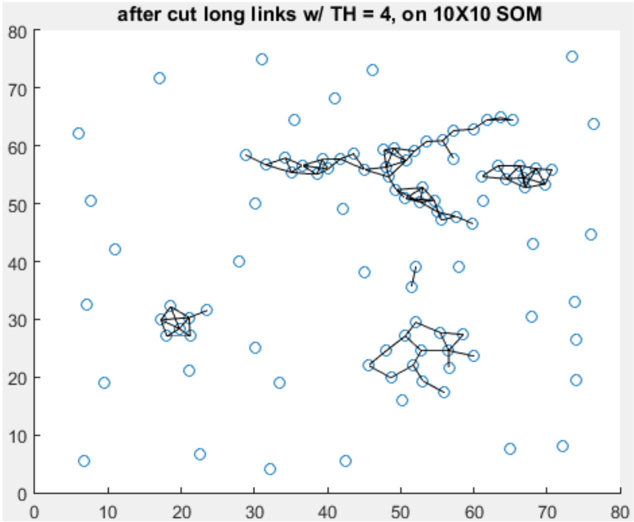


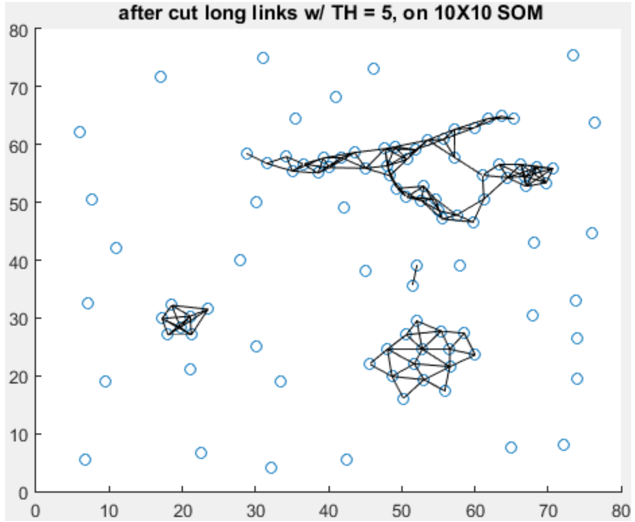


- Traverse the neuron connections based on their distance to its neighbors..

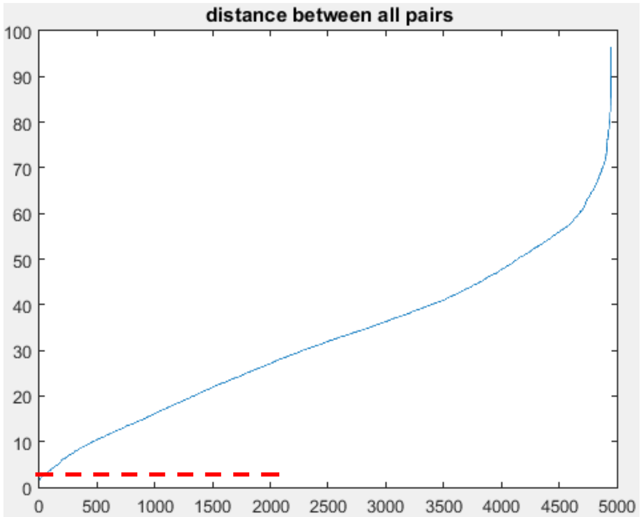
Code for Cutting Connections to Generate Connected Clusters

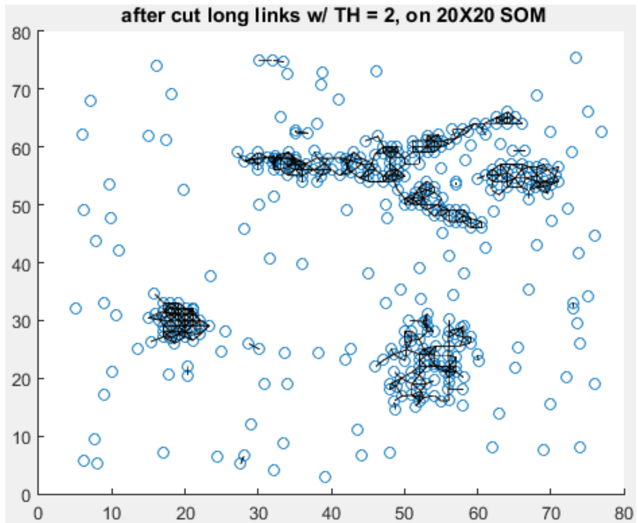




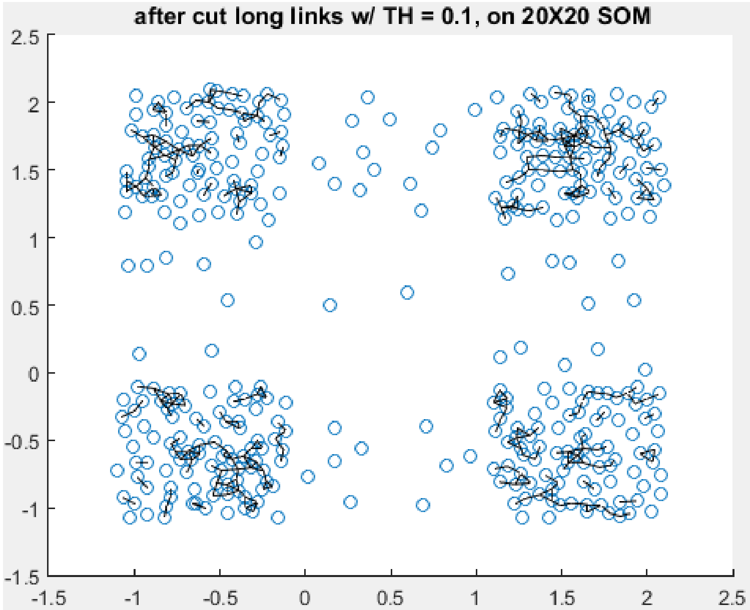


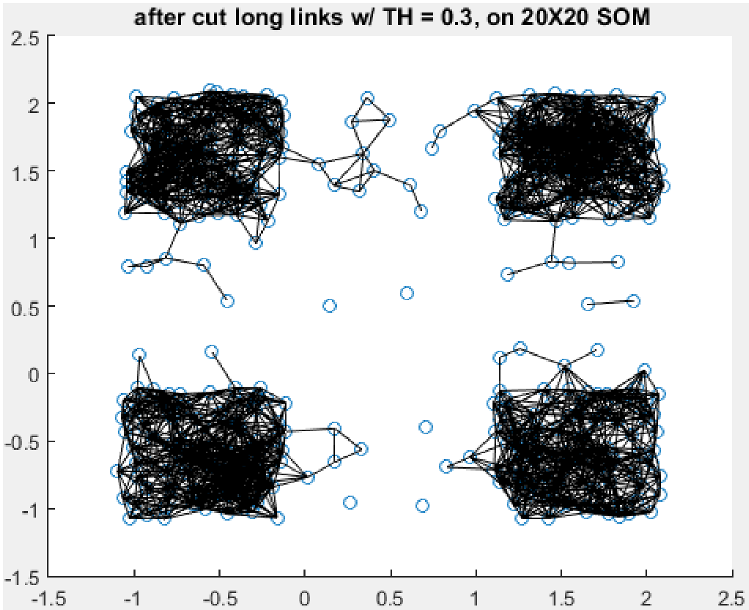

```
iw = net.IW{:};
pd = pdist(iw);
pdsq = triu(squareform(pd));

figure, scatter(iw(:,1), iw(:,2)),
hold on
pduu = pdsq;
pduu(find(pduu > Cut_TH))=0;
[rows cols] = find(pduu);
x1=[iw(rows, 1) iw(cols, 1)]';
y1=[iw(rows, 2) iw(cols, 2)]';
plot(x1,y1, 'k'), hold off
```

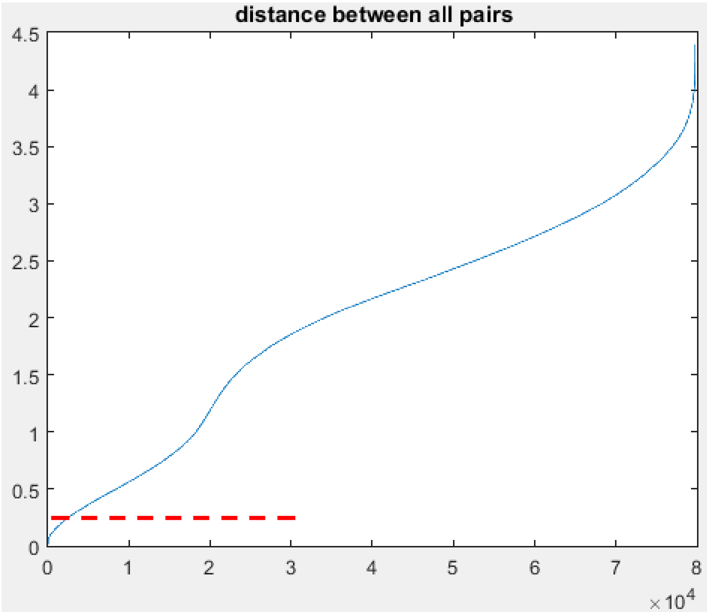


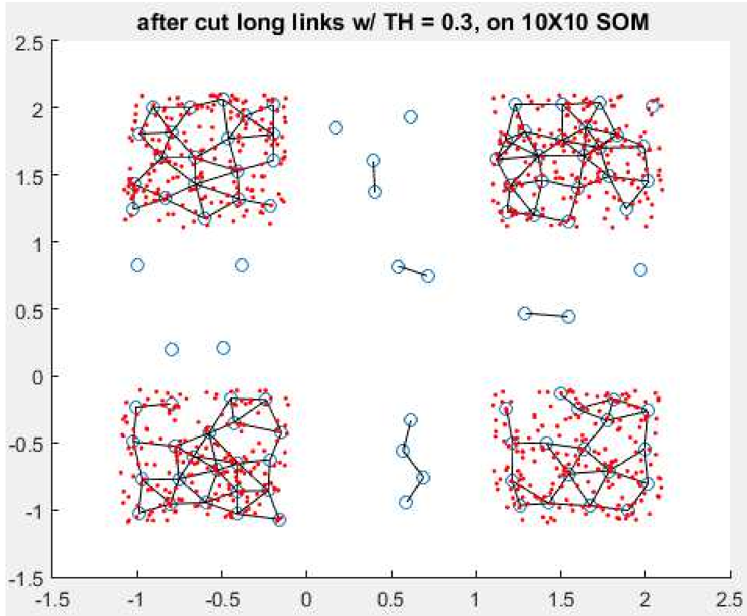


Another Example of Cutting Connections





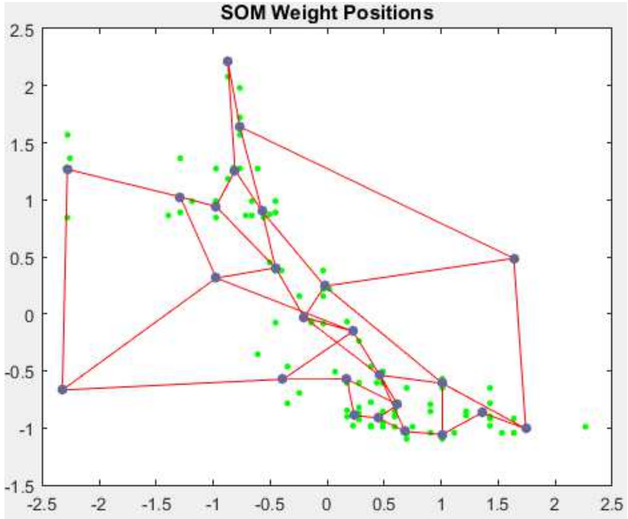


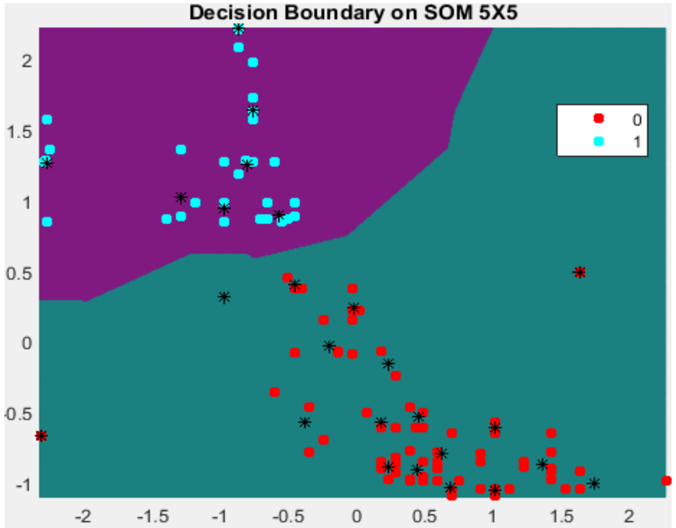


hold on,scatter(X(:,1),X(:,2), 'r', '.'); hold off

*k*NN Classification **AFTER** SOM Clustering

- For each **neuron** finds its class representation by ***k***nn search of **training data**, (i.e. ***k*** = 1).
- For each **test data** finds its class by another ***k***nn search of **neurons**, (i.e. ***k*** = 1).





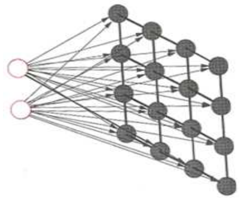
- Clustering “MPG + Displacement” to Predict # of Cylinders, ***k*** = 1
  - Remember to do **standardization**!!!

Digits Recognition (Classification) using SOM

- Training dataset from Machine Learning Repository of the Center for Machine Learning and Intelligent Systems.
  - 1934 handwritten digits from 0 to 9 collected from a total of 30 people.
  - Each digit sample has been normalized to 32x32 binary image
  - See http://www.codeproject.com/Articles/793537/Self-organizing-Map-Implementation

Class	Number of Samples
0	189
1	198
2	195
3	199
4	186
5	187
6	195
7	201
8	180
9	204

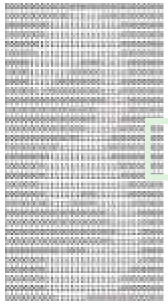
Workflow and Basic Program



```
load digit_som_train;   % 1024x1934
load digit_som_label;   % 1x1934
```

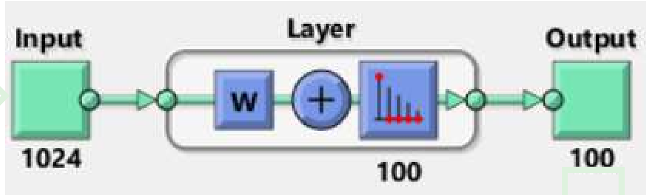
32

32

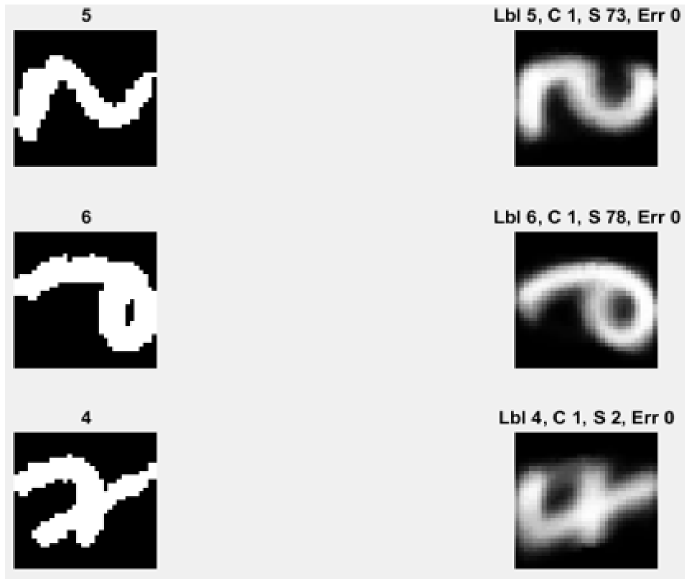


% Each neuron has 1024 connections (inputs)

```
net = selforgmap([10 10]);
net.trainParam.epochs = 50;
net = train(net, train_data);

tmp_neuro_vecs = sim(net, train_data);
TrainRec_LandOn_NeuronNo = vec2ind(tmp_neuro_vecs);
```



input **neurons**

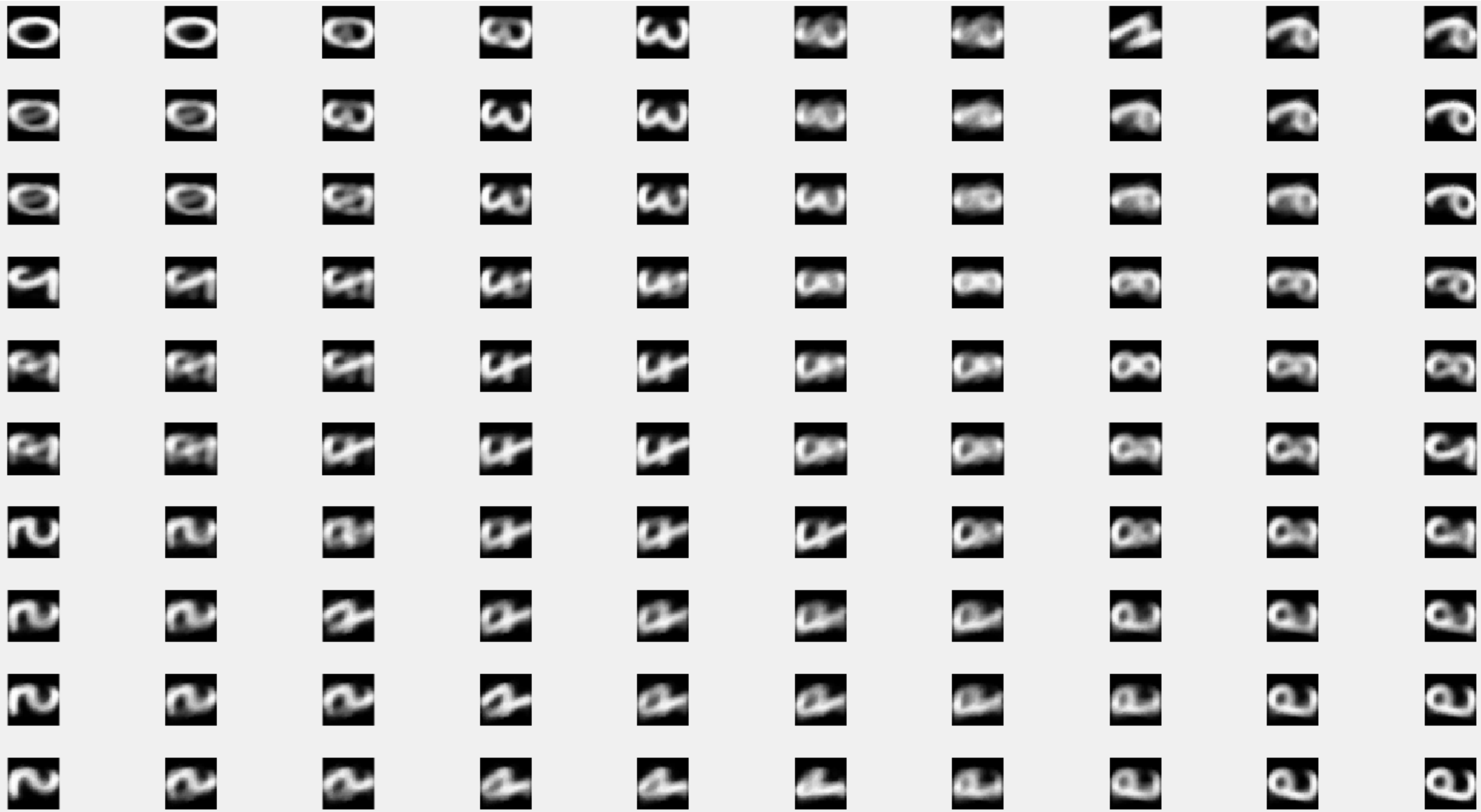


- You can then check which digits land on which neurons for how many times.
- Then, perform prediction and test accuracy…

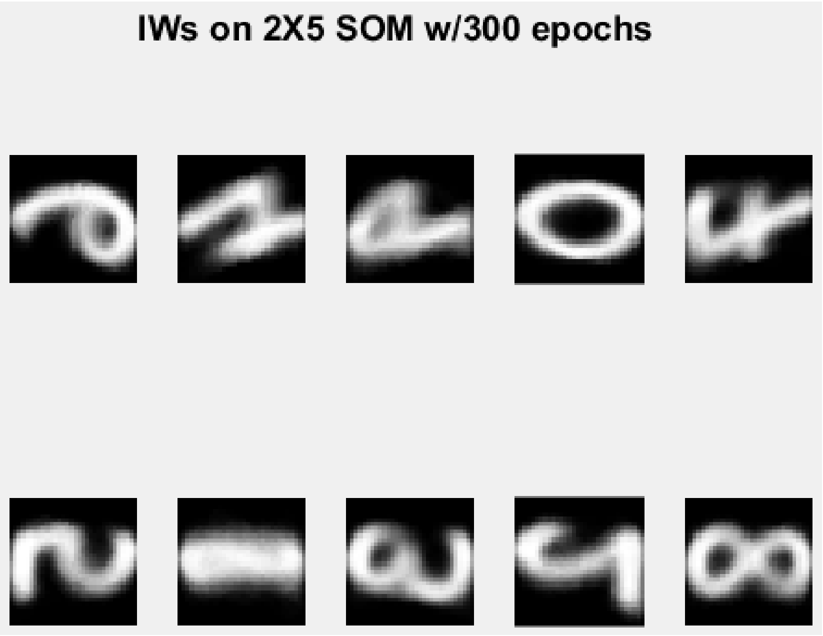
Results of Trained SOM

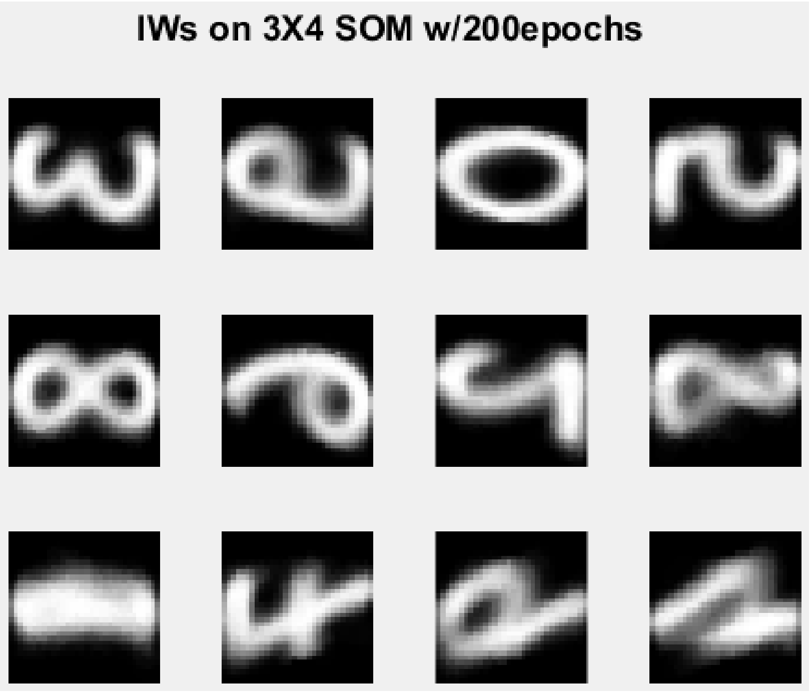
```
IW_Array = net.IW{1};
figure,
for i = 1 : 100,
    img=IW_Array(i,:);
    subplot(10, 10, i),
    imshow(reshape(img, [32 32]));
end
```

- Input weights on 10×10 SOM neurons.
  - Each neuron was trained by1024 (32×32) input pixels from all trainings.



Different SOM Configurations on Digit Recognition





Each Neuron Represents Which Digit?

- Table shows frequency of training records w/ different labels hits each neuron.
- A neuron predicts the label with maximum hits (by training).
  - Confidence of the prediction = max_hit / total_hit
  - Confidence(“neuron1 in predicting digit 2”) = 65 / 68 = 96%

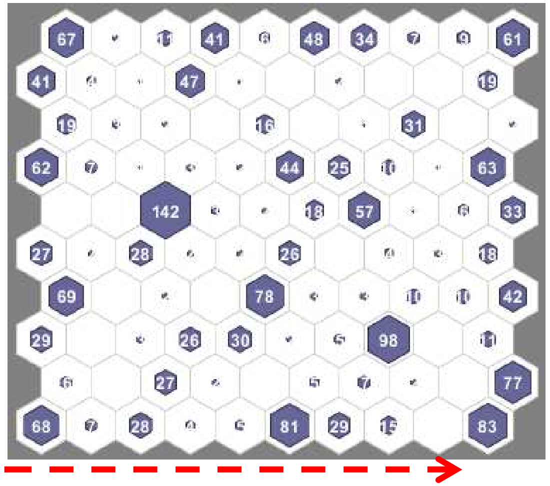
Neurons

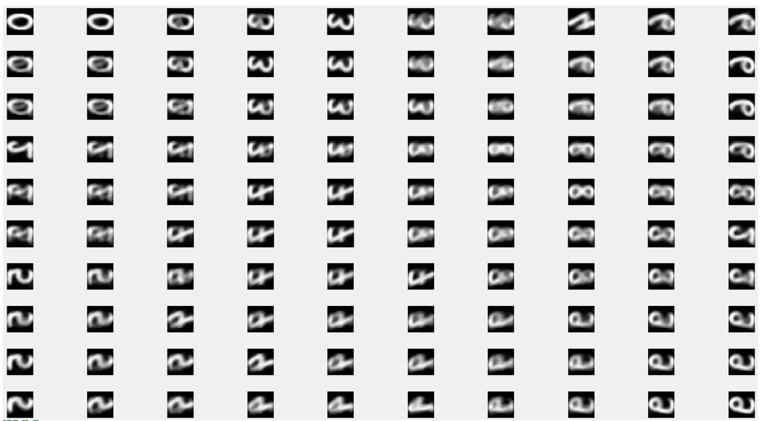
Hits by training digits 0..9

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	65	2	6	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	2	1	4	0	0	0	0
4	0	0	0	3	1	0	0	0	0	1	0	0
5	1	2	2	0	0	0	0	0	0	0	4	0
6	0	0	2	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	0
8	0	2	0	0	0	0	0	0	0	81	0	0
9	0	0	0	0	4	79	28	11	0	0	0	0
10	1	1	18	1	0	0	0	0	0	1	2	0

68 7 ……

Total hits of each neuron

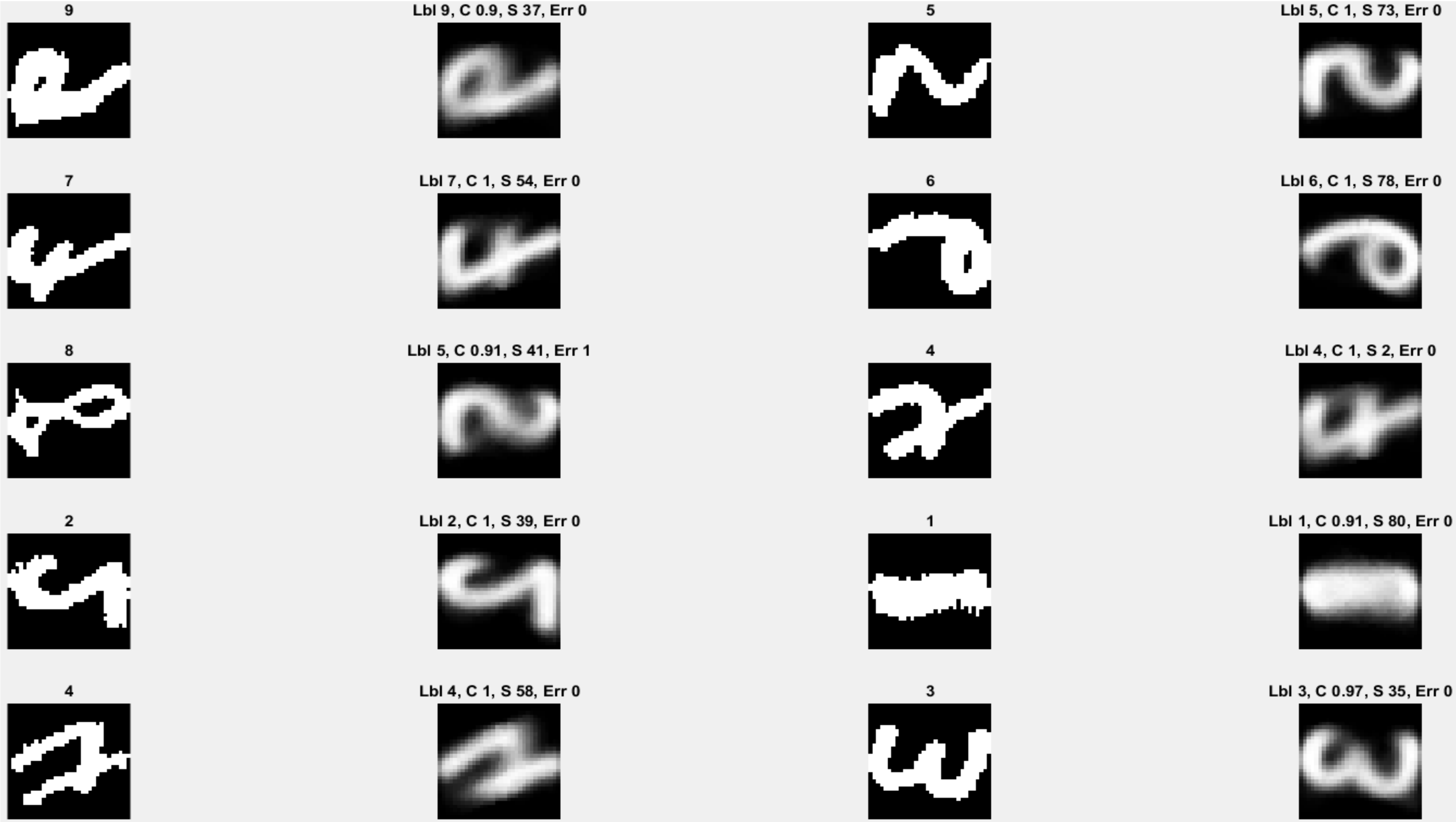




Classification **AFTER** SOM Clustering

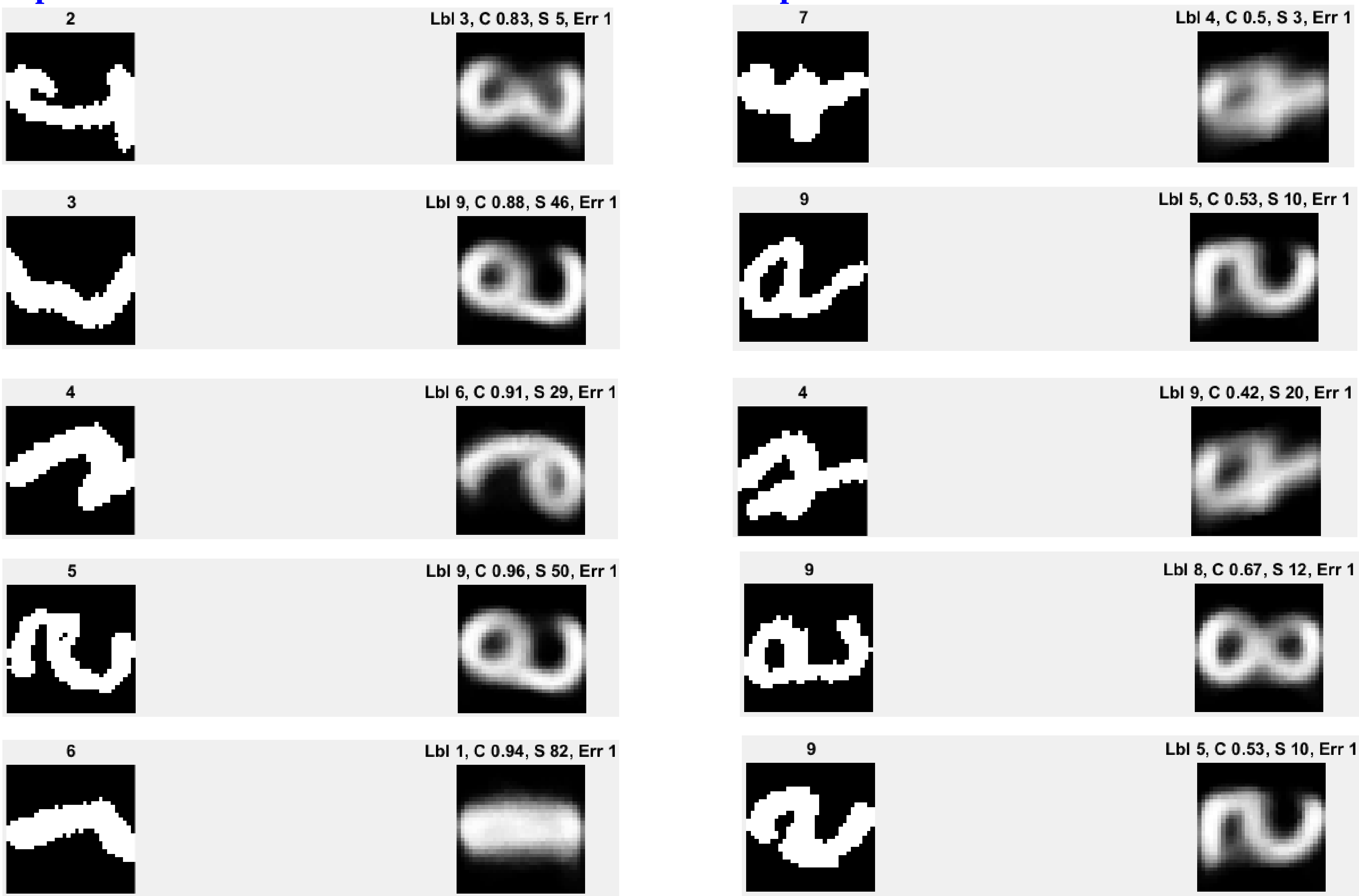
- In this test, accuracy = 90%.

input **hit neuron** **input** **hit neuron**



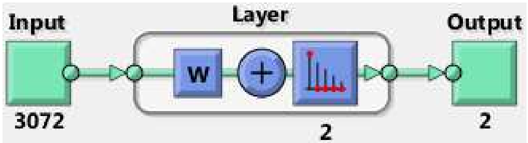
Cases for Incorrect Predictions

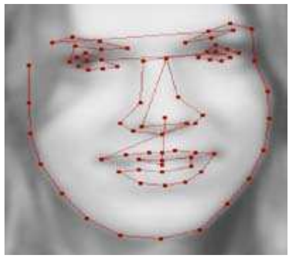
input **hit neuron** **input** **hit neuron**



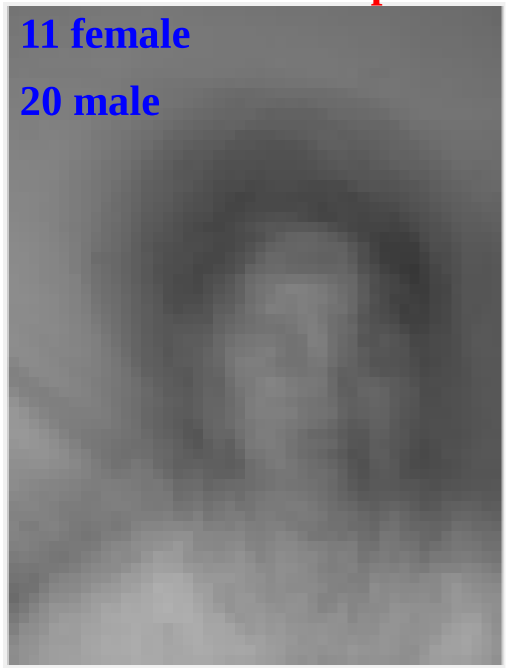
Gender Clustering

- MUCT Face DB www.milbo.org/muct
- Sample 36 female, 40 male.
  - Input 3072 gray-scale pixels.
  - Output SOM 2×1.
    - Average face? of different gender?





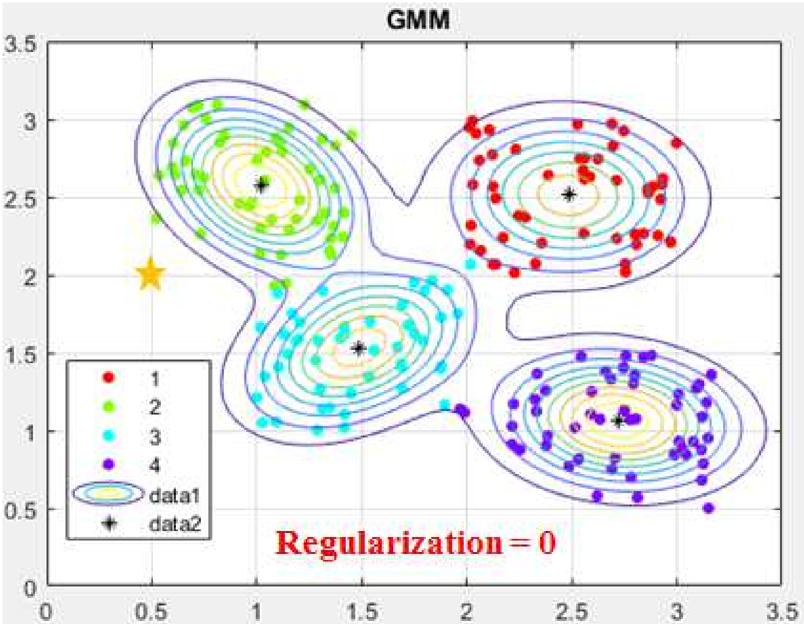
2×1 SOM output



Identify Outliers, GMM, One Class SVM, **How about SOM???**

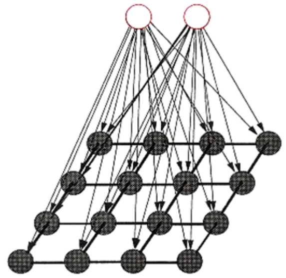
- Build **4** one-class SVM-RBF models.
  - RBF, **outlier** ϵ **C3**, but ***P*↓↓↓**
  - GMM, **outlier** ϵ **C2**, but ***P*↑↑↑** w/ PDF↓
  - RBF, **outlier** ϵ **C3**, but ***P*↓↓↓**
  - GMM, **outlier** ϵ **C1**, but ***P*↑↑↑** w/ PDF↓

Data Point = 0.5	**2**	**RBF1**	
P = 2.572e-06	5.3751e-05	**0.00013274**	1.1234e-06
Data Point = 1.0	**2.5**	**RBF2**	
P = 0.022418	**0.98502**	0.00017523	2.1481e-07
Data Point = 1.5	**1.5**	**RBF3**	
P = 0.0013203	0.0011747	**0.99333**	7.409e-07
Data Point = 5 5		**RBF4**	
P = 1.5639e-06	1.8047e-06	**0.0003285**	1.2992e-06



Data Point = 0.5	**2**	**GMM Regularization = 0**		
P = 0.0000	**0.9868**	0.0132	0.0000	0.0032
Data Point = 1.0	**2.5**			
P = 0.0000	**0.9999**	0.0001	0.0000	0.4794
Data Point = 1.5	**1.5**			
P = 0.0000	0.0029	**0.9965**	0.0005	0.4312
Data Point = 5 5				
P = **1.0000**	0.0000	0.0000	0.0000	0.0000

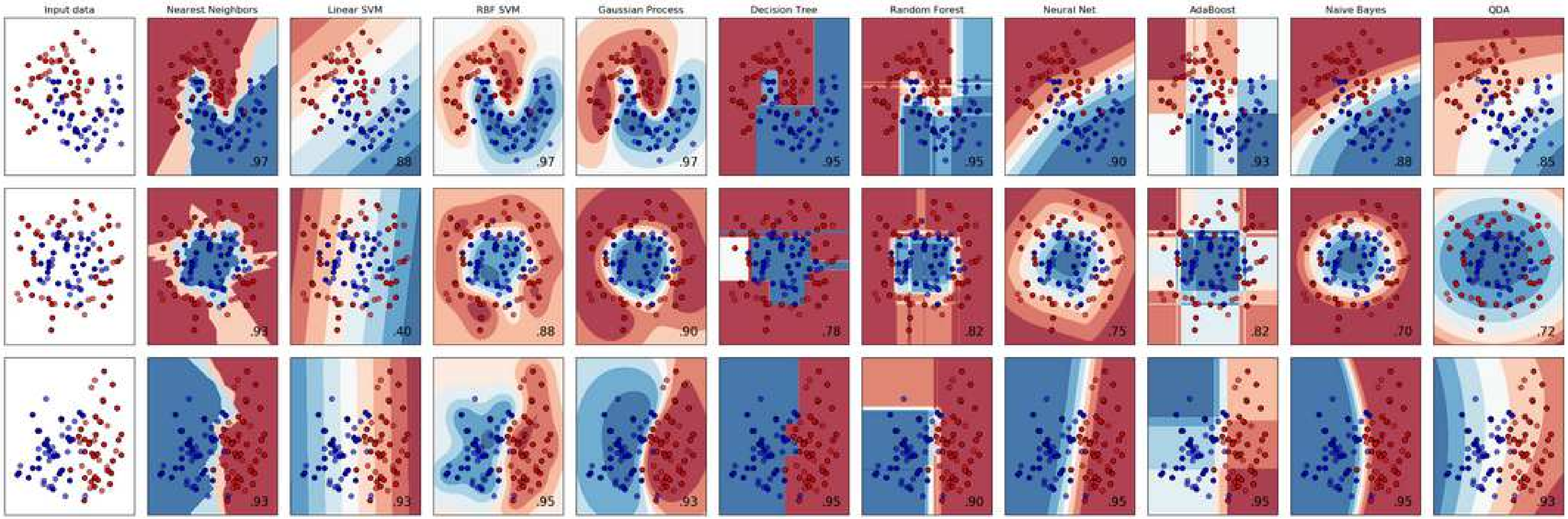
SOM Summary



- **Unsupervised** learning.
  - All other NNs discussed so far are supervised learning.
  - Use a neighborhood function to preserve topological properties of the input space.
  - Produce a low-dimensional (typically two-dimensional) map as a result.
  - Good for visualize high-D data in low-D, akin to multidimensional scaling.
  - Introduced by Finnish professor Teuvo Kohonen in 80s, as a **Kohonen map / NTWK**.
  - "Training" builds a map using inputs (a competitive process = vector quantization).
  - “Mapping" automatically classifies new unseen input vectors.
  - Usually arrange neurons is a two-dimensional grid.
  - Small SOM ≈ ***K***-means, larger SOM rearrange data in a totally different way.
  - https://en.wikipedia.org/wiki/Self-organizing_map

Classification Comparison

- Remember you can do clustering first, **THEN** do classification



- http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

Clustering Comparison

- **http://scikit-learn.org/stable/modules/clustering.html#dbscan**