Self-organizing Maps (SOM)

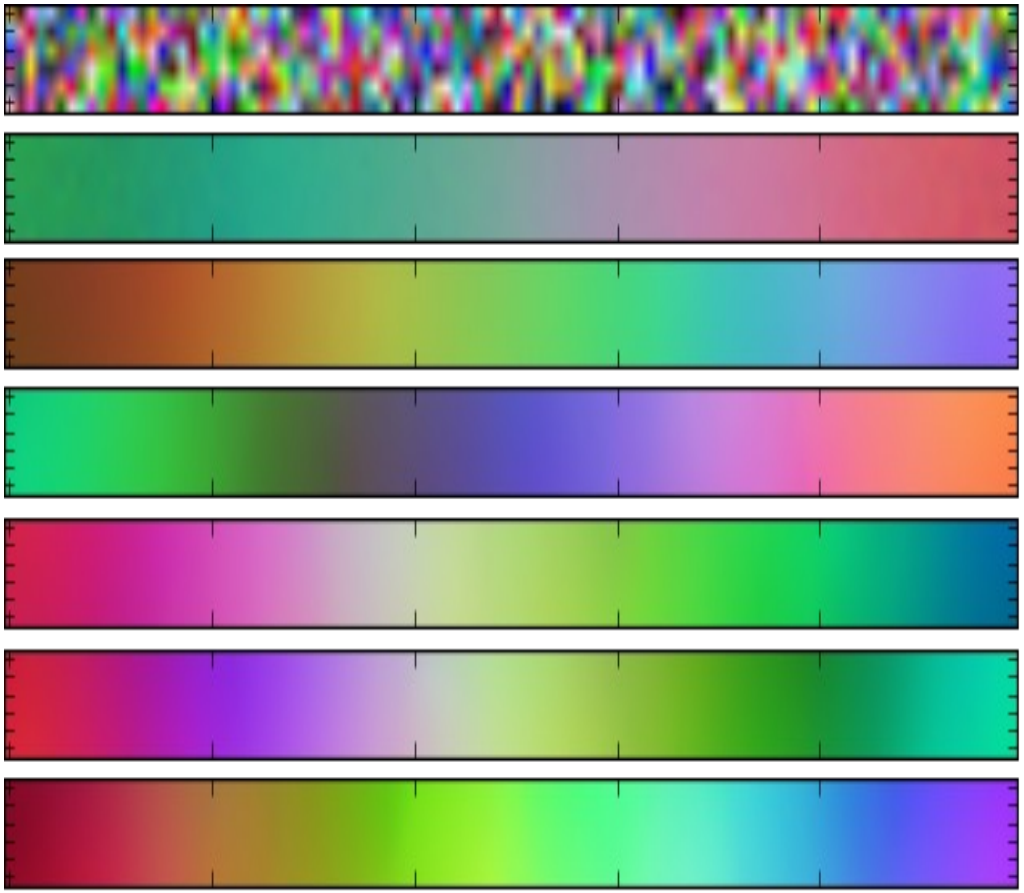
Like all maps, the basic aim of SOM is data compression. Given a large set of input vectors, find a smaller set of prototypes, vectors with the same features at the input, that can provide a good approximation to the whole input dataset.

Prototypes can become a basis for clustering the input data. For any vector in the dataset, find the prototype that is most like it. Call this prototype the “best matching unit” (BMU). Place input vectors with the same BMU in the same cluster.

“Self-organizing” because a machine-leaning algorithm will find the prototypes for us.

Colorful illustration of SOM in clustering

Input: randomly colored pixels. Output: ordered bands of color

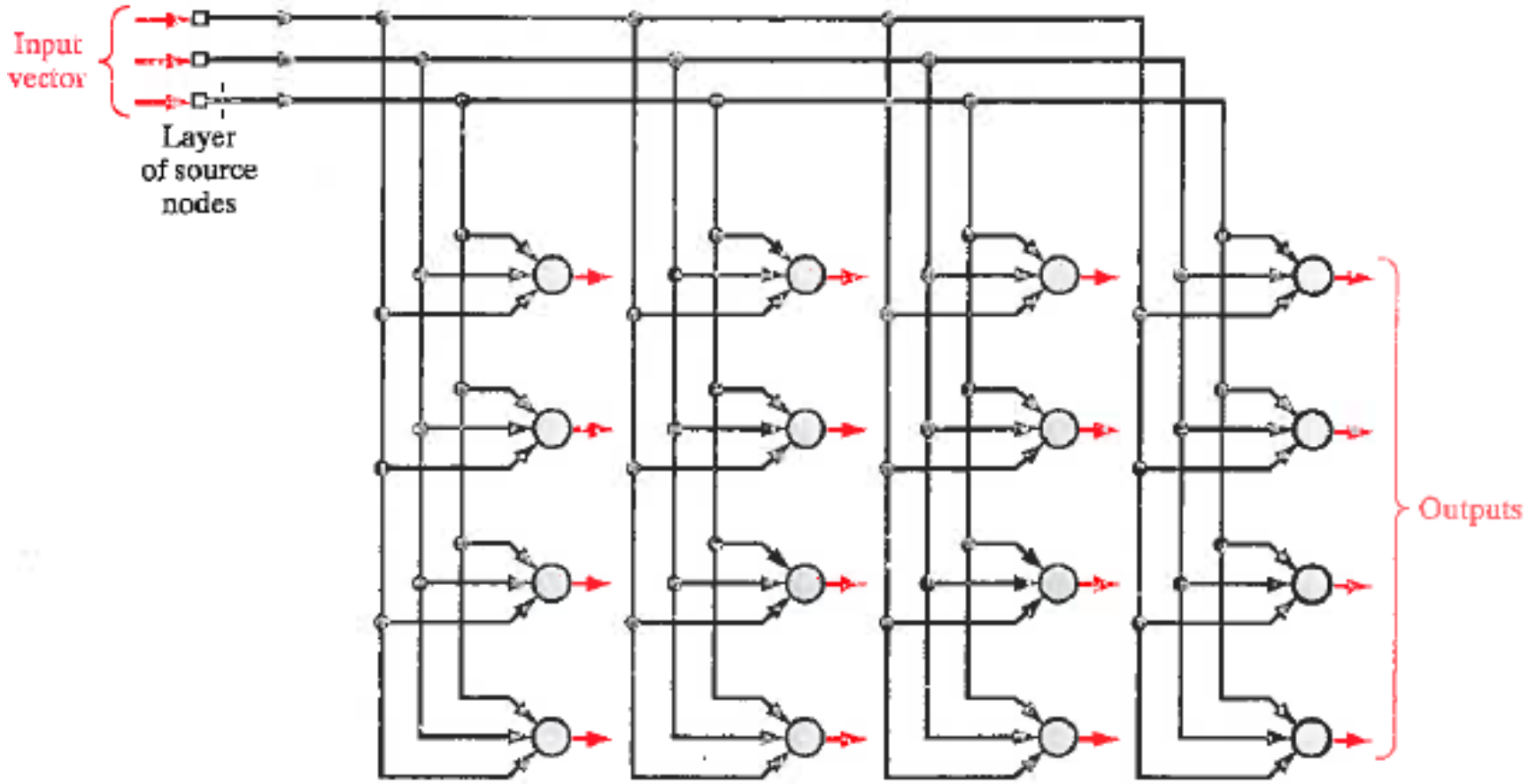


Each pixel has an input vector with specific RGB values. Find prototypes, vectors with RGB values that we recognize as purple, yellow, brown, etc. Cluster the pixels by BMUs and illustrate the clusters by placing pixels with the same BMU close together (color bands)

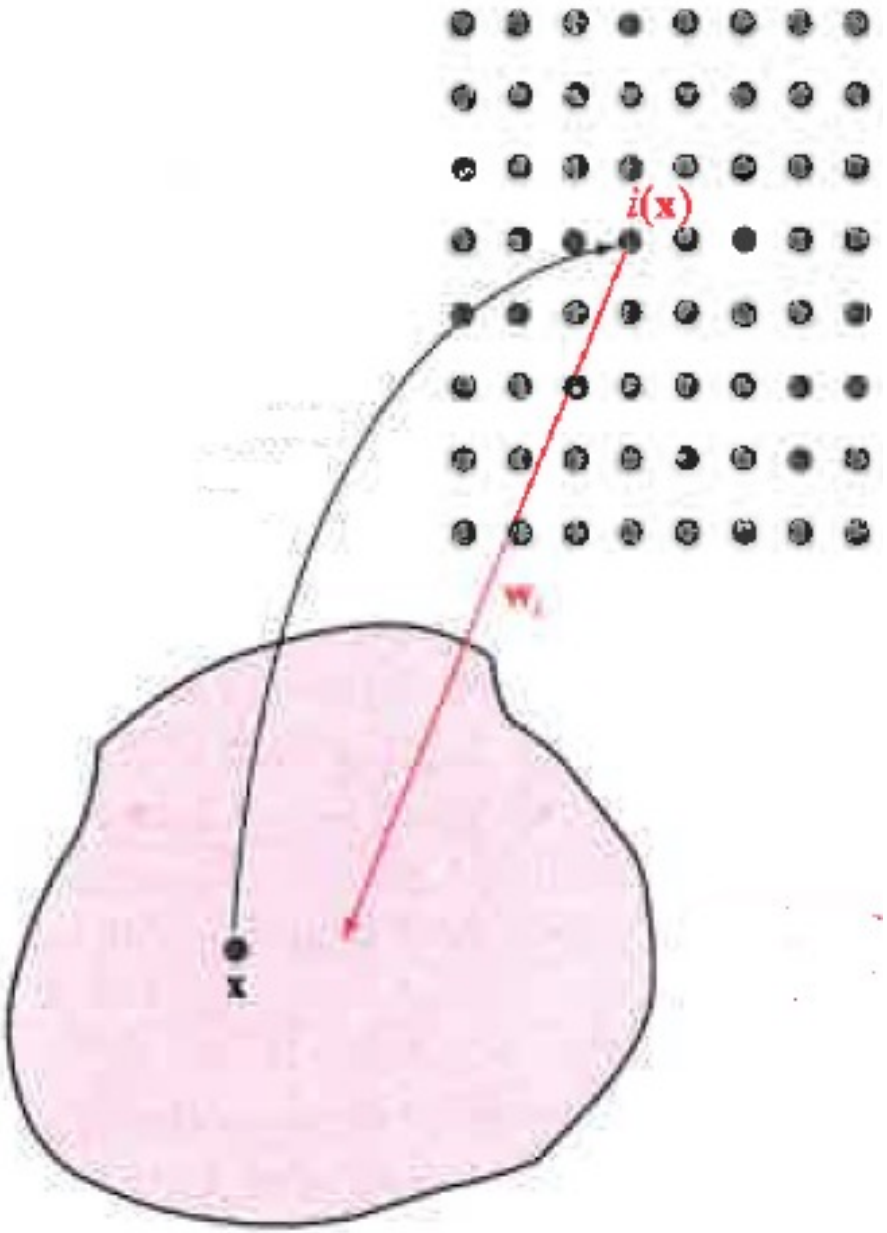
Process of finding prototypes is iterative. Color bands become more distinct as prototypes are refined.

In 1982 Kohonen published an algorithm to find prototypes based on input nodes connected to a 2D array of output nodes.

$\mathbf{w}_k$ = weight vector connecting input nodes to output node k



Creating prototypes



If the input space contains many instances like **x**, $i(\mathbf{x})$ will become the BMU of this type of input.

$\mathbf{w}_i$, the weight vector that connects input nodes to the i^{th} output node, will become a prototype of instances like **x**.

SOM algorithm

Initialize weights on all connections to small random numbers

3 aspects of training:

Output nodes compete for activation based on a discriminant function

Winning node, with largest value of discriminant, becomes the center of a cooperative neighborhood.

Neighborhood adapts to an input pattern because cooperation increases the susceptible to large values of the discriminant function.

Competitive learning

Given a randomly selected input vector **x**, the BMU is the output node with weight vector closes to **x**.

$$\boldsymbol{i}(\mathbf{x}) = \arg\min_{\boldsymbol{k}} \|\mathbf{x} - \mathbf{w}_{\boldsymbol{k}}\| \quad \boldsymbol{k} = 1, 2, \ldots L$$

where L = # of nodes in the output array and
$\mathbf{w}_{\boldsymbol{k}}$ = weight vector connecting input nodes to output node $\boldsymbol{k}$

Since $\mathbf{w}_{\boldsymbol{k}}$ have been normalized to unit length, the smallest Euclidean difference is equivalent to the largest dot product with **x** (the discriminant).

The output node with weight vector most like the randomly selected input vector wins the competition.

Gaussian cooperative neighborhood

$$h_{\mathbf{k,i(x)}} = \mathbf{exp}\left[-\frac{d^{\mathbf{2}}_{\mathbf{k,i}}}{2\sigma^2} \right]$$

Probability that output nodes k belongs to the neighborhood of winning output node **i**(**x**)

$d_{\mathbf{k,i}}$ is the lattice separation between k and **i**(**x**)

$\sigma(n) = \sigma_0 \exp(-n/n_{\mathbf{0}})$

Initially neighborhood is large.

Decreases on successive iterations.

Adaptation

$$\mathbf{w_k(n+1) = w_k(n) + \eta(n) h_{k,i(x)}(n)(x - w_k(n))}$$

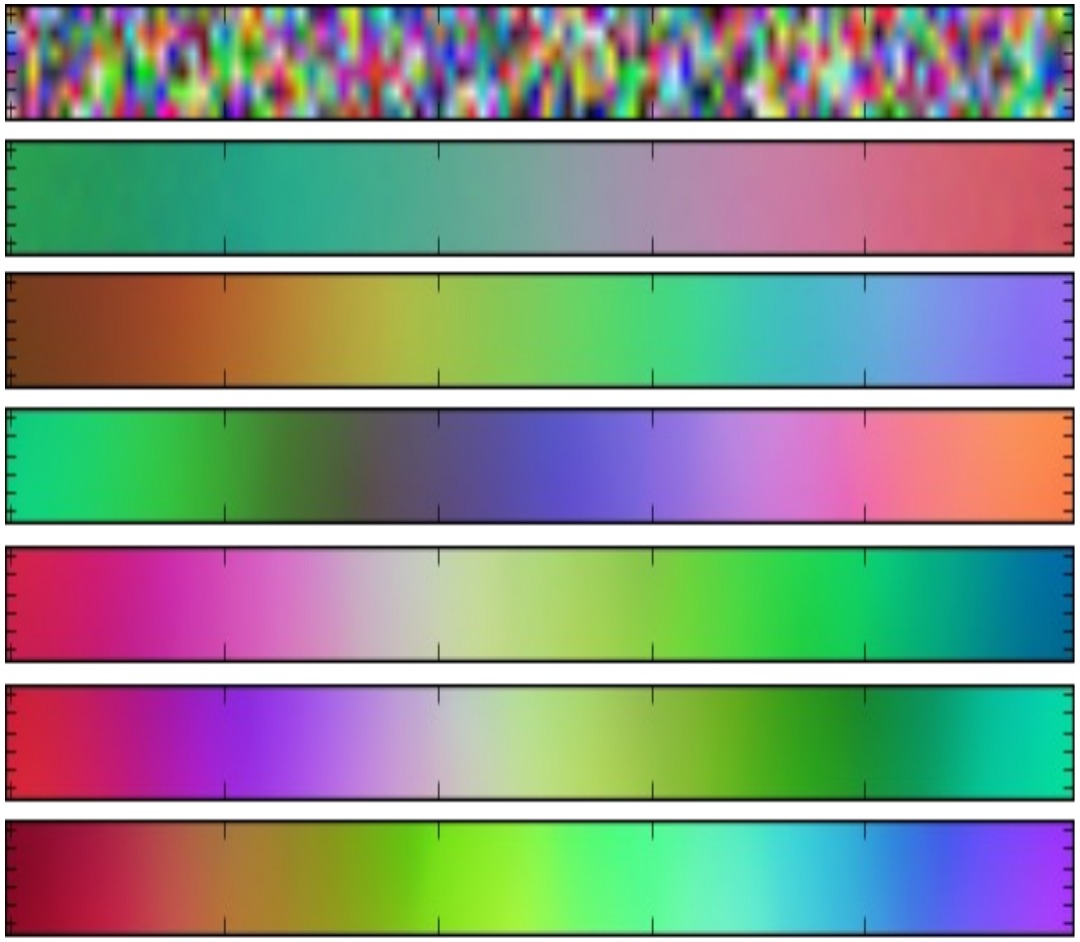
Modification on the n^{th} iteration is applied to the weights of all output nodes but focused on the neighborhood of winning node k.

Makes the weight vector connecting input **x** to the winning node more like **x**

Leaning rate $\eta(n) = \eta_0 \exp(-n/n_0)$

Colorful illustration of SOM in clustering

Input: randomly colored pixels. Output: ordered bands of color



σ ~ size of lattice

$$h_{\mathbf{k,i(x)}} = \mathbf{exp}\left[-\frac{d^{\mathbf{2}}_{\mathbf{k,i}}}{2\sigma^2}\right]$$

σ~nearest neighbors

Kohonen's famous example: SOM clustering of animals

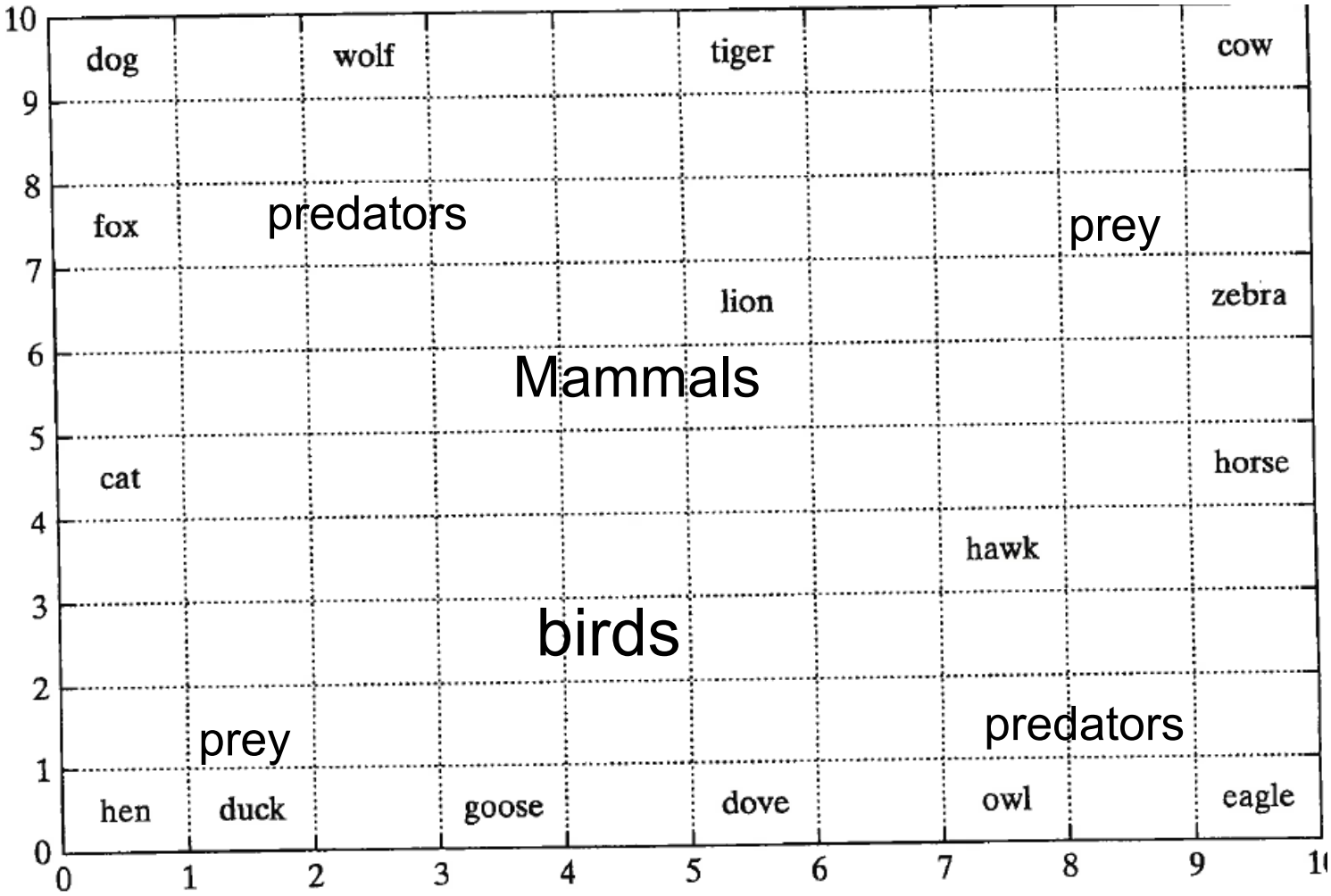
Animal		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

13 attributes of 16 animals

10 x 10 lattice, 2000 iterations of SOM training

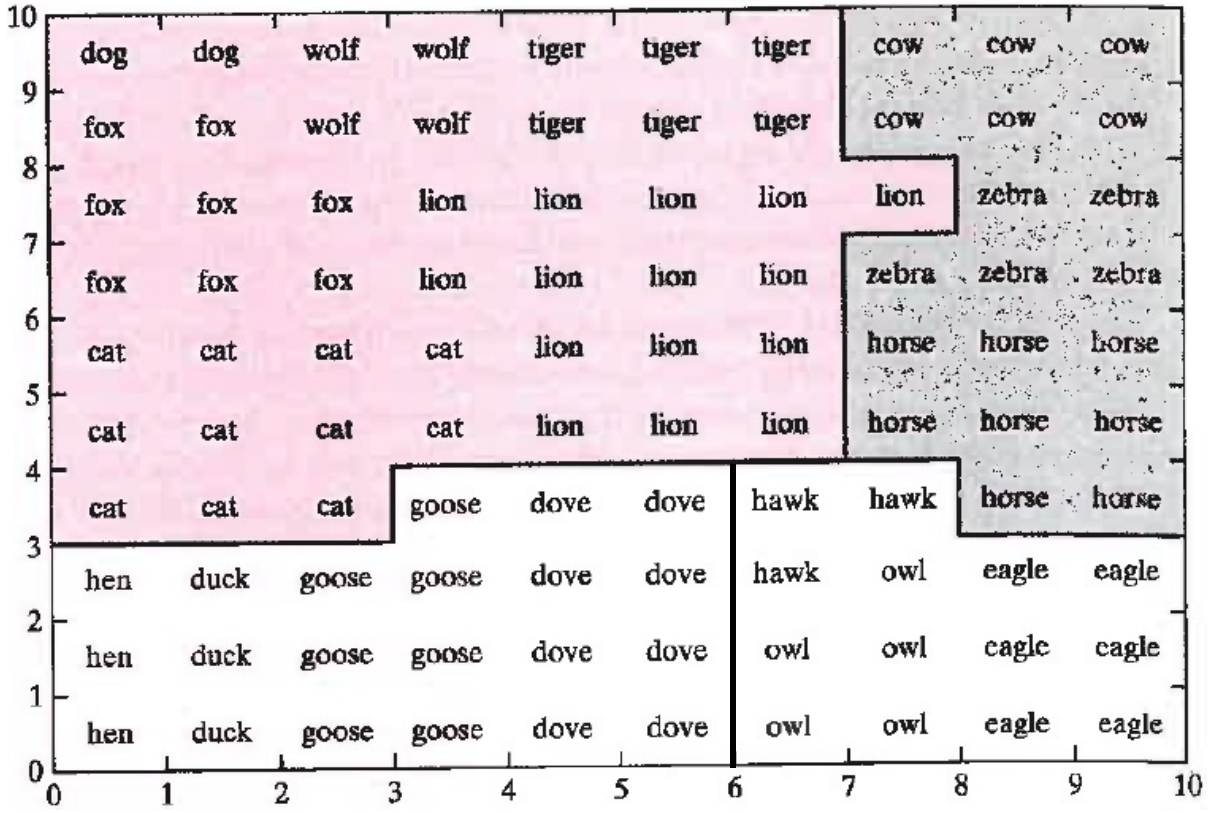
Note: not data compression!

Converged BMU of each training example is shown



Our prior knowledge detects topological ordering of BMUs
Design test examples to show strength of response of each node.

Semantic map: All lattice sites labeled by animal type inducing the strongest response test example.



Topology of responses define 4 region of output array based on our prior understanding of the meaning of the labels.

Unified distance matrix (u-matrix or UMAT) a function that defined the similarity of neighbors in output nodes

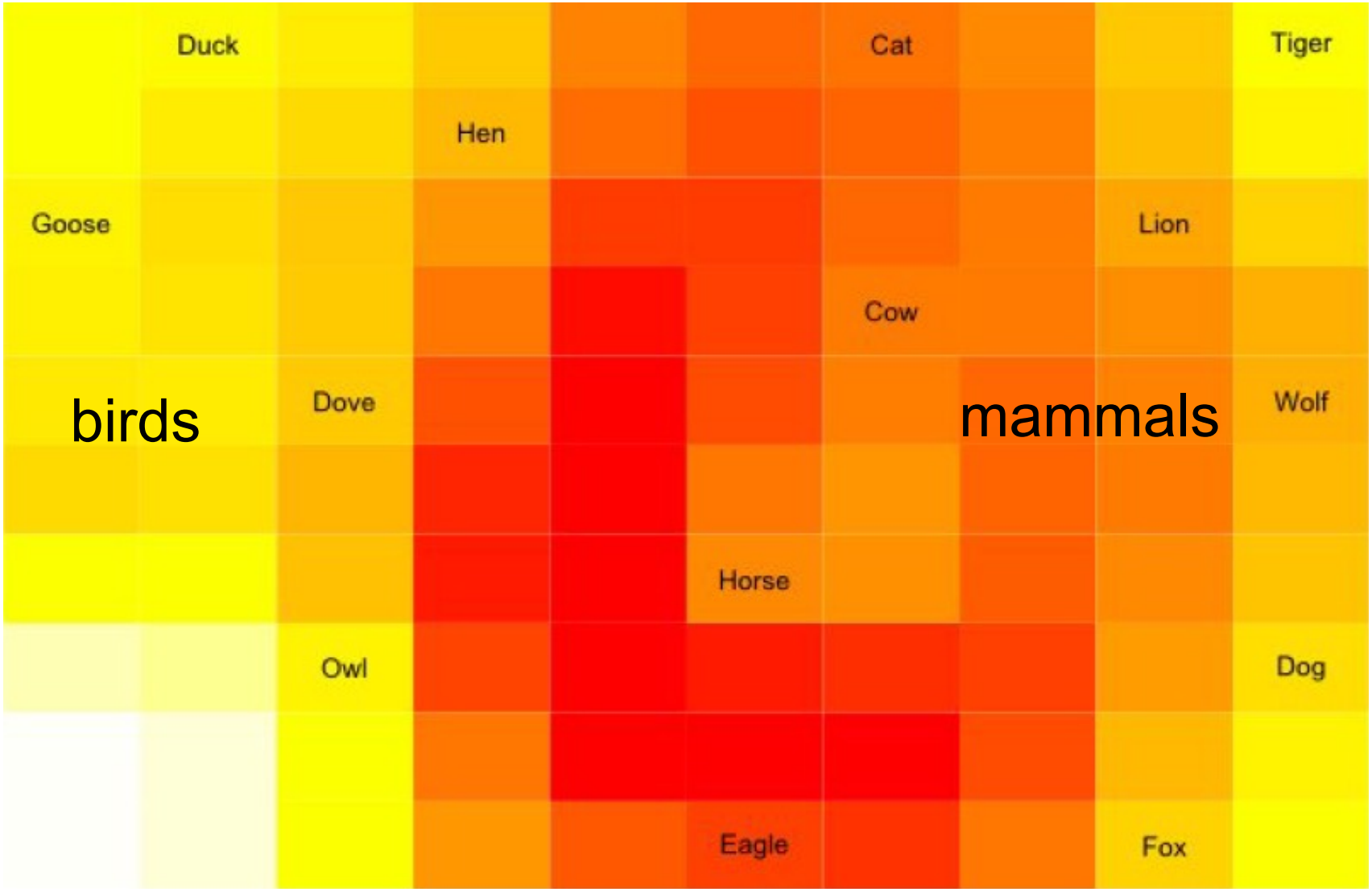
In a square lattice, corner, edge and internal output nodes have 2, 3, or 4 neighboring, respectively.

Euclidian distance between the prototype vectors of neighboring output nodes approximates the distance between different parts of the underlying input data space.

After averaging over neighbors, UMAT distance of each node is displayed as a heat map with darker colors for larger distance.

Similar prototypes denoted by groups of lightly colored nodes; darker colors suggests boundaries between the clusters of similar prototypes

UMAT of Kohonen's animal SOM: birds and mammals are in separate clusters of prototypes but fine structure is present.



SOM using MATLAB: Hands on example

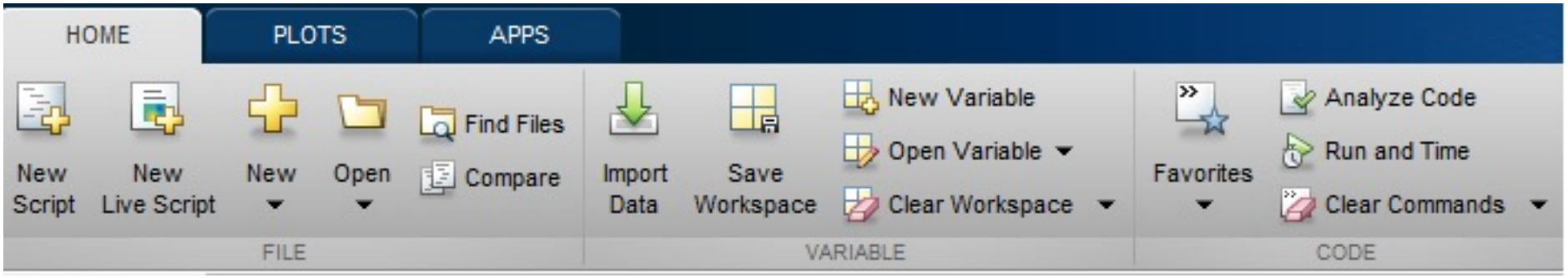
Get data, train SOM, show output

```
load iris_dataset
inputs=irisInputs;
d1=6;
d2=6;
net=selforgmap([d1,d2]);
[net,tr]=train(net,inputs);
outputs=net(inputs);
```

iris dataset: 3 classes,
50 records each,
4 attributes per record

Display options

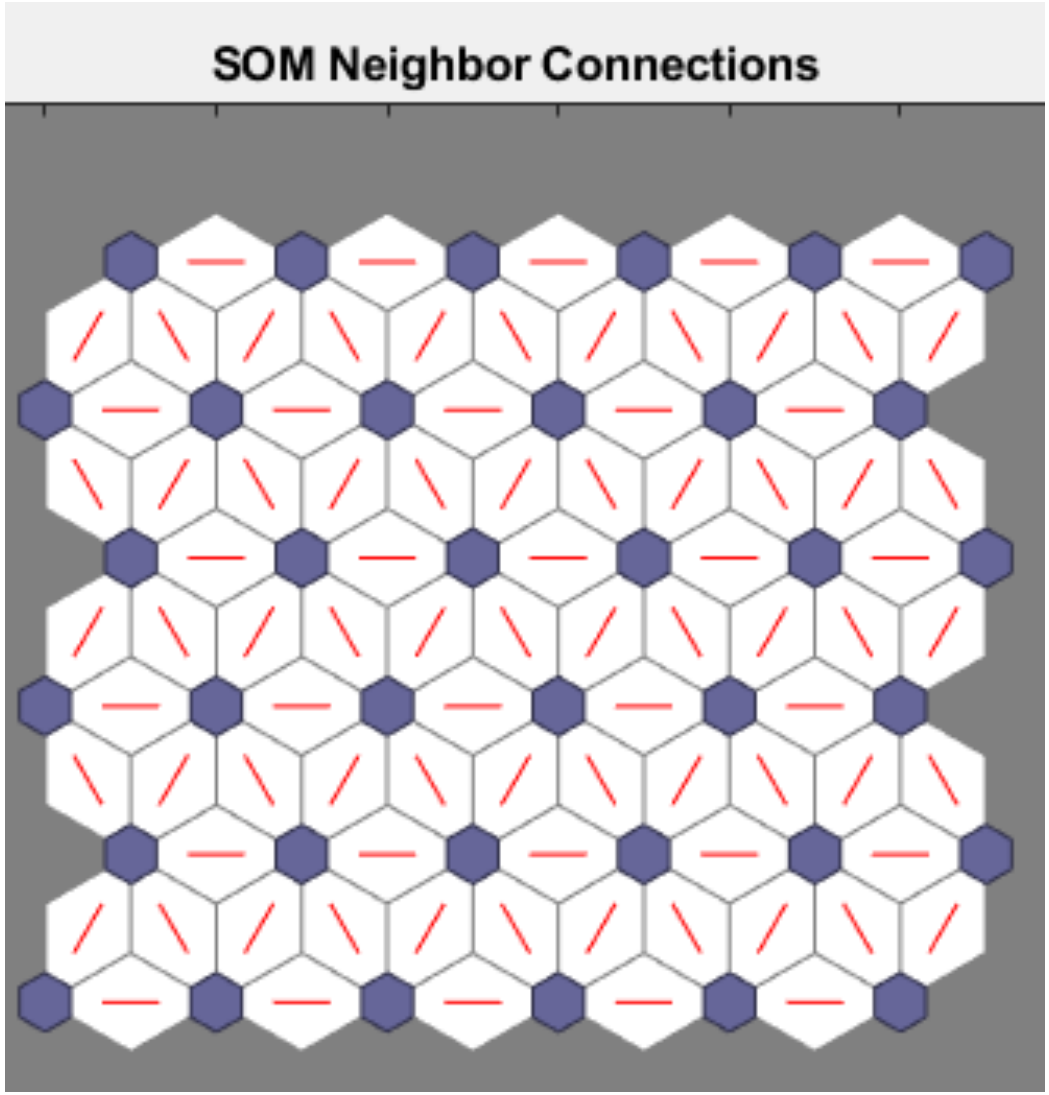
```
%view(net)
%figure, plotsomtop(net)
%figure, plotsomnc(net)
%figure, plotsomnd(net)
%figure, plotsomplanes(net)
%figure, plotsomhits(net,inputs)
%figure, plotsompos(net,inputs)
```



Click on New Script. Editor will open. Type the 7 lines of code below. Copy and paste into command window. Return.

```
load iris_dataset
inputs=irisInputs;
d1=6;
d2=6;
net=selforgmap([d1,d2]);
[net,tr]=train(net,inputs);
outputs=net(inputs);
```

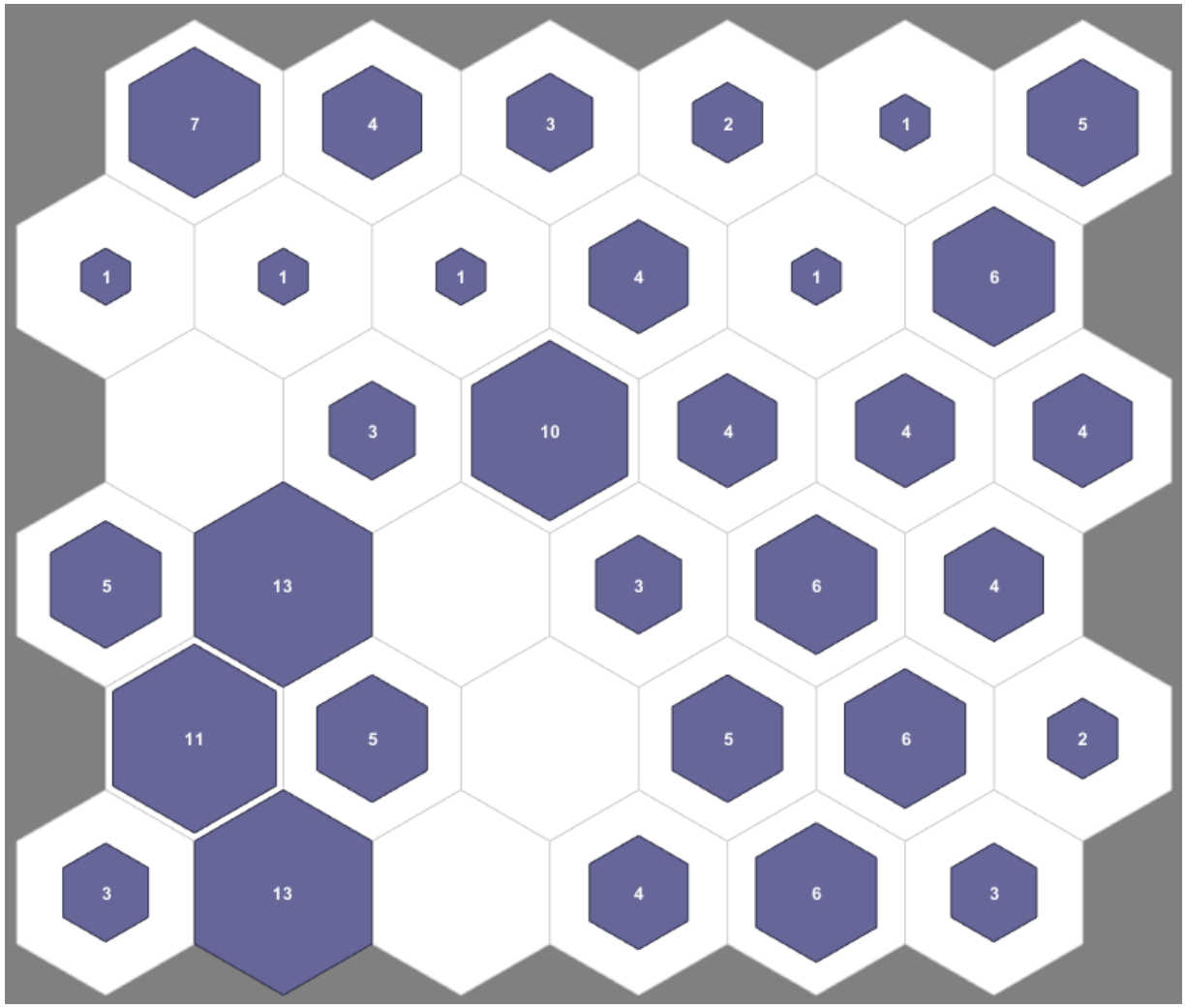
plotsomnc
6x6 hexagonal grid. Corner nodes: 2 or 3 neighbors, Edge nodes: 3 or 5 neighbors, Internal nodes: 6 neighbors.



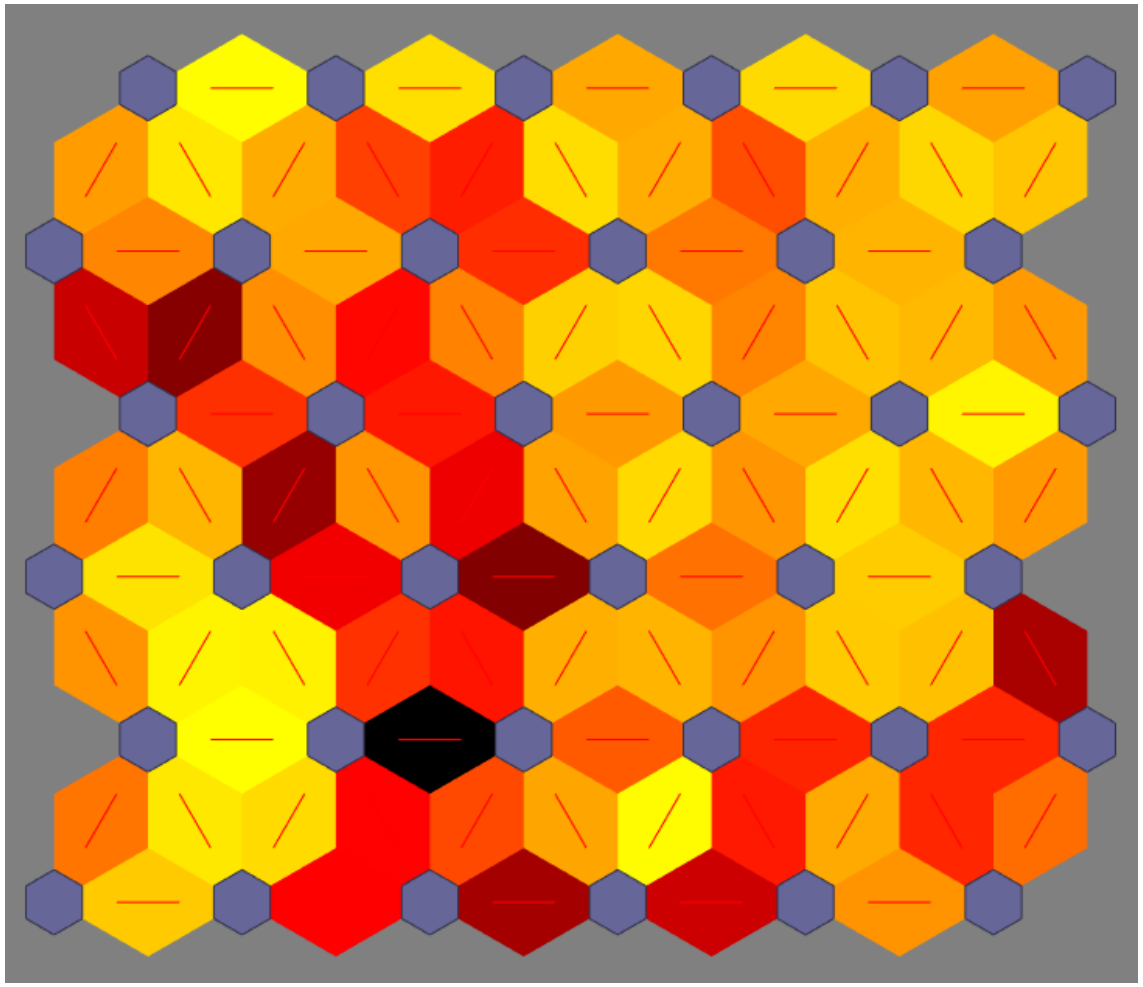
plotsomhits

Number of input instances for which a node is BMU

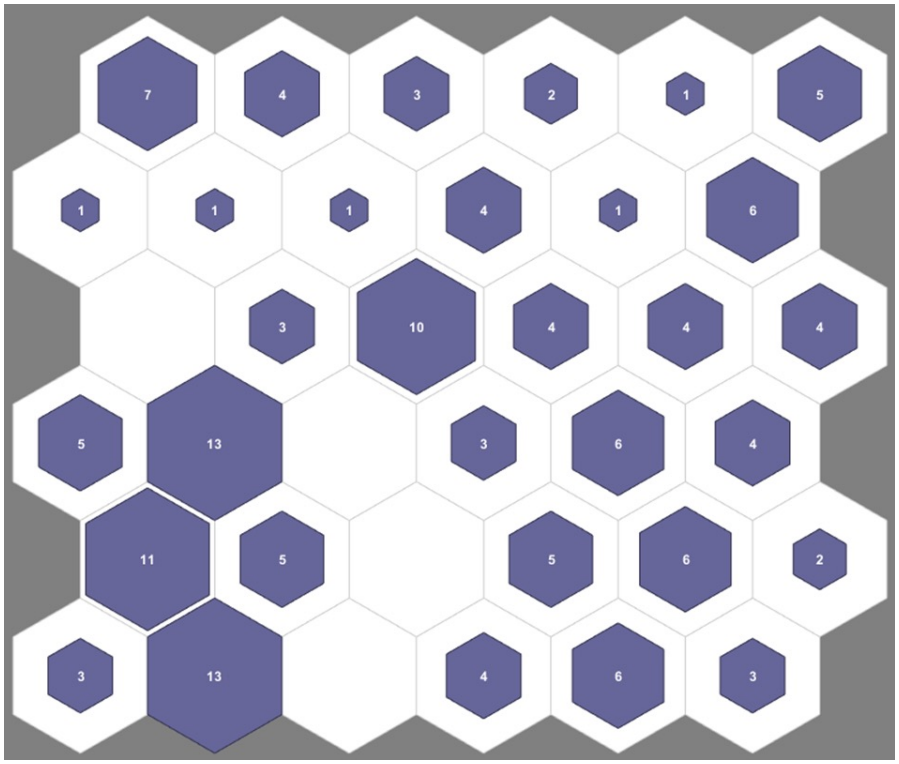
Your numbers may differ slightly.

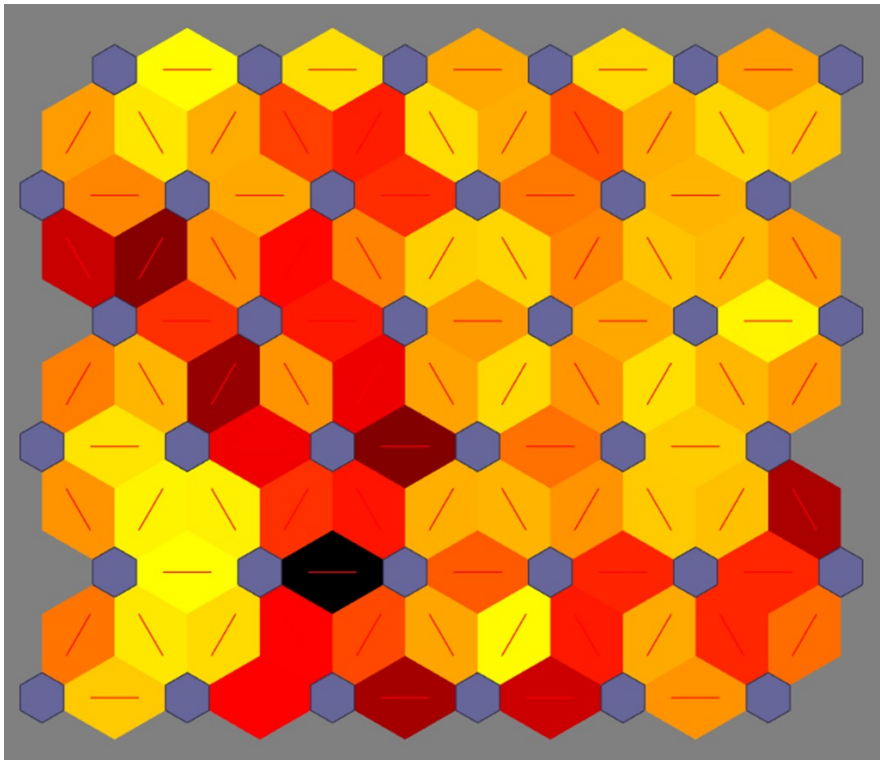


plotsomnd
UMAT displayed as heatmap on connections between nodes.
Bright colors highlight connections between similar nodes.



How many clusters are suggested by Hits and UMAT outputs? One boundary is clear. Labeled data has 3 classes. Is the larger region a cluster of 2 classes?





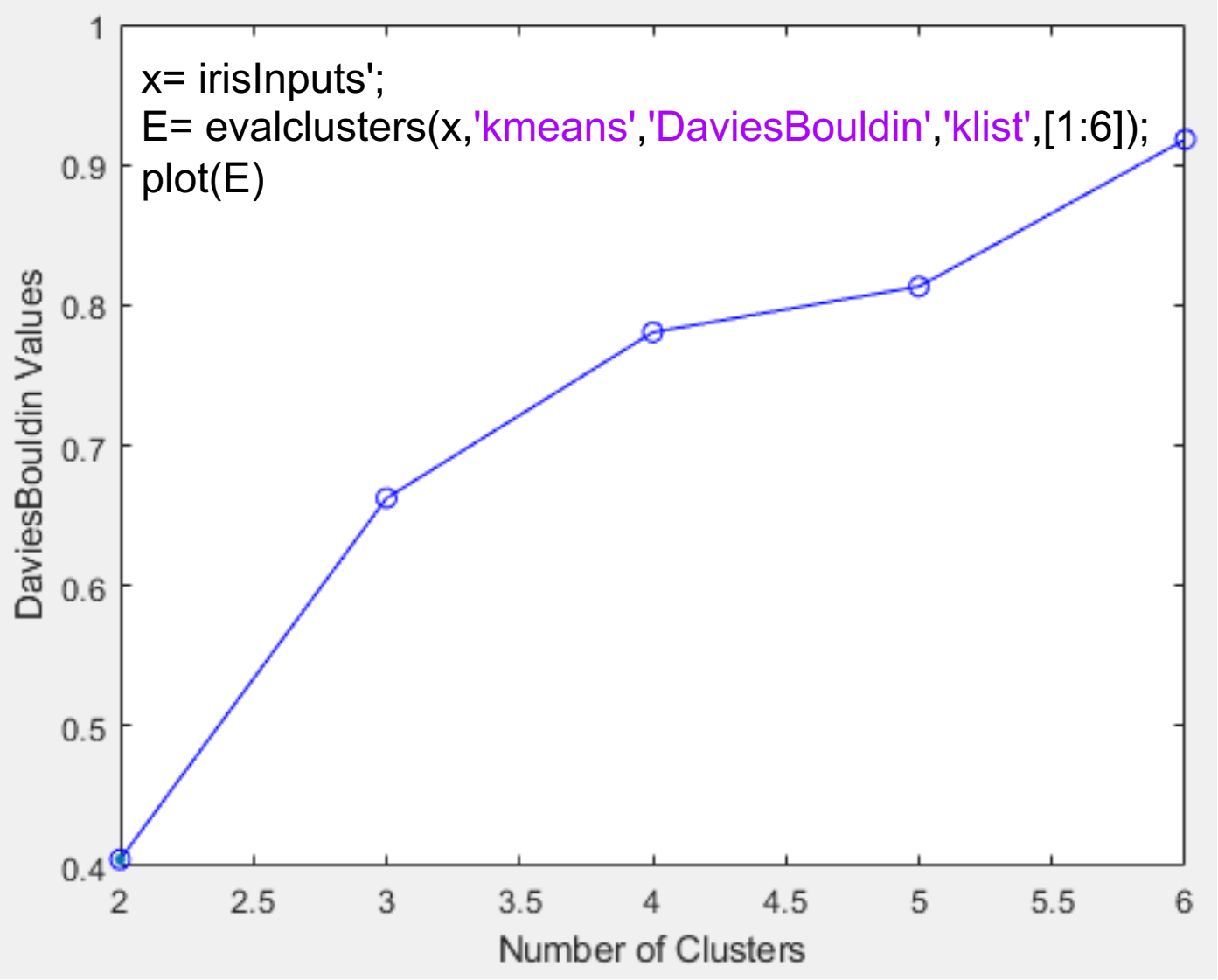
Are the suggestions of SOM consistent with K-means clustering of input data?

Add these lines to your script

```
x= irisInputs';
E= evalclusters(x,'kmeans','DaviesBouldin','klist',[1:6]);
plot(E)
```

Copy and enter them into command window at >>

BDI index for K-means. K=2 favored by minimum similarity.
SOM and K-means clustering of iris data are consistent.



Thanks for your attention.
Any questions?