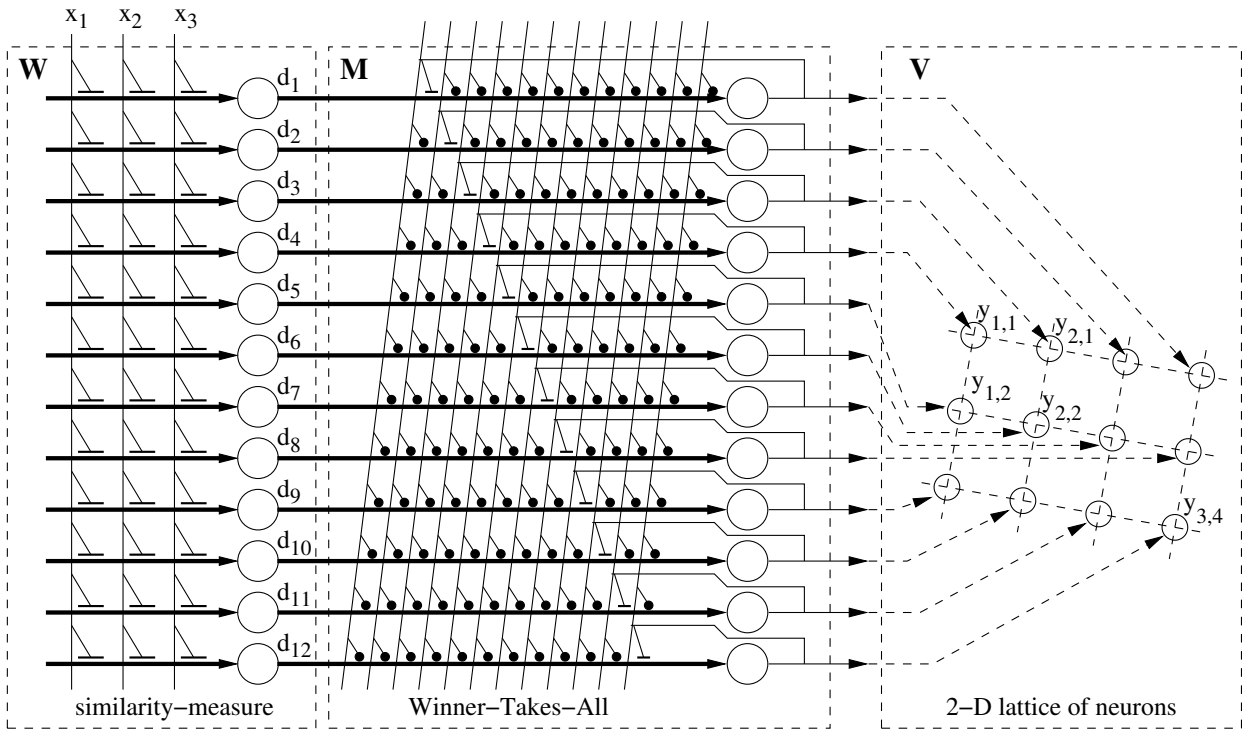
6.5 Self-Organizing Feature Maps

6.5.1 Structure of Self-Organizing Feature maps

Self-Organizing Feature Maps (SOFMs, or SOMs) also known as Kohonen maps or topographic maps were first introduced by von der Malsburg (1973) and in its present form by Kohonen (1982).

Self-Organizing Feature Maps are **competitive neural networks** in which, in addition, neurons are organized in an l-dimensional **lattice** (grid):

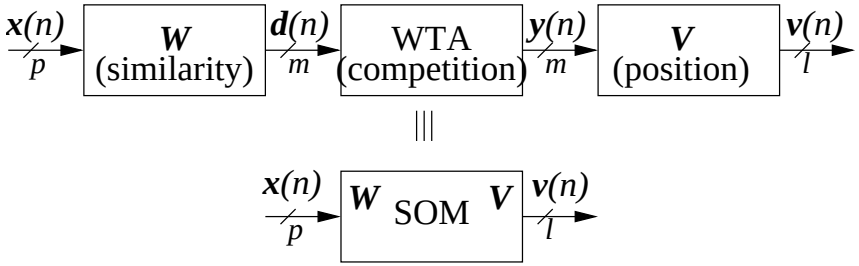


Example:

- $(p = 2)$-dimensional **input space**
- $m = 12$ neurons located on a 2-D $3 \times 4 = 12$ lattice.
- The **weight matrix** W is $m \times p$
- $(l = 2)$-dimensional **feature space**.
- Positions of neurons in the grid can be specified by a **positions matrix** V

Note the similarity measure layer, the competitive (WTA) layer and the neuronal grid.

General structure of a Self-Organizing Map:



The complete SOM is characterized by the following parameters:

p — dimensionality of the **input space**
l — dimensionality of the **feature space**
m — the total number of neurons
W — $m \times p$ weight matrix
V — $m \times l$ position matrix

We can say that for a given matrix of neuronal weights, W, the network **maps** an nth stimulus $\mathbf{x}(n)$ into a position of the **winner** $\mathbf{v}(n)$, that is, the neuron with the weight vector most similar to the stimulus.

$$\mathbf{v}(n) = g(\mathbf{x}(n); W) \; ; \quad \mathbf{v} \in \mathcal{R}^l \tag{6.11}$$

In subsequent considerations neurons will be identified either by their index $k = 1, \ldots, m$, or by their position vector $V(k, :)$ in the neuronal grid.

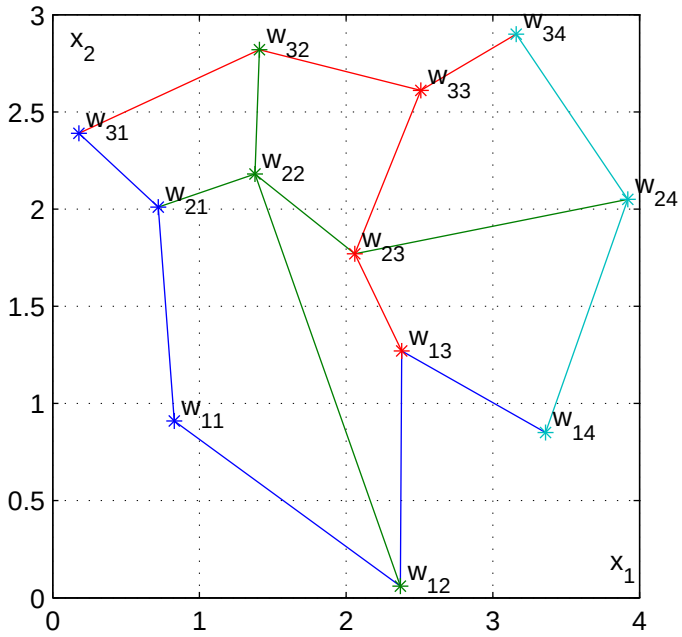
6.5.2 Feature Maps

If the dimensionality of the input (stimulus) space is low, typically $p = 1, 2, 3$, a **Feature Map** is a plot of synaptic **weights** in the **input space** in which weights of the neighbouring neurons are joined by lines or plane segments (patches).

Consider the following typical example of the weight and position matrices when both **input and feature spaces are two-dimensional**:

```
W = 0.83   0.91      V = 1  1
    0.72   2.01          2  1
    0.18   2.39          3  1
    2.37   0.06          1  2
    1.38   2.18          2  2
    1.41   2.82          3  2
    2.38   1.27          1  3
    2.06   1.77          2  3
    2.51   2.61          3  3
    3.36   0.85          1  4
    3.92   2.05          2  4
    3.16   2.90          3  4
```

A Feature map for 2-D input and feature map



The feature map describes the mapping of the input space into the future space.

Neurons linked with the grid are allocated to clusters of input data.

6.5.3 Learning Algorithm for Self-Organizing Feature Maps

The objective of the learning algorithm for a SOFM neural network is formation of a feature map which captures the essential characteristics of the p-dimensional input data and maps them on an l-D feature space.

The learning algorithm consists of two essential aspects of the map formation, namely, **competition** and **cooperation** between neurons of the output lattice.

Competition is implemented as in the competitive learning:

each input vector (stimulus) $\mathbf{x}(n)$ is compared with each weight vector from the weight matrix W and the position $V(k(n),:)$ of the winning neuron $k(n)$ is established.

For the winning neuron $k(n)$ the distance

$$d_k = |\mathbf{x}^T(n) - W(k(n),:)|$$

attains a minimum (as in MATLAB ':)' denotes all column elements of a matrix).

Cooperation All neurons located in a topological **neighbourhood of the winning neurons** $k(n)$ will have their weights updated

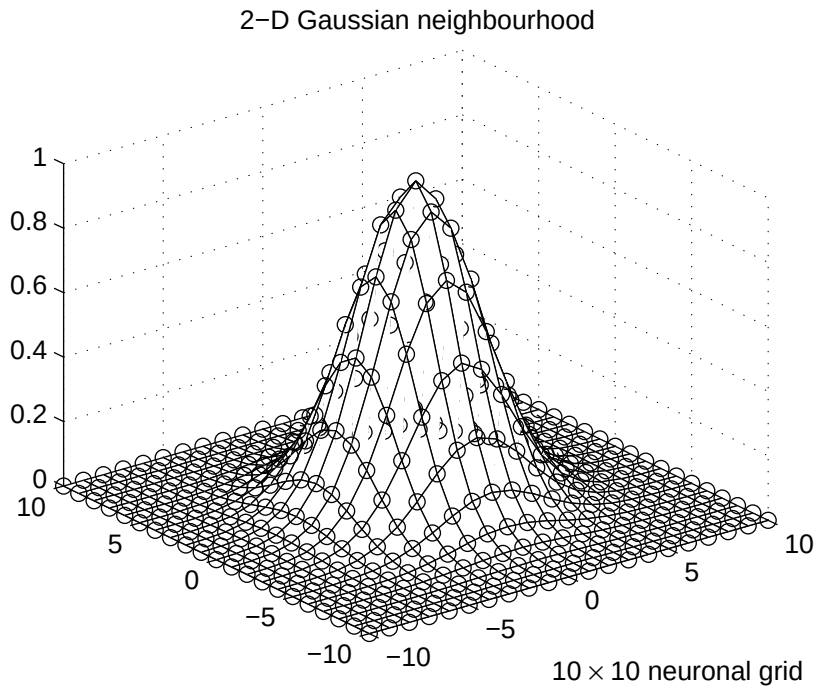
with a strength $\Lambda(j)$ related to their distance $\rho(j)$ from the winning neuron,

$$\rho(j) = |V(j,:) - V(k(n),:)| \quad \text{for} \quad j = 1, \ldots, m.$$

The **neighbourhood function**, $\Lambda(j)$, is usually an l-dimensional Gausssian function:

$$\Lambda(j) = \exp(-\frac{\rho^2(j)}{2\sigma^2})$$

where σ^2 is the variance parameter specifying the **spread** of the Gaussian function.



- The neighbourhood function should shrink when the learning progresses.
- Feature map formation is critically dependent on the learning parameters, namely, the learning gain, η, and the spread of the neighbourhood function specified for the Gaussian case by the variance, σ^2.
- In general, both parameters should be time-varying, and their values are selected experimentally.

Usually, the **learning gain**, η, should stay close to unity during the **ordering phase** of the algorithm which can last for, say, 1000 iteration (epochs).
After that, during the **convergence phase**, should be reduced to reach the value of, say, 0.1.

The **spread**, σ^2, of the neighbourhood function should initially include all neurons for any winning neuron and during the ordering phase should be slowly reduced to eventually include only a few neurons in the winner's neighbourhood.
During the convergence phase, the neighbourhood function should include only the winning neuron.

Details of the SOFM learning algorithm

The complete algorithm can be described as consisting of the following steps

1. Initialise:
   (a) the weight matrix W with a random sample of m input vectors.
   (b) the learning gain and the spread of the neighbourhood function.
2. for every input vector, $\mathbf{x}(n), \quad n = 1, \ldots, N$:
   (a) Determine the winning neuron, $k(n)$, and its position $V(k,:)$ as

$$k(n) = \arg\min_j |\mathbf{x}^T(n) - W(j,:)|$$

   (b) Calculate the neighbourhood function

$$\Lambda(n,j) = \exp(-\frac{\rho^2(j)}{2\sigma^2})$$

   where

$$\rho(j) = |V(j,:) - V(k(n),:)| \quad \text{for} \quad j = 1, \ldots, m.$$

(c) Update the weight matrix as

$$\Delta W = \eta(n) \cdot \Lambda(n) \cdot (\mathbf{x}^T(n) - W(j,:))$$

All neurons (unlike in the simple competitive learning) have their weights modified with a strength proportional to the neighbourhood function and to the distance of their weight vector from the current input vector (as in competitive learning).

The step (2) is repeated E times, where E is the number of epochs.

3. (a) During the ordering phase, shrink the neighbourhood until it includes only one neuron:

$$\sigma^2(e) = \frac{\sigma_0^2}{e}$$

where e is the epoch number and σ_0^2 is the initial value of the spread (variance).

(b) During the convergence phase, "cool down" the learning process by reducing the learning gain. We use the following formula to reduce η

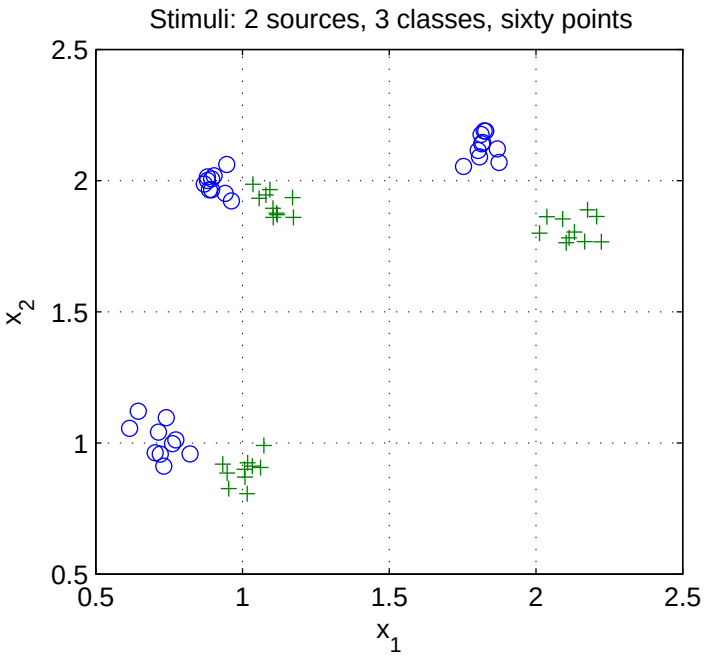
$$\eta(e) = \frac{\eta_0}{1 + \eta_p e}$$

where η_0 is the initial value of the learning gain, and η_p is selected so that the final value of the learning gain, reaches the prescribed value, $\eta(E) = \eta_f$.

6.5.4 Example of a Self-Organizing Feature Map formation

- In this example we consider two-dimensional stimuli generated by two sources marked by by o and + respectively.
- We use three groups of data each containing 10 stimuli (i.e. points in a two-dimensional space) in each source, sixty points all together.
- The sources can be thought of as producing, e.g., two dialects of a very limited protolanguage, each with three protophonemes.

Training stimuli:



- We can imagine that a source is a representation of a parent of a child pronouncing three phonemes in ten slightly different ways.
- The parallel is far from perfect but might be helpful for a conceptual understanding of the map formation and interpretation.
- Real sensory stimuli, like the phonemes of speech are of course larger in number and dimension.

The next step is to define details of the neural network and initialise the weights.

```
% Specification of the neuronal lattice
p = 2 ;      % dimensionality of afferents
m = [3 3] ;  % neuronal lattice

%  formation of the neuronal position matrix
[V2, V1] = meshgrid(1:m(2), 1:m(1)) ;
V = [V1(:), V2(:)] ;

W = % random initialization around the mean of stimuli
```

Before we start the process of self-organization, that is, mapping neurons to stimuli, we set first the relevant parameters:

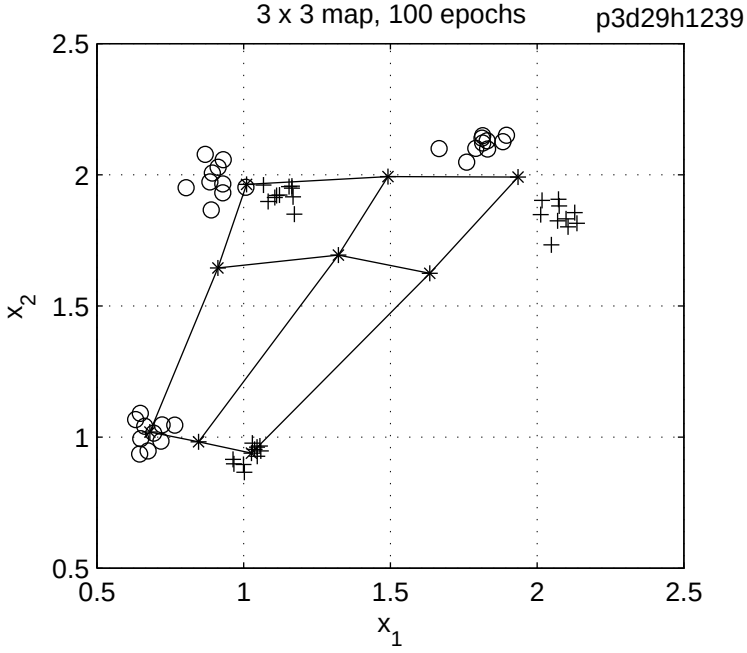
```
sig2 = 50 ;   % 2 sigma^2 initial spread of the neighbourhood function
eta0 = 0.1 ;  % The initial learning gain
etap = 0.02 ; % learning gain reducing factor
nepochs = 100;  % the number of epochs in each simulations
```

Finally, the script for the learning loop follows. We will plot the feature map at the end of each epoch.

```
%  Map formation
 for epch = 1:nepochs    %  the epoch loop
   eta = eta0/(1+etap*epch) ;  % gain is reduced for every epoch

   for n=1:N     %  stimulus loop
     % training data is applied in a random order
     xn = X(:, round((N-1)*rand)+1) ; % a new stimulus (datapoint)

     %  WTA: distance between the datapoint and weight vectors
     xW = xn(1:2,onesm) - W' ;
```

```
    % grid coordinates of the winning neuron V(kn, :)
    % and the difference between the datapoint and
    % the weight vector of the winner
    [TtlDif kn] = min(sum(xW.^2)) ;  vkn = V(kn, :) ;

    % squared distance from the winning neuron
    rho2 = sum(((vkn(onesm, :) - V).^2), 2) ;
    % Gaussian neighbourhood function, Lambda, is
    %        centered around winning neuron
    Lambda = exp(-rho2*epch/sig2) ;
    % Kohonen learning law
    W = W + eta*Lambda(:,ones(p,1)).*xW';
end  % stimulus loop
```

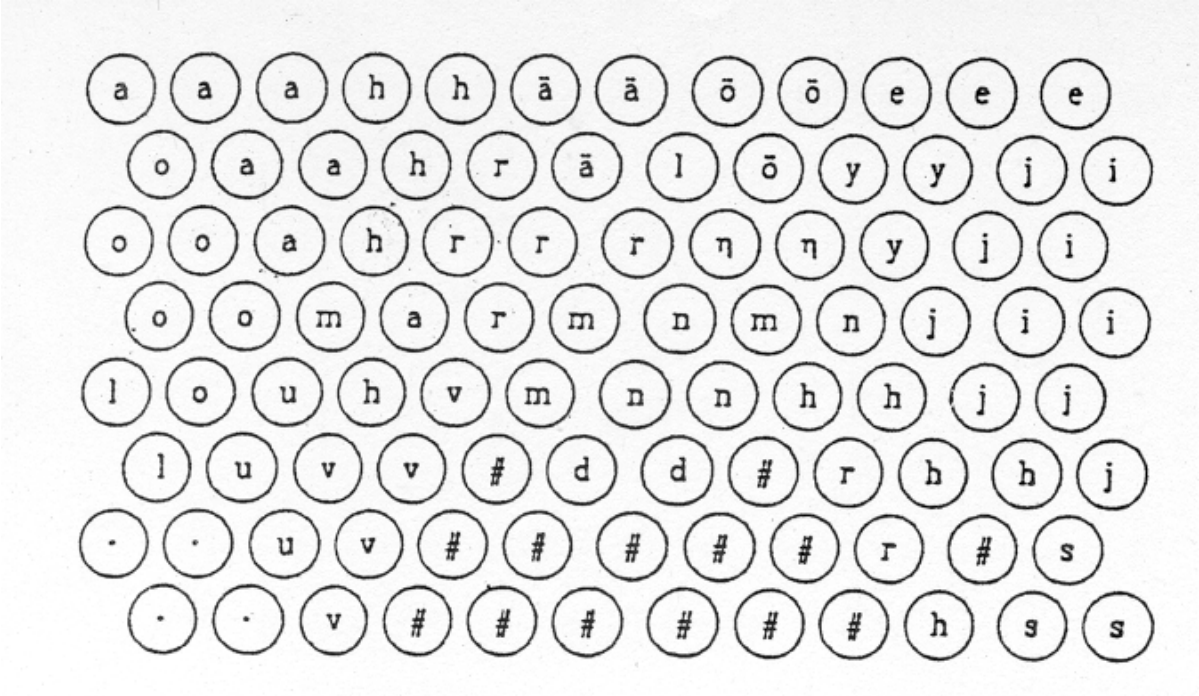
- Self-organizing map of 3×3 lattice of neurons
- The resulting feature map assigns nine neurons to three pairs of stimuli clusters (60 stimuli),
- Note that neurons are assigned either to data clusters, or to groups of clusters to have the best approximation of data distribution.
- Three neurons in the centre of the grid seem to be not assigned to any group.



6.5.5 Higher dimension Kohonen maps — Categorization of data

- If the dimensionality of data is high, practically higher than 3, we cannot visualise the feature map in the input space.
- Assuming that the dimensionality of the feature (neuronal) space is 2, the feature map can be visualised on the neuronal grid, assigning a label of the closest data cluster to each neuron.

- Classical example of a hexagonal SOM mapping Finnish phonemes to a grid of 5×12 neurons.
- The neurons, shown as circles, are labeled with the symbols of the phonemes with which they made the best match.
- A phoneme is typically described by 12 to 14 parameters describing its frequency components.



from: T. Kohonen: *Self-Organizing Maps*. Springer, 2001

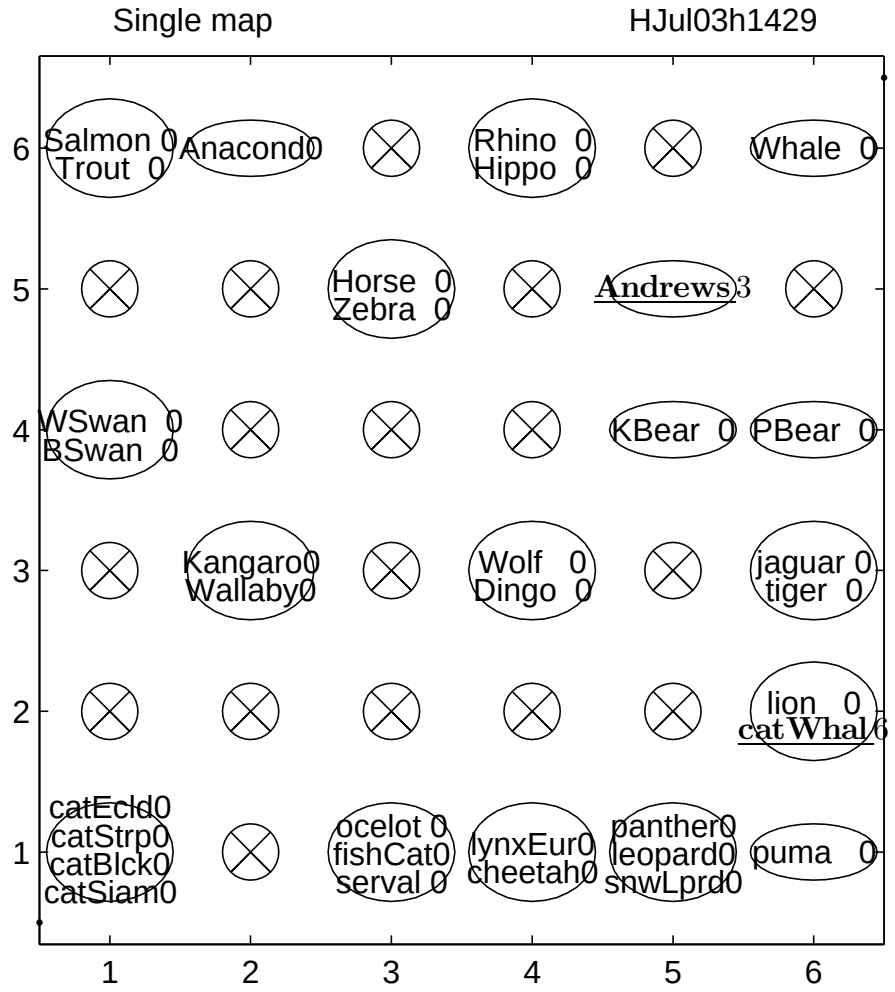
Categorization of animals — Example

- Our animal kingdom consists of 32 animals each characterized by 18-dimensional data.
- The animals are listed below sorted according to their weight:
  Grey whale, Hippopotamus, White rhinocerous, Kodiak bear, Polar bear, Grevy's zebra, Przewalski's horse, Tiger, Lion, Anaconda, Jaguar, Puma or cougar, Panther, Leopard, Snow Leopard, Canis lupus (Wolf), Atlantic salmon, Cheetah, Grey western kangaroo, Eurasian lynx, Rainbow trout, Dingo, Swamp wallaby, Serval, Ocelot, Fishing cat, Mute (white) swan, Black swan, Domestic cat (even coloured), Domestic cat (striped), Domestic cat (black), Siamese cat. (Half of the animals are variety of cats.)
- In addition, for testing of the generalization of the multi-map structure we use two unusual animals, one being a domestic cat weighing two tonnes ('catWhale'), the other, *Andrewsarchus mongoliensis*, is an extinct cloven-hoofed 1-tonne carnivour.
- The features chosen to characterize these animals are as follows:

x_1	log(weight)
$x_2 \in \{1, 2, 3\}$	food (herbivores, omnivores, carnivores)
$x_3, x_4, x_5, x_6 \in \{0, 1\}$	locomotion (fins, wings, two legs, four legs)
$x_7 \in \{0, 1\}$	equipped with hooves (perissodactyls) or cloven hooves (artiodactyls)
$x_8 \in \{0, 1\}$	equipped with claws
$x_9 \in \{0, 1\}$	equipped with other feet
$x_{10} \in \{1, 2, 3\}$	cover (fur, feathers and scales)
$x_{11} \in \{0, 1\}$	colour black
$x_{12} \in \{0, 1\}$	colour white
$x_{13} \in \{0, 1\}$	even coloured
$x_{14} \in \{0, 1\}$	spotted
$x_{15} \in \{0, 1/4\}$	striped
$x_{16} \in \{2, 4\}$	facial feature (short faced, long faced)
$x_{17} \in \{1, 2, 3\}$	aquatic
$x_{18} \in \{1, 2, 3\}$	social behaviour (single living, pair living, group living).

- After learning that consisted of 400 epochs of presentations of input vectors to the network, the following map has been obtained:

- The haviest animals are grouped at the top right part of the map, the lightest, the house cats, are at the bottom left corner.
- All type of cats are grouped in the bottom right part of the maps.
- The two animals that were presented to the map without any prior learning, the fictitious catwhale and the extinct Andrewsarchus are represented in a reasonable way:
- The catwhale is close to the lion which is a good compromise for an animal with the conflicting features of a house cat except for the weight of a rhino.
- The Andrewsarchus is closed to Hippopotamus and White rhinocerous, which is reasonable, given the features by which it was described.



Un-assigned neurons are located between the neurons that are closest to a given animal or a group of animals.