UNIVERSITÀ DEGLI STUDI DI MILANO

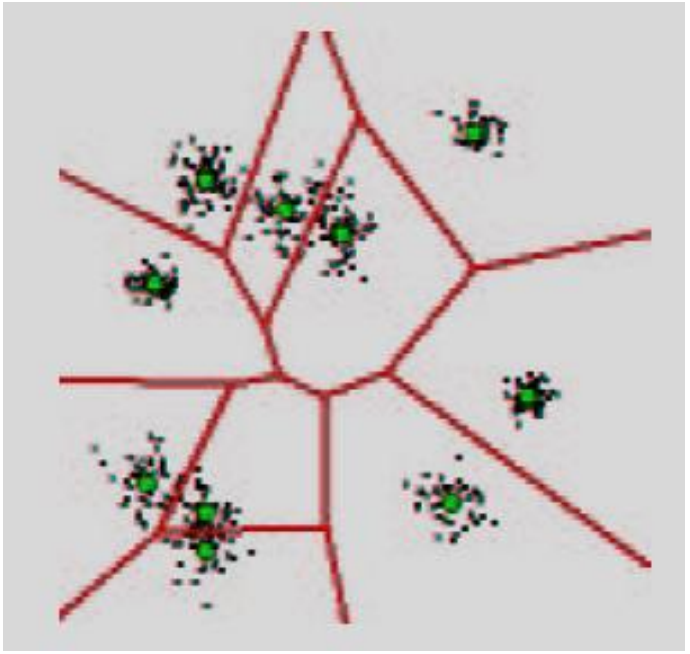
Clustering Using Neural Networks

Ruggero Donida Labati

Dipartimento di Tecnologie dell'Informazione
via Bramante 65, 26013 Crema (CR), Italy
ruggero.donida@unimi.it

Clustering

- Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).



Useful definitions

- In pattern recognition and machine learning, a **feature vector** is an n-dimensional vector of numerical features that represent some object.

- In topology and related branches of mathematics, a **topological space** is a set of points, along with a set of neighbourhoods for each point, that satisfy a set of axioms relating points and neighbourhoods.

Some clustering methods

- k-means
- Kohonen Self Organising Feature Maps (SOMs)
- Neural Gas
- Growing Neural Gas

k-means

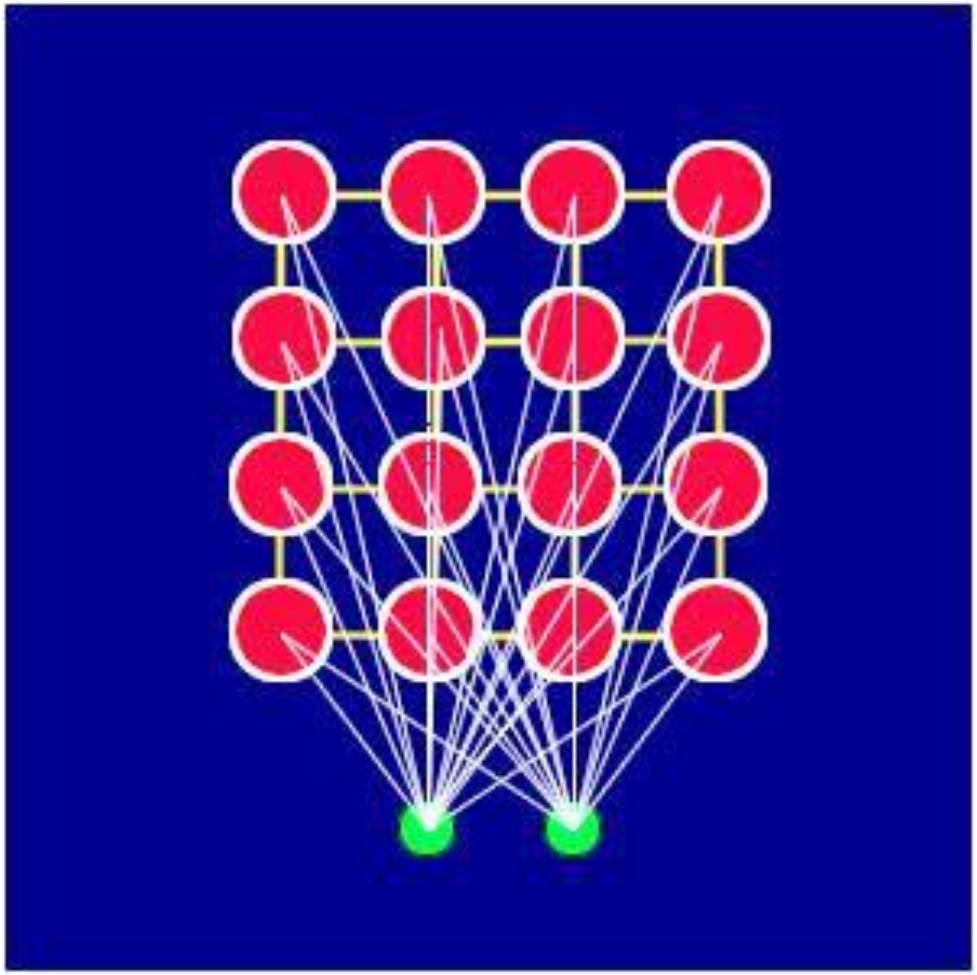
$U \in \mathbb{N}^{K \times N}$ ← Matrix representing the patitions $u_{i,j}$ (binary values that define if the sample *j* pertain to the cluster *i*)

$C = C_1, C_2, \ldots, C_K$ ← Vector of the centroids (*k* elements)

$$V(U, C) = \sum_{i=1}^{K} \sum_{X_j \in P_i} ||X_j - C_i||^2$$

- Iterative method
  1. Create random values for U_v and C_v
  2. Compute U_n that minimizes $V(U, C_v)$
  3. Compute C_n that minimizes $V(U_v, C)$
  4. If the algorithm does not converge, $U_v = U_n$, $C_v = C_n$, step 2

SOMs



- Each node has a specific topological position (an *x*, *y* coordinate in the lattice) and contains a vector of weights of the same dimension as the input vectors. That is to say, if the training data consists of vectors, *V*, of *n* dimensions:
- $V_1, V_2, V_3 \ldots V_n$
- Then each node will contain a corresponding weight vector *W*, of n dimensions:
- $W_1, W_2, W_3 \ldots W_n$

The learning algorithm (1/4)

1. Initializing The Weights
2. Calculating the Best Matching Unit
3. Determining the Best Matching Unit's Local Neighbourhood
4. Adjusting the Weights

The learning algorithm (2/4)

- Initializing The Weights
  - Prior to training, each node's weights must be initialized. Typically these will be set to small standardized random values. The weights in the SOM from the accompanying code project are initialized so that $0 < w < 1$.

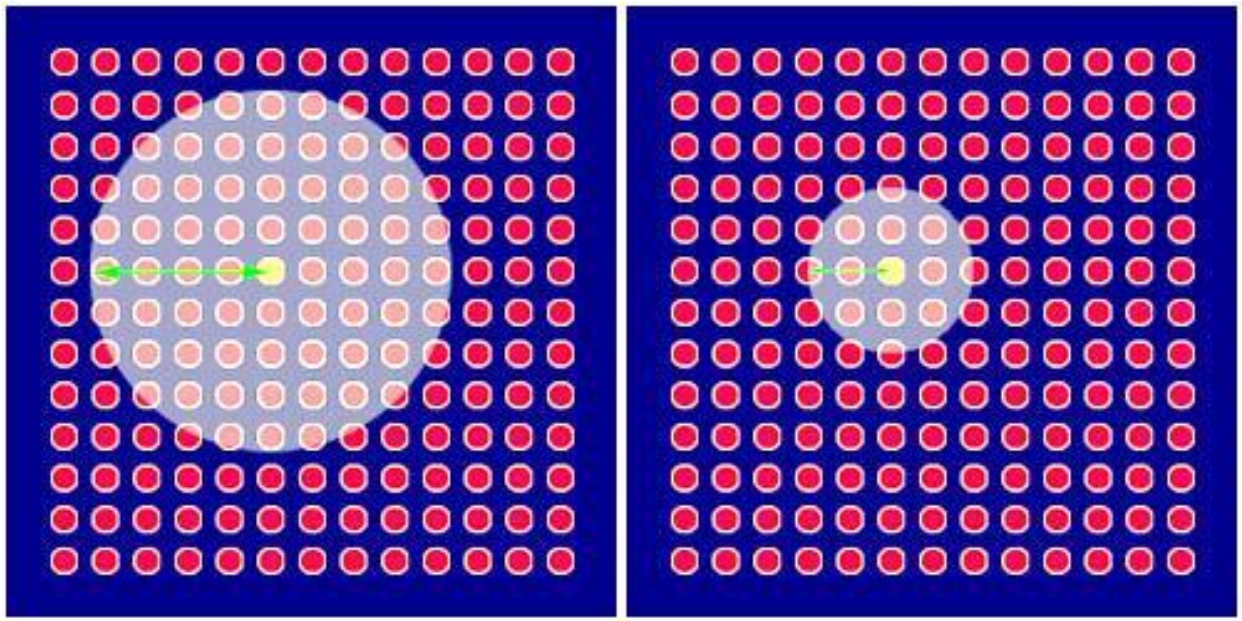
- Calculating the Best Matching Unit (BMU)

$$Dist = \sqrt{\sum_{i=0}^{i=n} (V_i - W_i)^2}$$

The learning algorithm (3/4)

- Determining the Best Matching Unit's Local Neighbourhood

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \qquad t = 1,2,3, \ldots$$



The learning algorithm (4/4)

- Adjusting the Weights
  - Every node within the BMU's neighbourhood (including the BMU) has its weight vector adjusted according to the following equation:

$$W(t+1) = W(t) + L(t)\Theta(t)(V(t) - W(t))$$

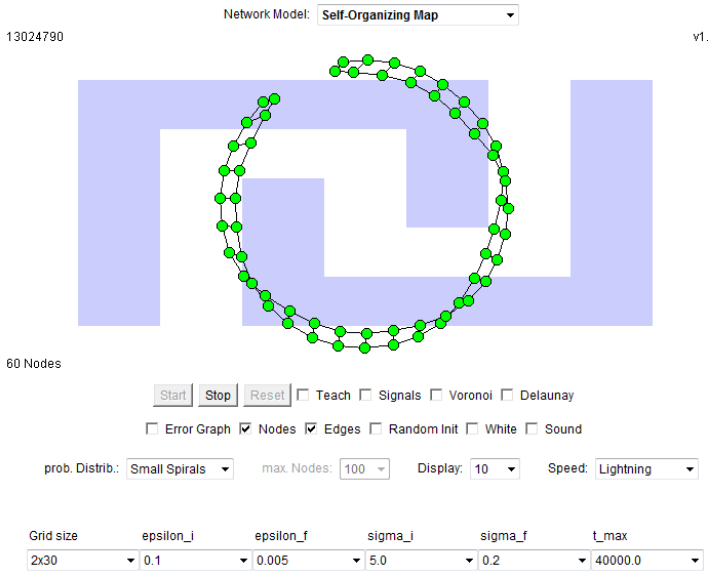
where

$$L(t) = L_0 \exp\left(-\frac{t}{\lambda}\right) \qquad t = 1,2,3,\ldots$$

$$\Theta(t) = \exp\left(-\frac{dist^2}{2\sigma^2(t)}\right) \qquad t = 1,2,3,\ldots$$

SOMs: demo

- DemoGNG (Version 1.5)



  - http://sund.de/netze/applets/gng/full/GNG-U_0.html
  - http://www.demogng.de/JavaPaper/node1.html

SOMs in Matlab

- Basic concepts
  - http://www.mathworks.it/help/nnet/gs/cluster-data-with-a-self-organizing-map.html
- Parameters
  - http://www.mathworks.it/help/nnet/ug/cluster-with-self-organizing-map-neural-network.html

Exercise

- Iris clustering
  - http://www.mathworks.it/help/nnet/examples/iris-clustering.html
- Iris clustering

```
%Step1 : load
load iris.dat;
setosa = iris((iris(:,5)==1),:);       % data for setosa
versicolor = iris((iris(:,5)==2),:);   % data for versicolor
virginica = iris((iris(:,5)==3),:);    % data for virginica
obsv_n = size(iris, 1);              % total number of observations
P = iris(:,1:4)';

%Step2 : plot in 2D
Characteristics = {'sepal length','sepal width','petal length','petal width'};
pairs = [1 2; 1 3; 1 4; 2 3; 2 4; 3 4];
h = figure('Name','Dati iniziali');
for j = 1:6,
   x = pairs(j, 1);
   y = pairs(j, 2);
   subplot(2,3,j);
   plot([setosa(:,x) versicolor(:,x) virginica(:,x)],...
       [setosa(:,y) versicolor(:,y) virginica(:,y)], '.');
   xlabel(Characteristics{x});
   ylabel(Characteristics{y});
end
```

```
ny = 1;
nx = 3;
net = selforgmap([ny nx]);
nCluster = ny * nx;
view(net)

[net,tr] = train(net,P);
nntraintool

y = net(P);
cluster_index = vec2ind(y);

figure
for j = 1:3
   subplot(1,3,j);
   hist( cluster_index(find(iris(:,5)==j)));
   xlabel('cluster')
   ylabel('n. samples')
   title(sprintf('class %i', j))
 end
```

Neural Gas (1/2)

- Given a probability distribution $P(x)$ of data vectors x and a finite number of feature vectors wi, i=1,...,N.
- With each time step t a data vector randomly chosen from P is presented. Subsequently, the distance order of the feature vectors to the given data vector x is determined. i_0 denotes the index of the closest feature vector, i_1 the index of the second closest feature vector etc. and i_{N-1} the index of the feature vector most distant to x. Then each feature vector (k=0,...,N-1) is adapted according to

$$w_{ik}^{t+1} = w_{ik}^{t} + \epsilon \cdot e^{-\frac{k}{\lambda}} \cdot \left(x - w_{ik}^{t}\right)$$

- with ε as the adaptation step size and λ as the so-called neighborhood range. ε and λ are reduced with increasing t. After sufficiently many adaptation steps the feature vectors cover the data space with minimum representation error.
- The adaptation step of the neural gas can be interpreted as gradient descent on a cost function

Neural Gas (2/2)

- Test

```
% Input data
circle = load('local_sphere_shell');
P = circle.Data;

figure
plot3(P(1,:), P(2,:), P(3,:), '.')
title('Input');
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis square

% neural gas
nIt       = 100;
Ei        = .05;
Ef        = .005;
Li        = 350;
Lf        = .5;
nCluster = 500;
Y = ngas(P', nCluster, nIt, Li, Lf, Ei, Ef);
Y = Y';
```

```
% Plot the results
figure
plot3(P(1,:), P(2,:), P(3,:), '.')
hold on
plot3(Y(1,:), Y(2,:), Y(3,:), '.r')
title('Output');
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis square
```

- Function ngas() was provided by

  Prof. Stefano Ferrari

Growing Neural Gas (GNG)

- B. Fritzke, " A growing neural gas network learns topologies," Advances in Neural Information Processing Systems 7 (NIPS'94), MIT Press, Cambridge, MA, pp. 625-632, 1995 (http://web.cs.swarthmore.edu/~meeden/DevelopmentalRobotics/fritzke95.pdf).

- Toolbox
  - http://www.mathworks.it/matlabcentral/fileexchange/43665-unsupervised-learning-with-growing-neural-gas--gng--neural-network

Suggested readings

- A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
- A. K. Jain, R.P.W Duin, J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.22, no.1, pp.4,37, Jan 2000.
- T. Kohonen, “The Self-Organizing map”, *Proceedings of the IEEE*, 78, pp. 1464-1480, 1990.
- B. Fritzke, “ A growing neural gas network learns topologies,” *Advances in Neural Information Processing Systems 7 (NIPS’94)*, MIT Press, Cambridge, MA, pp. 625-632, 1995.