COMPUTATIONAL INTELLIGENCE

Unsupervised Learning and Self-Organizing Maps

SOM

AGH

Adrian Horzyk

Unsupervised Learning

Unsupervised learning is a kind of adaptation algorithms (learning methods) that has not defined any goal of learning. We say that training data are unlabeled.

Training data contain only input data without:

- desired class information for classification tasks (supervised learning),
- desired function values for various regression and approximation tasks (supervised learning),
- expert judgment on the quality of the computed outputs (reinforcement learning).

Unsupervised learning is typically used for:

- initial features extraction in various deep learning algorithms and networks,
- clustering tasks which group training data into some number of clusters, sets, groups, etc.,
- anomaly detection, discriminating outliers, which are not grouped because of the lack of their similarity to the other samples.

Clustering

Clustering is an operation which divides data into clusters (groups, subsets).
It is a task of grouping a set of objects after their similarities that can be variously defined and achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. The primary task is to define constraints and conditions which point out different groups of objects.

We can distinguish various clustering methods, e.g.:

- K-means
- Mixture models
- Hierarchical clustering
- SOM

Clusters and Clustering

Clusters can be created using different models:

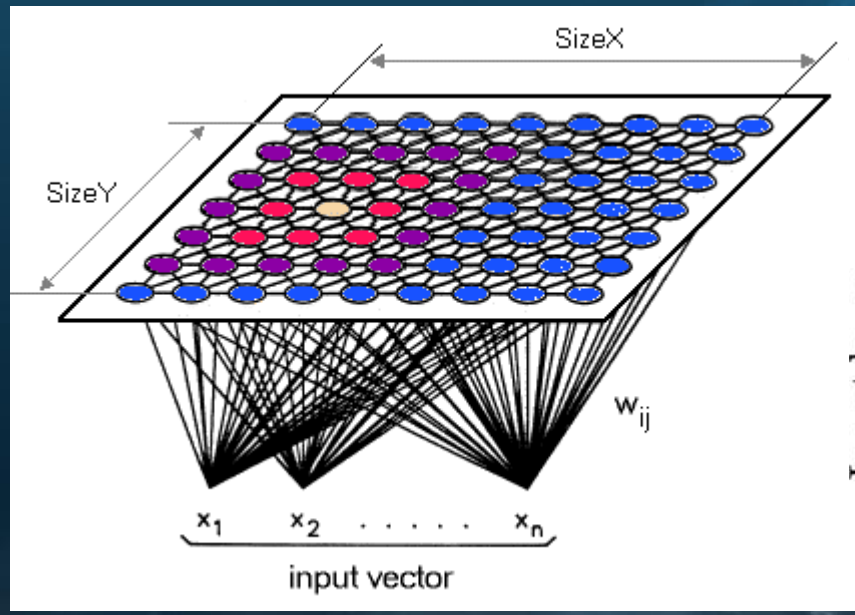
- centroid – represents each cluster by a single mean vector (e.g. k-means algorithm),
- connectivity – are based on distance connectivity (e.g. hierarchical clustering),
- density – defines clusters as connected dense regions in the data space (e.g. DBSCAN and OPTICS),
- bi-clustering – uses both cluster members and relevant attributes,
- grouping – does not provide a refined model for their results and just provide the grouping information,
- graph-based – looking for a clique, i.e. a subset of nodes in a graph such that every two nodes in this subset are connected by an edge can be considered as a prototypical form of the cluster (e.g. HCS clustering algorithm),
- statistical distribution – such as multivariate normal distributions (e.g. expectation-maximization algorithm).

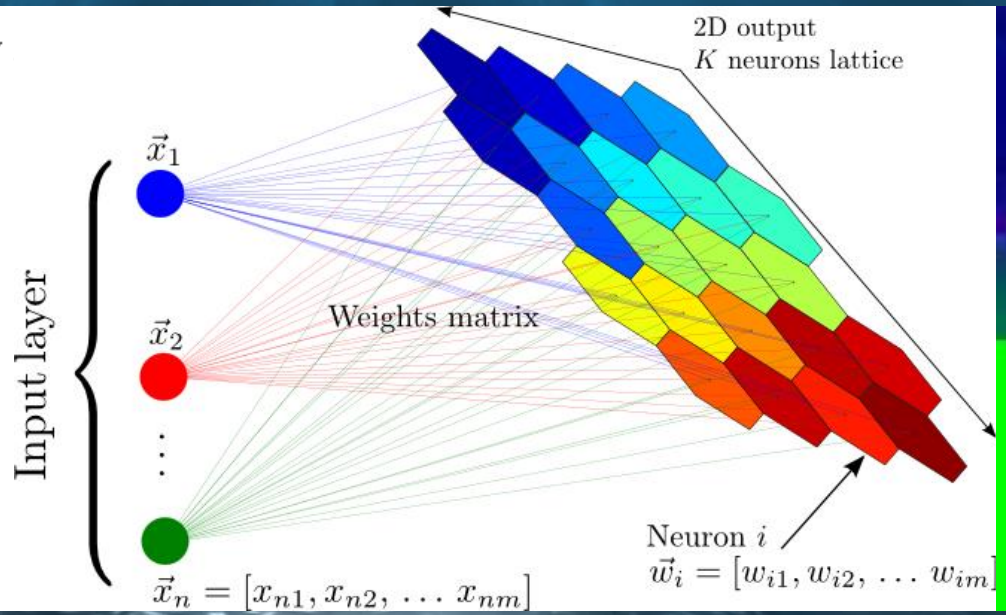
Clusters can be hard/sharp or soft/fuzzy, i.e. each object belongs to each cluster to a certain degree.

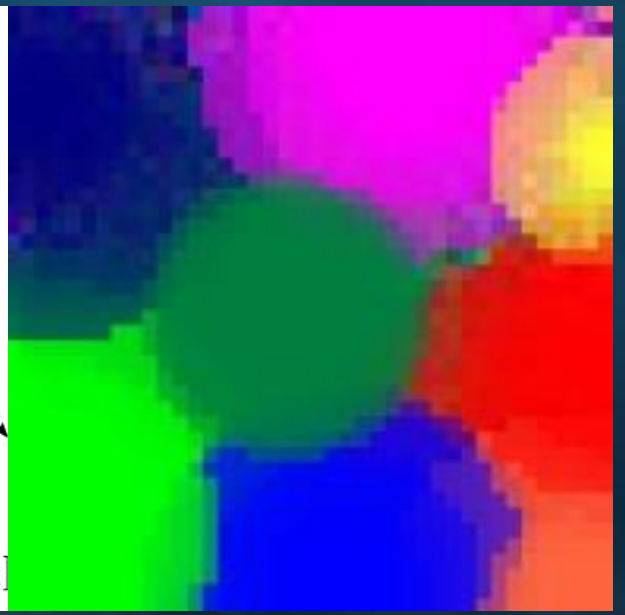
Self-Organizing Maps – SOM

Self-Organizing Maps (SOM) introduced by prof. T. Kohonen are networks consisting of nodes that can resemble neurons. However, the connection weights of these nodes do not weigh the input signals but try to reproduce them, and nodes compute the distance to the input vector, not a weighted sum!

The nodes are placed in the defined grid of nodes and have neighbors.

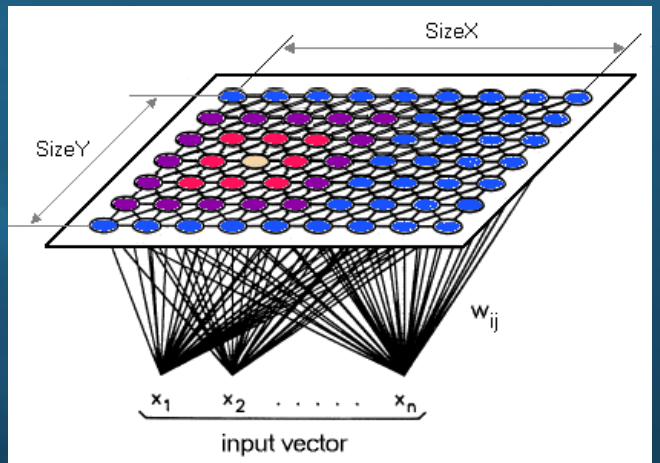


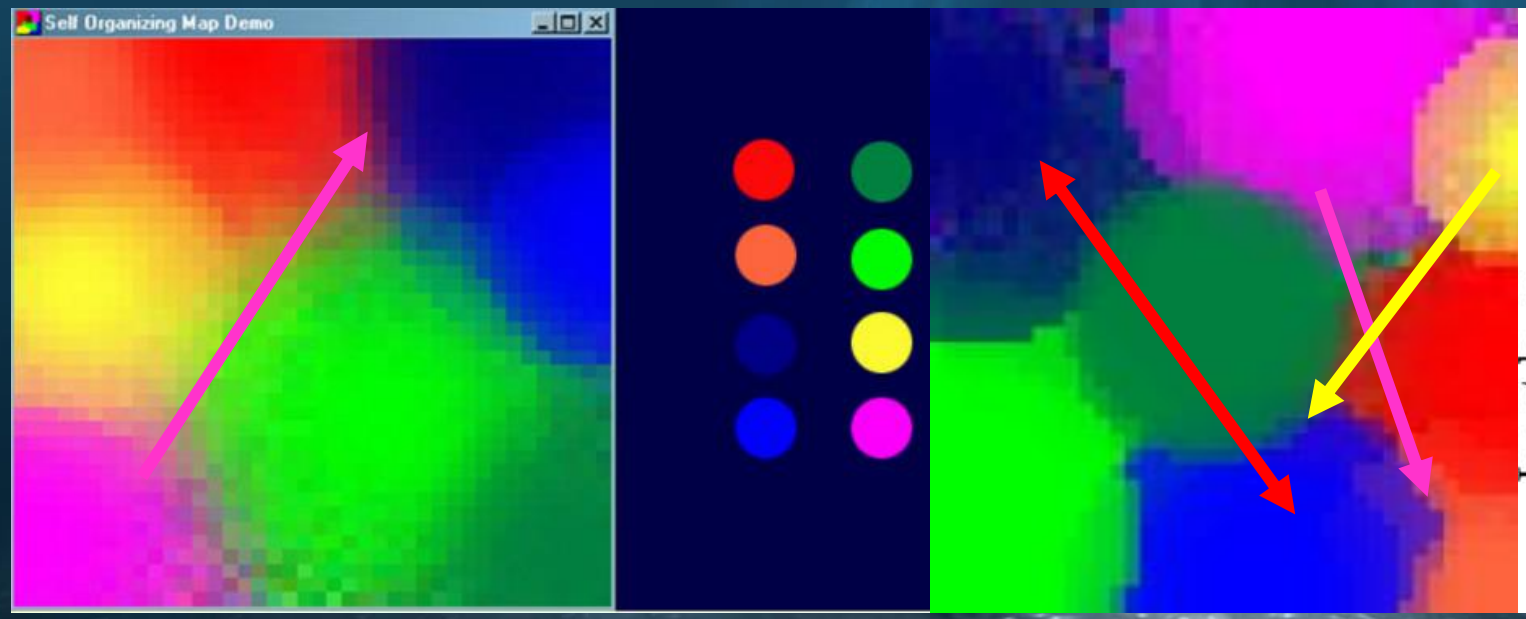


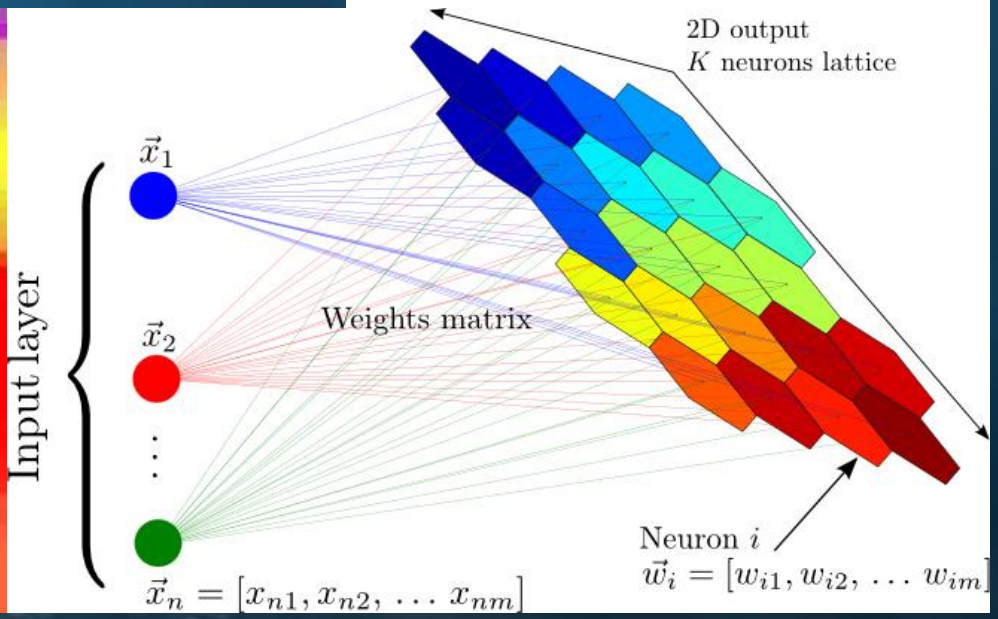


Self-Organizing Maps – SOM

Self-Organizing Maps (SOM) make the projection of an n-dimensional input data space into an m-dimensional output space, where m ≤ n. The dimension of the output data space is typically equal 2 (m = 2), however, it is usually not the best choice, because the output data space should not only group similar input patterns but also represent relations between these groups. Consequently, m should not be less than the number of the independent inputs! The lack of the ability to appropriately represent relations between 3 independent variables representing RGB color is noticeable in the 2D output space, in which not all similar colors lie side by side:







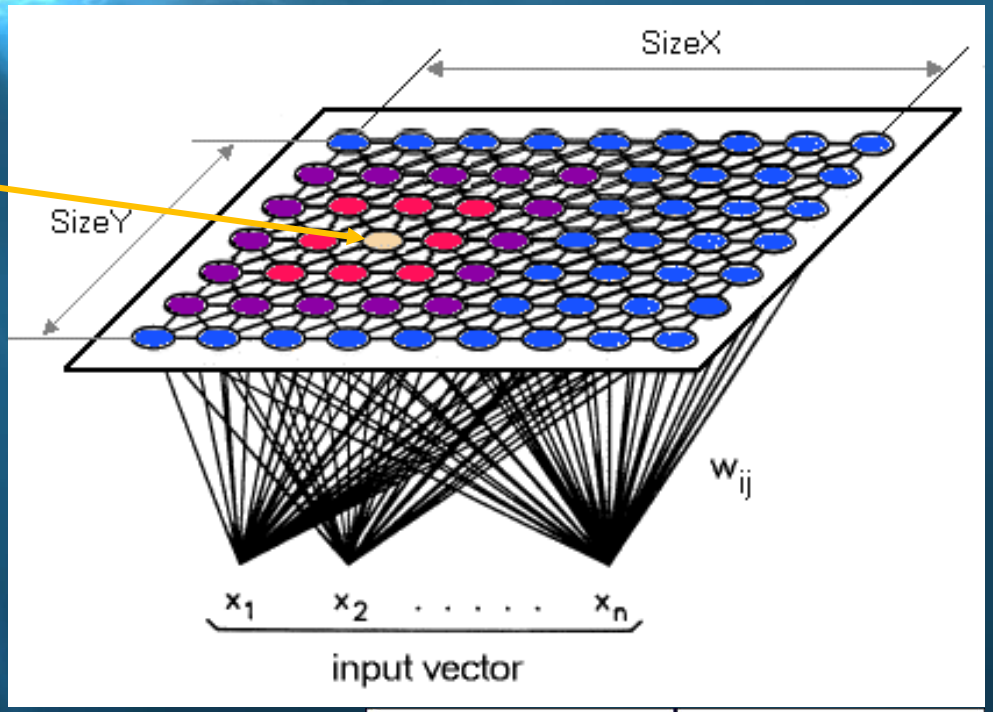
Competition of Nodes

In the Self-Organizing Maps (SOM), nodes compete with each other using the strategy winner-takes-most.

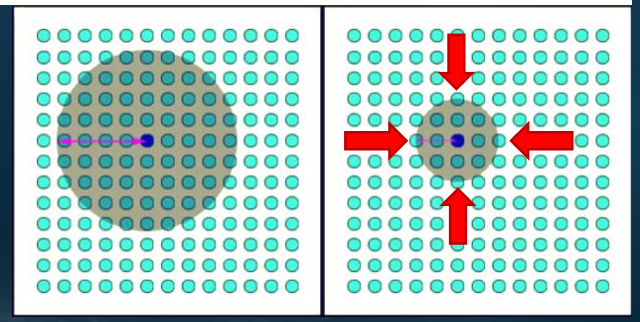
The winner is a node which weight vector is the nearest to the presented input vector. The distance between each weight vector $W^{i,j} = \left[w_1^{i,j}, w_2^{i,j}, \ldots, w_n^{i,j}\right]$ and the input vector $X^k = \left[x_1^k, x_2^k, \ldots, x_n^k\right]$ is typically computed using the Euclidean distance:

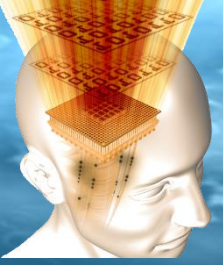
$$d\left(X^k, W^{i,j}(t)\right) = \sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{n}\left(x_k^k - w_k^{i,j}(t)\right)^2}$$

where weights are initially set using small random numbers.



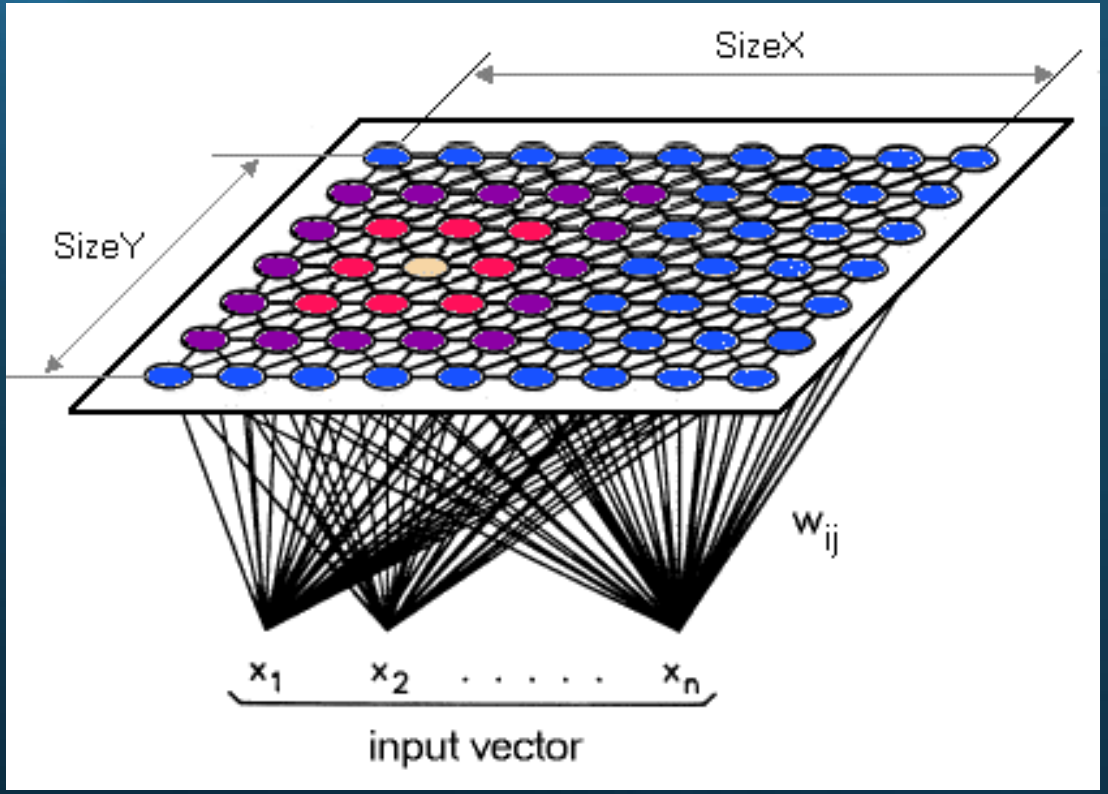
The winner has right to change its weights at most towards the input data vector.
Its closest neighbors do fewer changes in their weights towards the input data vector.
Its more distant neighbors do still fewer changes towards the input data vector.

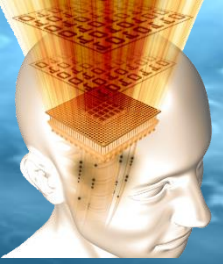




Learning Algorithm of SOM

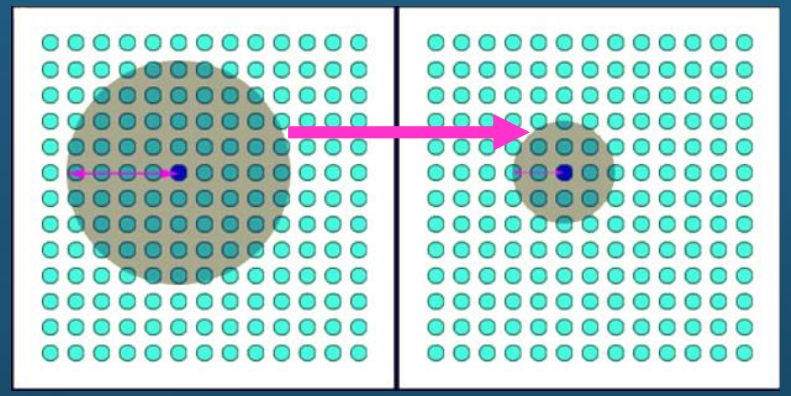
1. Create an output grid of nodes, which number should be much less than the number of training samples.
2. Initialize weights with small random numbers that should be less than the input data but greater than 0:
   $\frac{x_i^{max}-x_i^{min}}{100} \leq w_i \leq \frac{x_i^{max}-x_i^{min}}{10}$ where x_i^{min}, x_i^{max} are the minimum and maximum feature values
3. Take the subsequent or random input samples and compute the output values for all nodes.
4. Fix the closest node to the subsequent input sample „winner”: $(a,b) = arg \min_{i,j} d\left(X_k, W_{i,j}(t)\right)$
5. Update the weights of this node with the biggest learning rate, and do the same also to its direct and indirect neighbors with the decreasing strength.
6. Narrow down the sphere of neighbors and slightly reduce the learning rate.
7. Go back to the step number 3 and do it until each training sample is not represented by any of the grid nodes.





Training Parameters

During the SOM training, the radius of the updated nodes $\sigma(t)$ is still smaller and smaller, starting from the given initial radius σ_0 which can initially cover even the total grid of all neurons: $\sigma(t) = \sigma_0 \cdot e^{-\frac{t}{\alpha}}$



We also update the learning rate that represents the adaptation strength of the weight vector according to the training step:

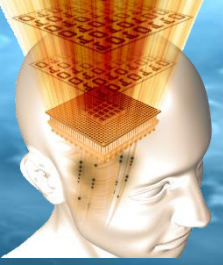
$\gamma(t) = \gamma_0 \cdot e^{-\frac{t}{\alpha}}$ where γ_0 is the initial learning rate γ_0 = **1**, and α = **1000** is the constant of narrowing.

Consequently, weights are updated after the following formula:

$$W_{i,j}(t+1) = W_{i,j}(t) + \delta(t) \cdot \gamma(t) \cdot \left(X_k - W_{i,j}(t)\right)$$

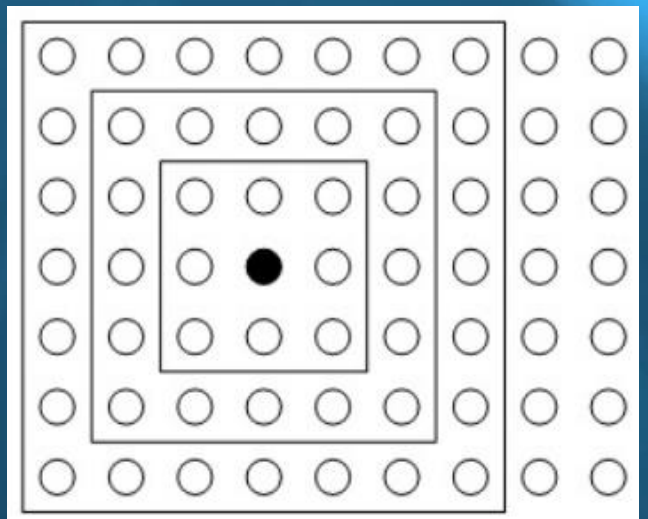
Where parameter $\delta(t)$ is winner distance-dependent and is equal:

$\delta(t) = e^{-\frac{d\left(N_{i,j}(t), N_{a,b}(t)\right)^2}{2 \cdot \sigma^2(t)}}$ where $N_{i,j}(t)$ is a neuron placed at the coordinates (i,j) in a 2D output space.

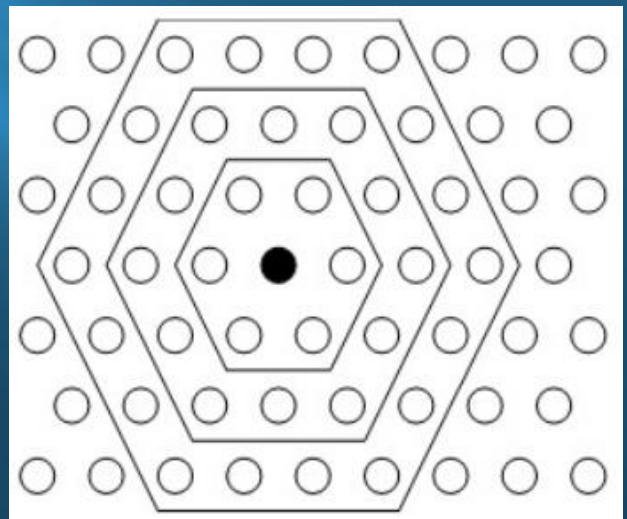


Grids, Neighborhood, and Distance

We can consider various grids and variously defined neighborhood:



Rectangular Grid



Hexagonal Grid

The distance between the winner and other nodes are usually computed after:

$$d\left(N_{i,j}(t), N_{a,b}(t)\right) = |i - a| + |j - b|$$

or

$$d\left(N_{i,j}(t), N_{a,b}(t)\right) = \sqrt{(i - a)^2 + (j - b)^2}$$

Tips and Tricks for Better Results

Self-Organizing Maps (SOM) can be better adapted to represent compact groups of samples and suitable distances between groups reproducing their similarity if:

- ✓ Weights will be correctly initialized by small random numbers.
- ✓ The dimension of the output SOM space will not be less than the number of the independent input variable.
- ✓ There will be used grids with the bigger number of neighbors:

- ✓ The number of nodes cannot be less than the expected number of clusters (groups) but should be significantly less than the number of training samples, e.g. sqrt(number of all training samples) to force each node to represent at least several training samples on average.
- ✓ We usually start with the output dimension equal two, but we should start from the dimension equal the number of independent input variables, and gradually increase this dimension until the winners representing the same groups of neurons get more or less the same distance in various instances of SOM networks starting from different initial weights.

Implementation and Presentation

Start with the Iris data and the 2D grid 4x4 or 5x5 neurons.

Associate one independent color from RGB color space with each class, e.g.:

- **Setosa – Red**
- **Versicolor – Green**
- **Virginica – Blue**

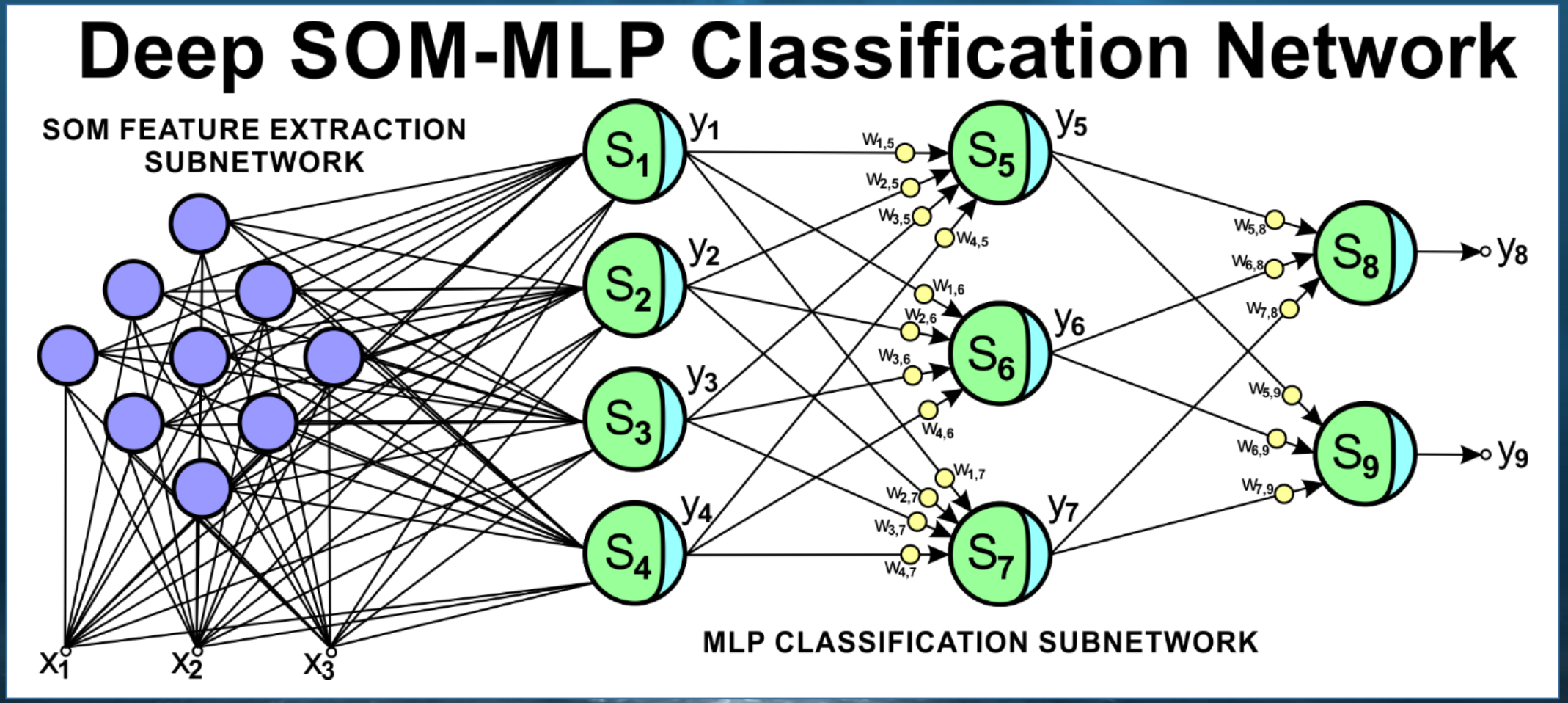
For each node create a three element table, which elements will separately count up the number of patterns of the above-mentioned classes to compute the final color of the node after the number of representatives of the given class divided by the number of all samples which won in this node:

e.g. R = 255 * (7 / 10), G = 255 * (2 / 10), B = 255 * (1 / 10).

Such computed colors can be used in the graphical interface of this method to represent nodes.

Deep networks using SOM and MLP

Self-Organizing Maps (SOM) are willingly used in an initial unsupervised feature extraction process because the SOM nodes represent groups of similar objects:



Bibliography and References

ACADEMIC WEBSITE - ADRIAN HORZYK, PhD, DSc.

AGH University of Science and Technology in Cracow, Poland
Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering
Department of Biocybernetics and Biomedical Engineering, Field of Biocybernetics

Dossier | Research | Publications | Courses | Graduates | Consultations | Contact

AGH

LECTURES

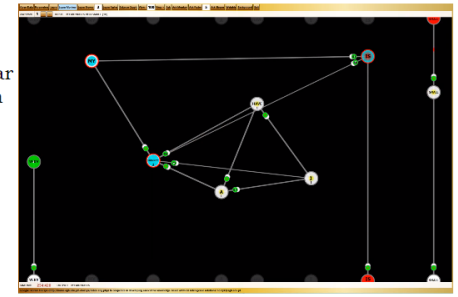
(will be renewed and expanded during the semester)

- Introduction to Artificial and Computational Intelligence
- Artificial Neural Networks, Multilayer Perceptron MLP, and Backpropagation BP
- Radial Basis Function Networks RBFN
- Unsupervised Training and Self Organizing Maps SOM
- Recurrent Neural Networks
- Introduction of Final Projects and Description of Requirements
- Associative Neural Graphs and Associative Structures
- Deep Associative Semantic Neural Graphs DASNG
- Associative Pulsing Neural Networks
- Deep Learning Strategies and Convolutional Neural Networks
- Support Vector Machines SVM
- Fuzzy Logic and Neuro-Fuzzy Systems
- Motivated and Reinforcement Learning
- Linguistic, Semantic Memories, and Cognitive Neural Systems
- Psychological Aspects of Intelligence, Human Needs, and Personality
- Writing Journal Papers

COMPUTATIONAL INTELLIGENCE

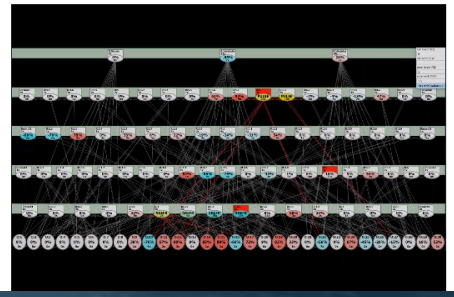
This course includes 28 lectures, 14 laboratory classes, and 14 project classes.

What is this course about?



This course is intended to give students a broad overview and deep knowledge about popular solutions and efficient neural network models as well as to learn how to construct and train intelligent learning systems in order to use them in everyday life and work. During the course we will deal with the popular and most efficient models and methods of neural networks, fuzzy systems and other learning systems that enable us to find specific highly generalizing models solving difficult tasks. We will also tackle with various CI and AI problems and work with various data and try to model their structures in such a way to optimize operations on them throughout making data available without necessity to search for them. This is a unique feature of associative structures and systems. These models and methods will allow us to form and represent knowledge in a modern and very efficient way which will enable us to mine it and automatically draw conclusions. You will be also able to understand solutions associated with various tasks of motivated learning and cognitive intelligence.

Lectures will be supplemented by laboratory and project classes during which you will train and adapt the solution learned during the lectures on various data. Your hard work and practice will enable you not only to obtain expert knowledge and skills but also to develop your own intelligent learning system implementing a few of the most popular and efficient CI methods.



Expected results of taking a part in this course:

- **Broad knowledge** of neural networks, associative and fuzzy systems as well as other intelligent learning systems.
- **Novel experience** and **broaden skills** in construction, adaptation and training of neural networks and fuzzy systems.
- **Ability to construct** intelligent learning systems of various kinds, especially deep learning solutions.
- **Good and modern practices** in modelling, construction, learning and generalization.
- **Own intelligent learning system** to use in your life or work.
- **Satisfaction** of enrollment to this course.

http://home.agh.edu.pl/~horzyk/lectures/ahdydci.php