Artificial Intelligence

Neural Networks

Lesson 13: Self-Organizing Maps

Vincenzo Piuri

Università degli Studi di Milano

Contents

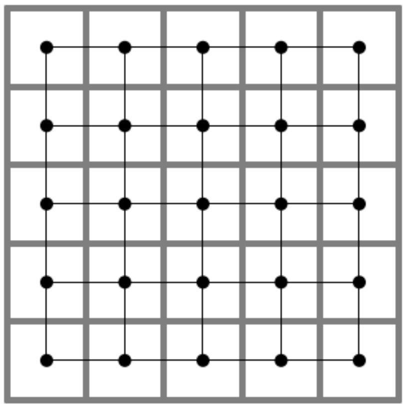
Self-Organizing Maps (1)

- A **self-organizing map** (or **Kohonen feature map**) is a feed-forward 2-layered neural network, similar to the Vector Quantization Learning networks with local connections only among neighboring hidden/output neurons
- The network input function of each output neuron is a distance function of input and weight vector
- The activation function of each output neuron is a **radial function**
  - Monotonically decreasing function
  - $f: \mathbb{R}_0^+ \to [0,1]$ with $f(0) = 1$ and $\lim_{x \to \infty} f(x) = 0$

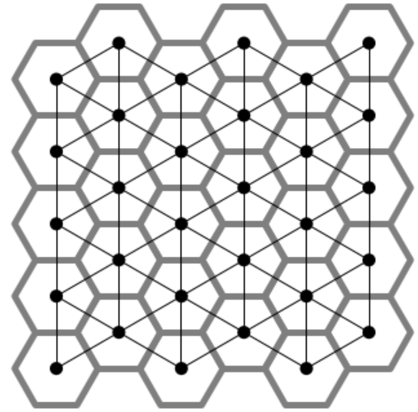
Self-Organizing Maps (2)

- The output function of each output neuron is the identity.
- The output is often discretized according to the “winner takes all” principle.
- On the output neurons a neighborhood relationship is defined

Self-Organizing Maps (3)



quadratic grid

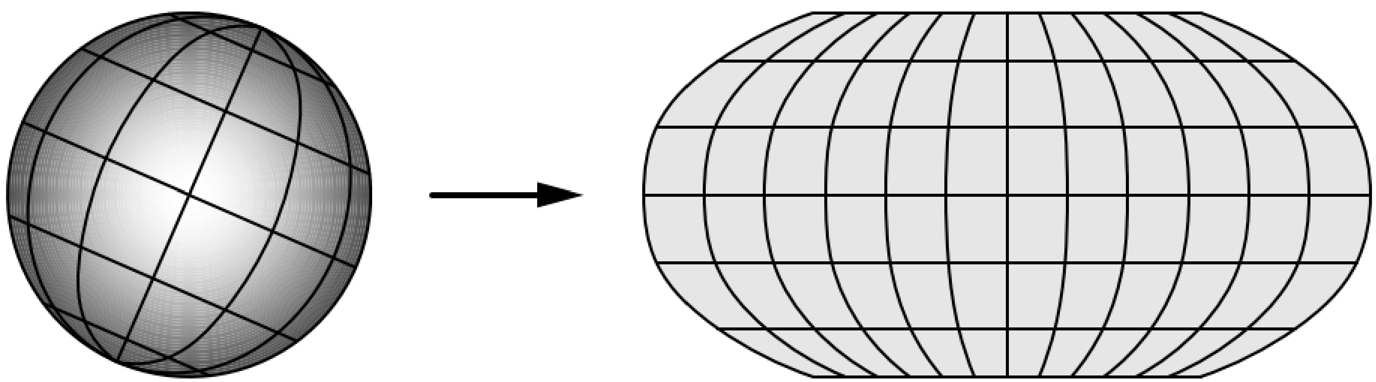


hexagonal grid

- Neighborhood of the output neurons: neurons form a grid
  - Thin black lines: Indicate nearest neighbors of a neuron
  - Thick gray lines: Indicate regions assigned to a neuron for visualization

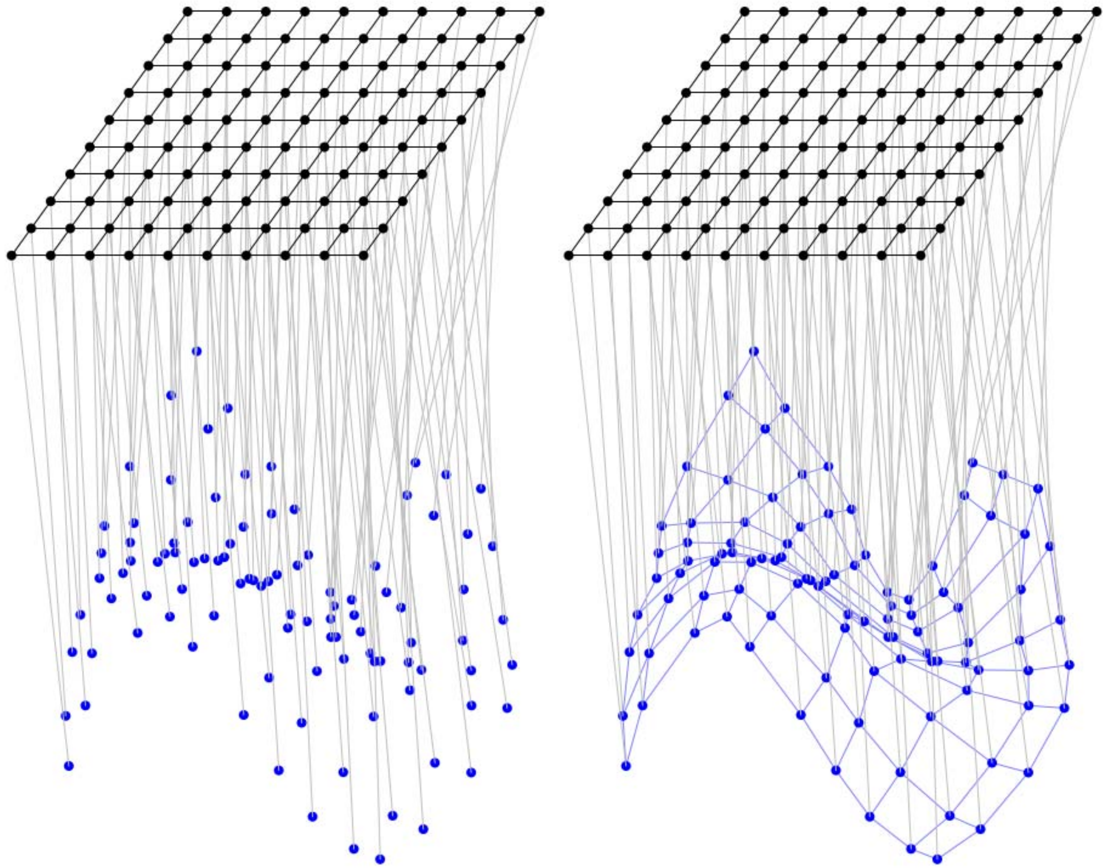
Topology Preserving Mapping (1)

- Images of points close to each other in the original space should be close to each other in the image space
  - Robinson projection of the surface of a sphere (maps from 3 dimensions to 2 dimensions)
  - The topology is preserved, although distances, angles, areas may be distorted



Topology Preserving Mapping (2)

- Neuron space/grid
  - Usually 2D quadratic or hexagonal grid
- Input/data space
  - Usually high-dim.
  - (here: only 3D)

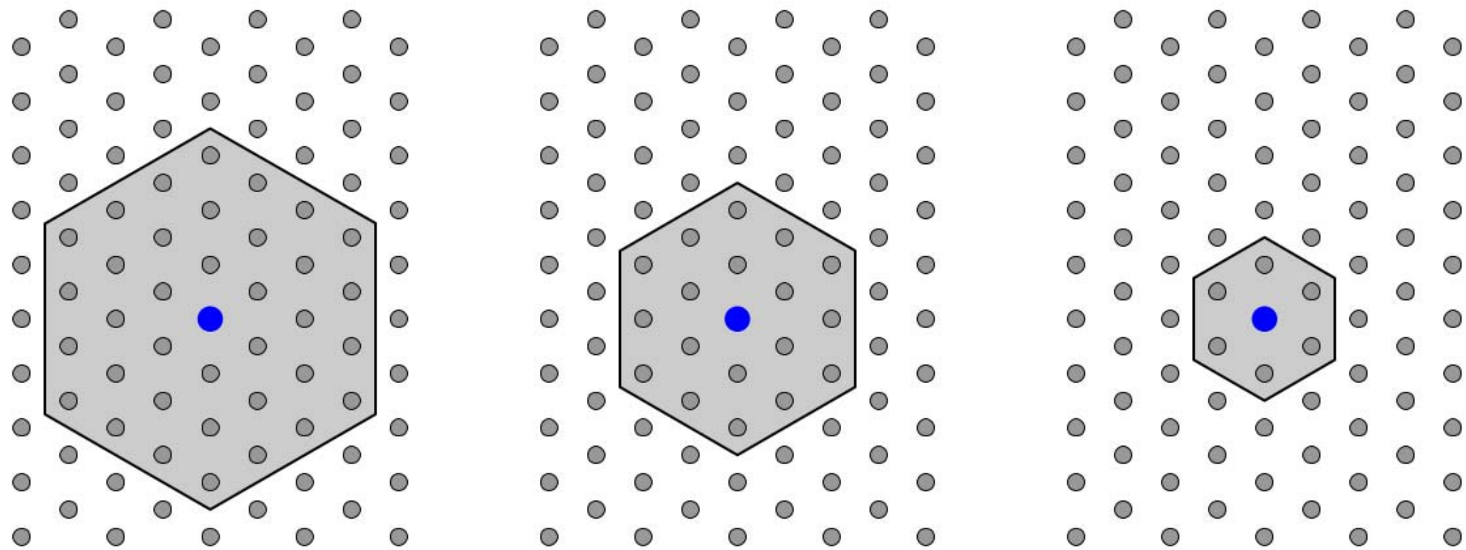


Topology Preserving Mapping (3)

- Find topology preserving mapping by respecting the neighborhood
  - $\vec{r}_u^{\,(new)} = \vec{r}_u^{\,(old)} + \eta(t)\, f_{nb}(d_{neurons}(u, u_*), \varrho(t))(\vec{x} - \vec{r}_u^{\,(old)})$
  - $u_*$is the winner neuron
(reference vector closest to data point)
  - The neighborhood function f_{nb} is a radial function
  - Time dependent learning rate
  - Time dependent neighborhood radius

Topology Preserving Mapping (4)

- The neighborhood size is reduced over time



  - A neighborhood function that is not a step function has a “soft” border and thus allows for a smooth reduction of the neighborhood size

Training

- Initialize the weight vectors of the neurons of the self-organizing map
  - Place initial reference vectors in the input/data space
  - Randomly selecting training examples
- Training
  - Choose a training sample / data point
  - Find the winner neuron with the distance function in the data space (the neuron with the closest reference vector)
  - Compute the time dependent radius and learning rate and adapt the corresponding neighbors of the winner neuron

Examples of Self-Organizing Maps (1)

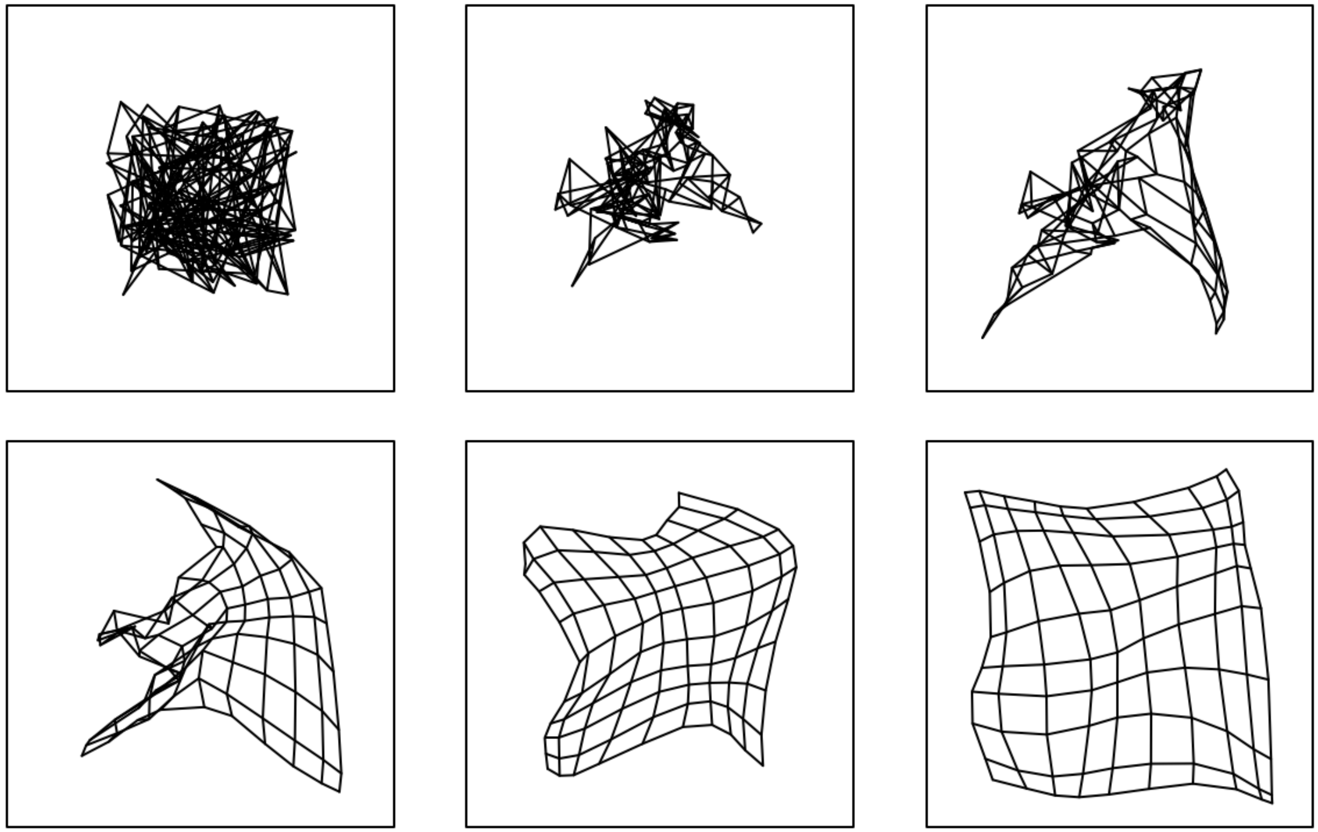
- Self-organizing map with 10×10 neurons (quadratic grid)
- Initialization with random reference vectors
- Gaussian neighborhood function

$$f_{\text{nb}}(d_{\text{neurons}}(u, u_*), \varrho(t)) = \exp\left(-\frac{d^2_{\text{neurons}}(u, u_*)}{2\varrho(t)^2}\right).$$

- Time-dependent neighborhood radius
- Time-dependent learning rate

Examples of Self-Organizing Maps (2)

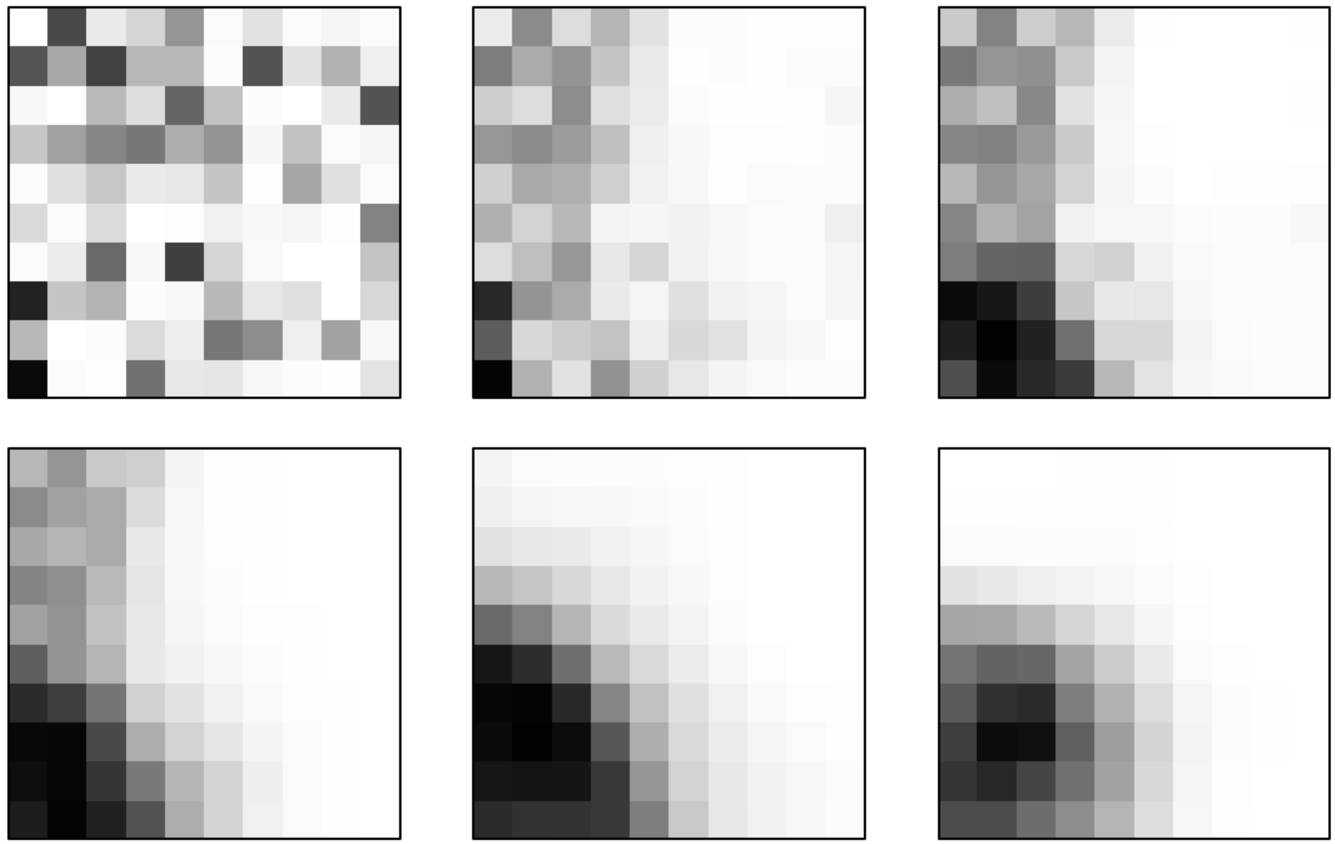
Data space



- SOM state after 10, 20, 40, 80 and 160 training steps

Examples of Self-Organizing Maps (3)

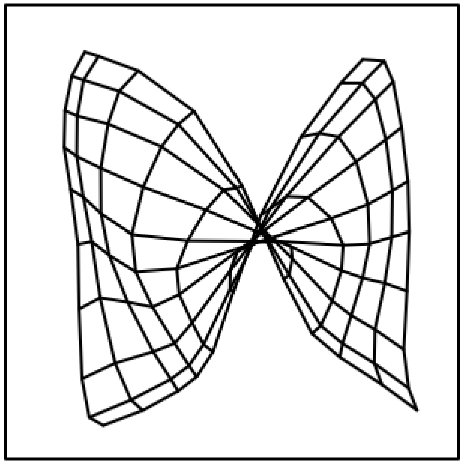
Neuron grid



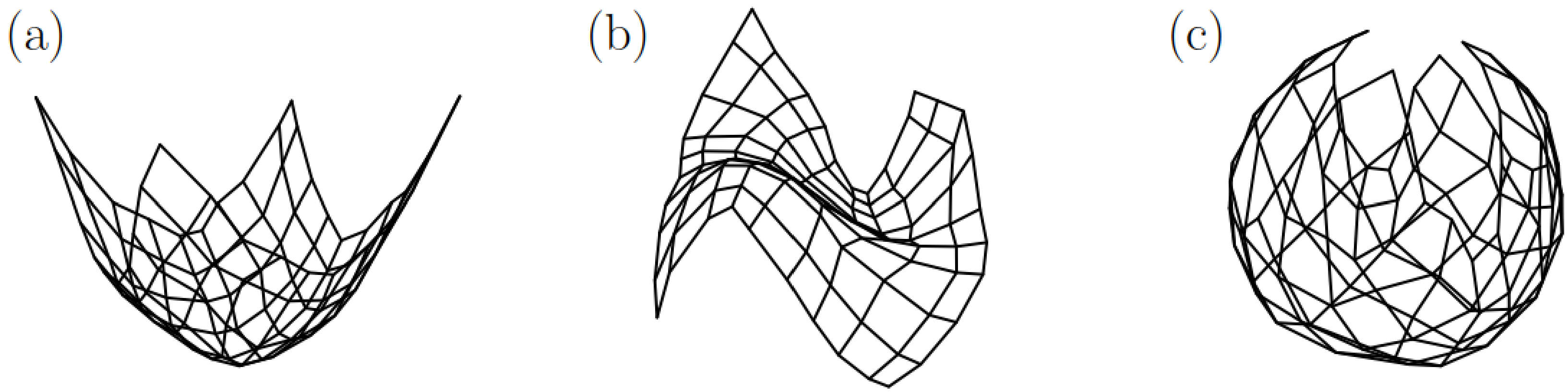
- SOM state after 10, 20, 40, 80 and 160 training steps

Examples of Self-Organizing Maps (4)

- Training a self-organizing map may fail if
  - the (initial) learning rate is chosen too small or
  - the (initial) neighborhood radius is chosen too small.

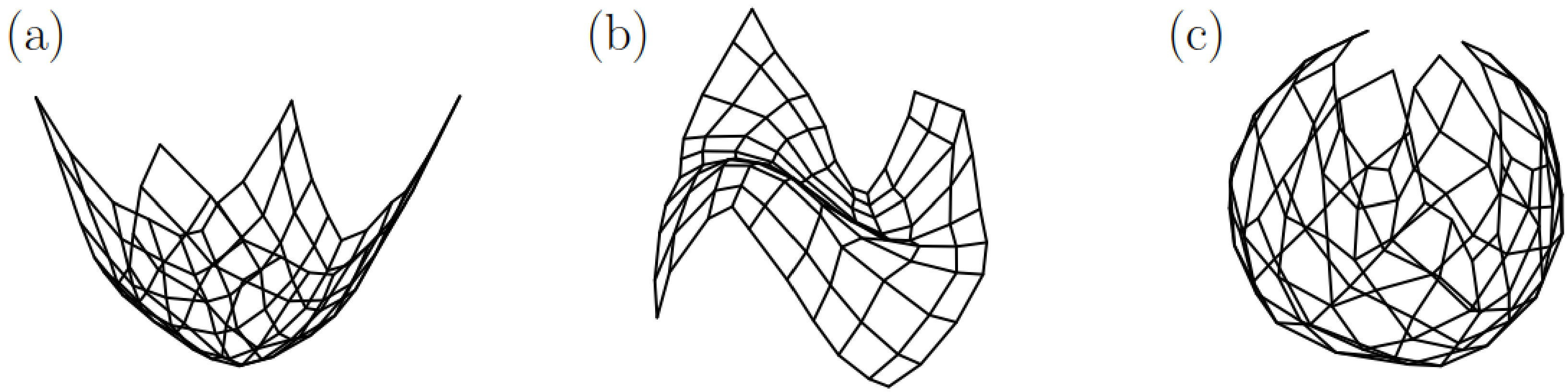


Examples of Self-Organizing Maps (5)



- Self-organizing maps that have been trained with random points from
  - (a) a rotation parabola
  - (b) a simple cubic function
  - (c) the surface of a sphere

Examples of Self-Organizing Maps (6)



- Original space and image space have different dimensionality
  - (In the previous example they were both two-dimensional)
- Self-organizing maps can be used for dimensionality reduction