Self-Organizing Maps

Self-Organizing Maps

A **self-organizing map** or **Kohonen feature map** is a neural network with a graph $G = (U, C)$ that satisfies the following conditions

(i) $U_{\text{hidden}} = \emptyset$, $U_{\text{in}} \cap U_{\text{out}} = \emptyset$,

(ii) $C = U_{\text{in}} \times U_{\text{out}}$.

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**, that is, a monotonically decreasing function

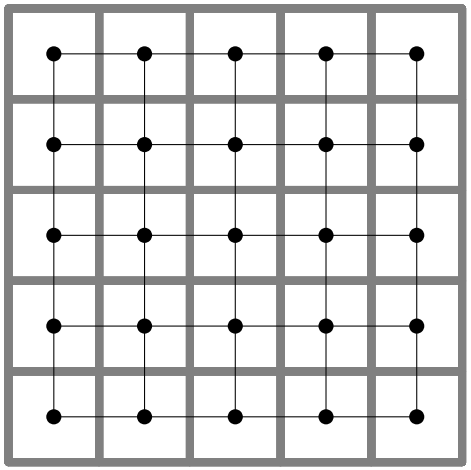
$$f : \mathbb{R}_0^+ \to [0, 1] \quad \text{with} \quad f(0) = 1 \quad \text{and} \quad \lim_{x \to \infty} f(x) = 0.$$

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:

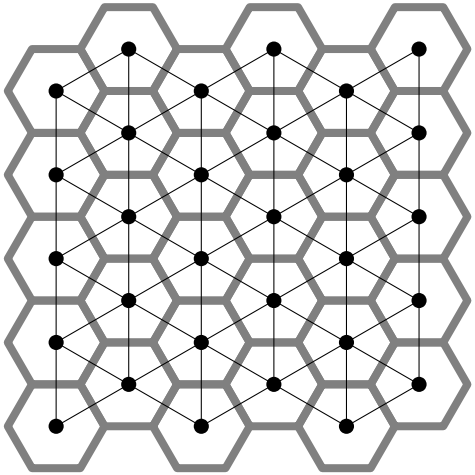
$$d_{\text{neurons}} : U_{\text{out}} \times U_{\text{out}} \to \mathbb{R}_0^+ .$$

Self-Organizing Maps: Neighborhood

Neighborhood of the output neurons: neurons form a grid



quadratic grid



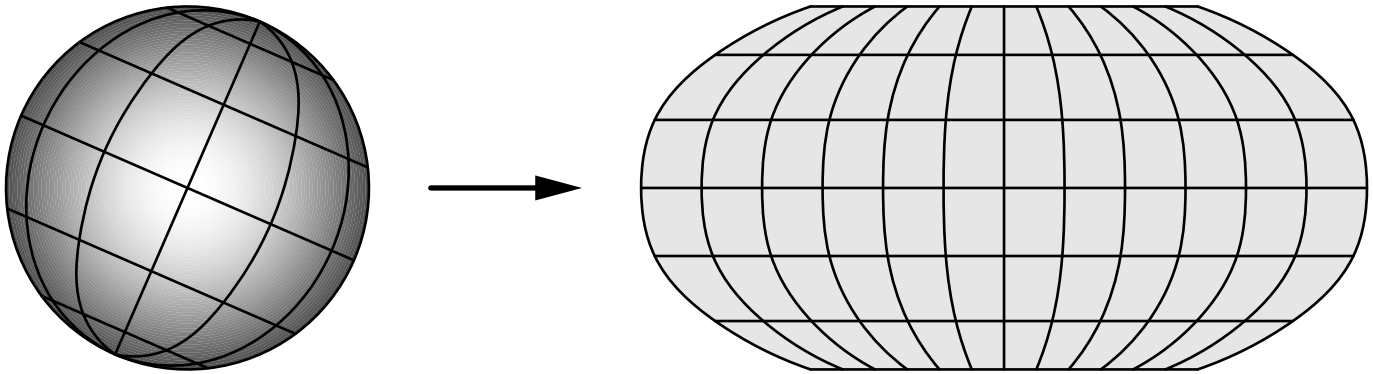
hexagonal grid

- Thin black lines: Indicate nearest neighbors of a neuron.
- Thick gray lines: Indicate regions assigned to a neuron for visualization.
- Usually two-dimensional grids are used to be able to draw the map easily.

Topology Preserving Mapping

Images of points close to each other in the original space should be close to each other in the image space.

Example: **Robinson projection** of the surface of a sphere
(maps from 3 dimensions to 2 dimensions)



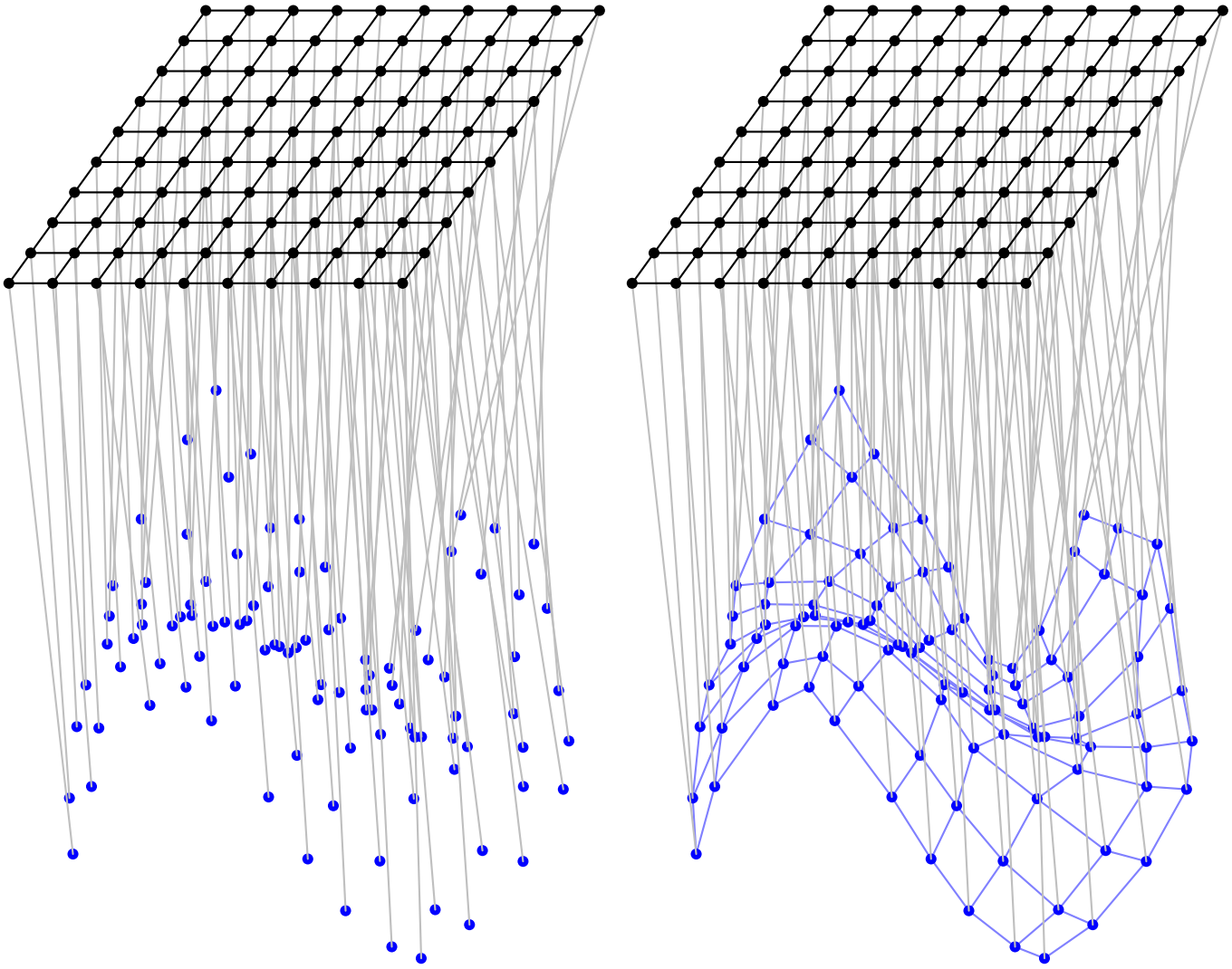
- Robinson projection is/was frequently used for world maps.
- The topology is preserved, although distances, angles, areas may be distorted.

Self-Organizing Maps: Topology Preserving Mapping

neuron space/grid
usually 2-dimensional
quadratic or
hexagonal grid

input/data space
usually high-dim.
(here: only 3-dim.)
blue: ref. vectors

Connections may
be drawn between
vectors corresponding
to adjacent neurons.



Self-Organizing Maps: Neighborhood

Find topology preserving mapping by respecting the neighborhood

Reference vector update rule:

$$\vec{r}_u^{\,(\text{new})} = \vec{r}_u^{\,(\text{old})} + \eta(t) \cdot f_{\text{nb}}(d_{\text{neurons}}(u, u_*), \varrho(t)) \cdot (\vec{x} - \vec{r}_u^{\,(\text{old})}),$$

- u_* is the winner neuron (reference vector closest to data point).
- The neighborhood function f_{nb} is a radial function.

Time dependent learning rate

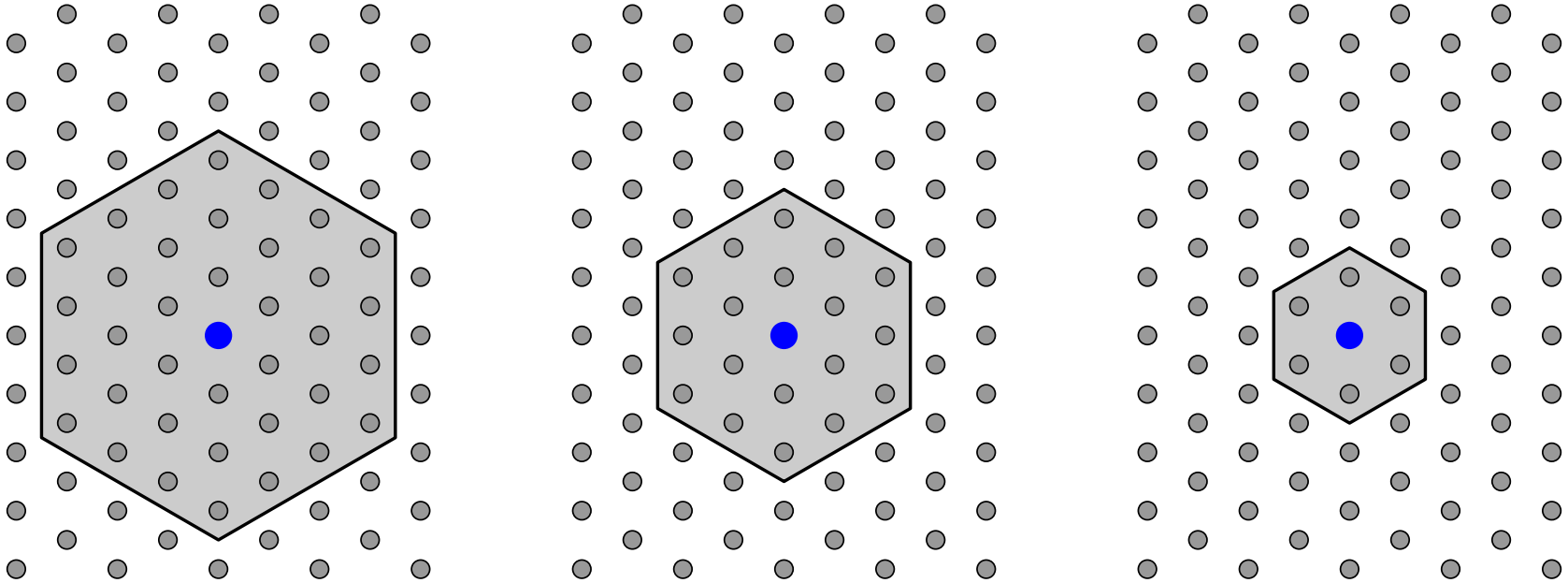
$$\eta(t) = \eta_0 \alpha_\eta^t, \quad 0 < \alpha_\eta < 1, \qquad \text{or} \qquad \eta(t) = \eta_0 t^{\kappa_\eta}, \quad \kappa_\eta < 0.$$

Time dependent neighborhood radius

$$\varrho(t) = \varrho_0 \alpha_\varrho^t, \quad 0 < \alpha_\varrho < 1, \qquad \text{or} \qquad \varrho(t) = \varrho_0 t^{\kappa_\varrho}, \quad \kappa_\varrho < 0.$$

Self-Organizing Maps: Neighborhood

The neighborhood size is reduced over time: (here: step function)



Note that a neighborhood function that is not a step function has a "soft" border and thus allows for a smooth reduction of the neighborhood size (larger changes of the reference vectors are restricted more and more to the close neighborhood).

Self-Organizing Maps: Training Procedure

- Initialize the weight vectors of the neurons of the self-organizing map, that is, place initial reference vectors in the input/data space.
- This may be done by randomly selecting training examples (provided there are fewer neurons that training examples, which is the usual case) or by sampling from some probability distribution on the data space.
- For the actual training, repeat the following steps:
  - Choose a training sample / data point (traverse the data points, possibly shuffling after each epoch).
  - Find the winner neuron with the distance function in the data space, that is, find the neuron with the closest reference vector.
  - Compute the time dependent radius and learning rate and adapt the corresponding neighbors of the winner neuron (severity of weight changes depend by neighborhood and learning rate).

Self-Organizing Maps: Examples

Example: **Unfolding of a two-dimensional self-organizing map.**

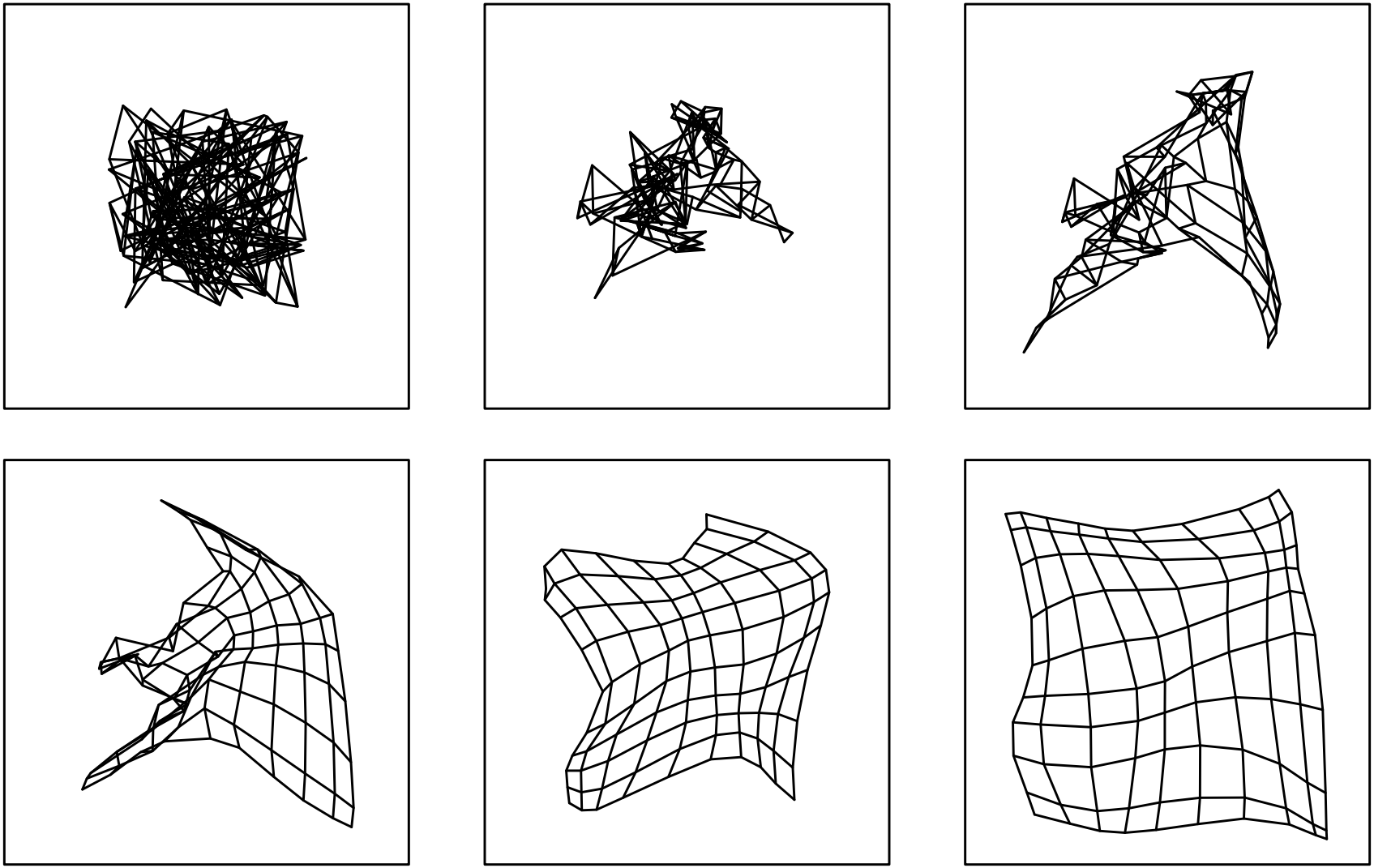
- Self-organizing map with 10×10 neurons (quadratic grid) that is trained with random points chosen uniformly from the square $[-1, 1] \times [-1, 1]$.
- Initialization with random reference vectors chosen uniformly from $[-0.5, 0.5] \times [-0.5, 0.5]$.
- Gaussian neighborhood function

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

- Time-dependent neighborhood radius $\varrho(t) = 2.5 \cdot t^{-0.1}$
- Time-dependent learning rate $\eta(t) = 0.6 \cdot t$.
- The next slides show the SOM state after 10, 20, 40, 80 and 160 training steps. In each training step one training sample is processed. Shading of the neuron grid shows neuron activations for $(-0.5, -0.5)$.

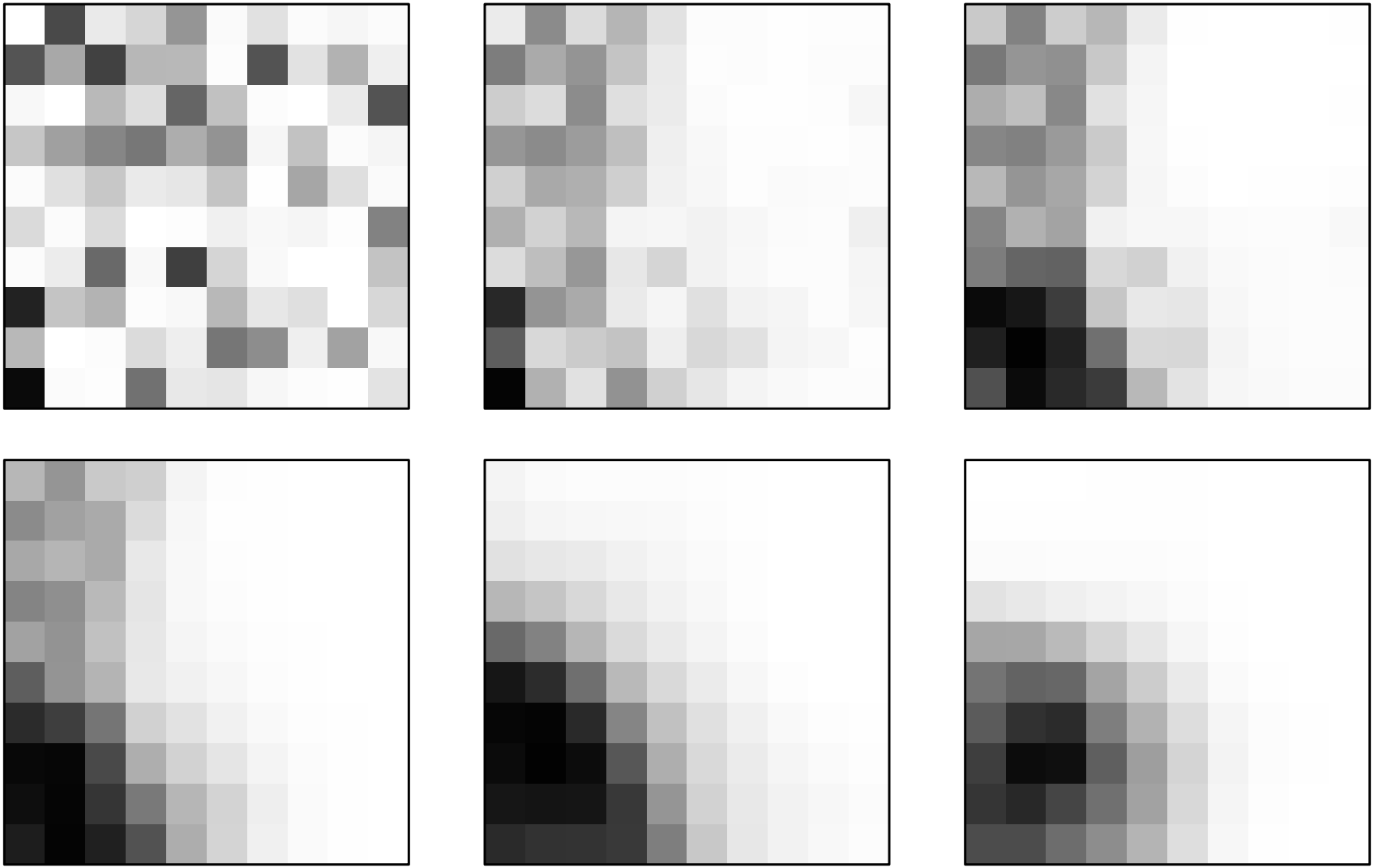
Self-Organizing Maps: Examples

Unfolding of a two-dimensional self-organizing map. (data space)



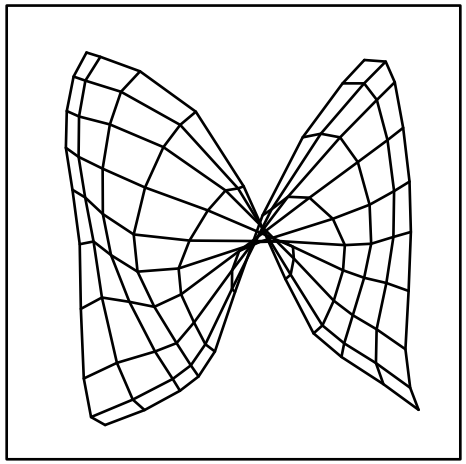
Self-Organizing Maps: Examples

Unfolding of a two-dimensional self-organizing map. (neuron grid)



Self-Organizing Maps: Examples

Example: Unfolding of a two-dimensional self-organizing map.

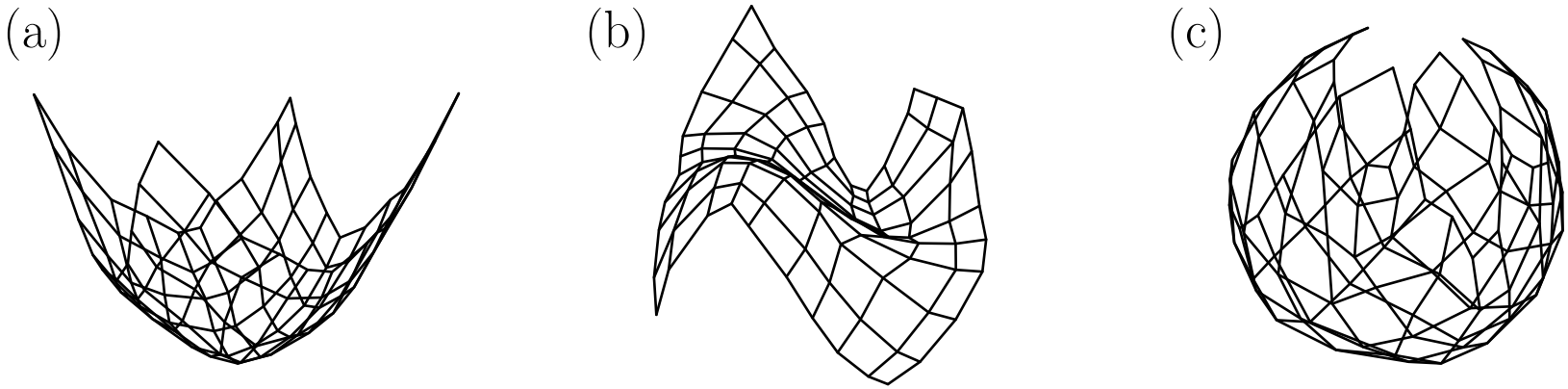


Training a self-organizing map may fail if

- the (initial) learning rate is chosen too small or
- or the (initial) neighborhood radius is chosen too small.

Self-Organizing Maps: Examples

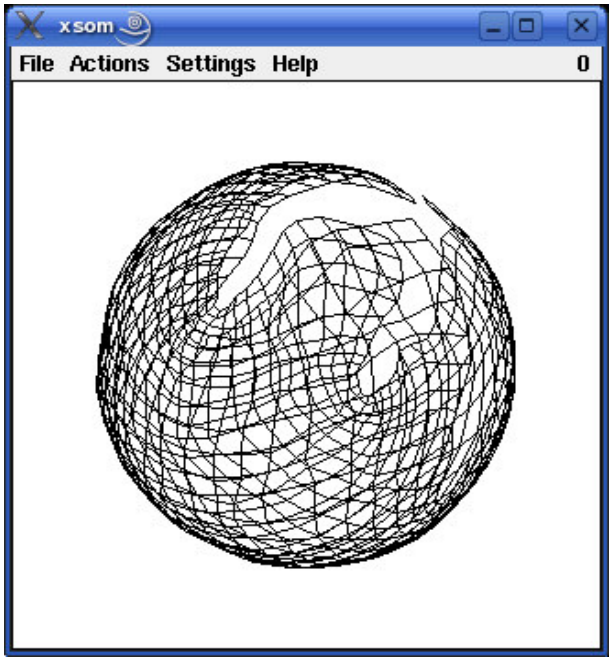
Example: Unfolding of a two-dimensional self-organizing map.

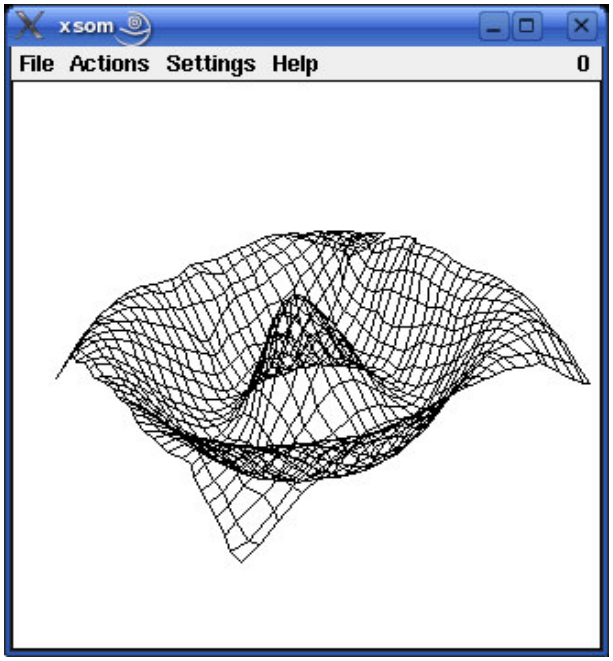


Self-organizing maps that have been trained with random points from
(a) a rotation parabola, (b) a simple cubic function, (c) the surface of a sphere.
Since the data points come from a two-dimensional subspace, training works well.

- In this case 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
  (in a quantized fashion, but interpolation may be used for smoothing).

Demonstration Software: xsom/wsom





Demonstration of self-organizing map training:

- Visualization of the training process
- Two-dimensional areas and three-dimensional surfaces
- `http://www.borgelt.net/somd.html`