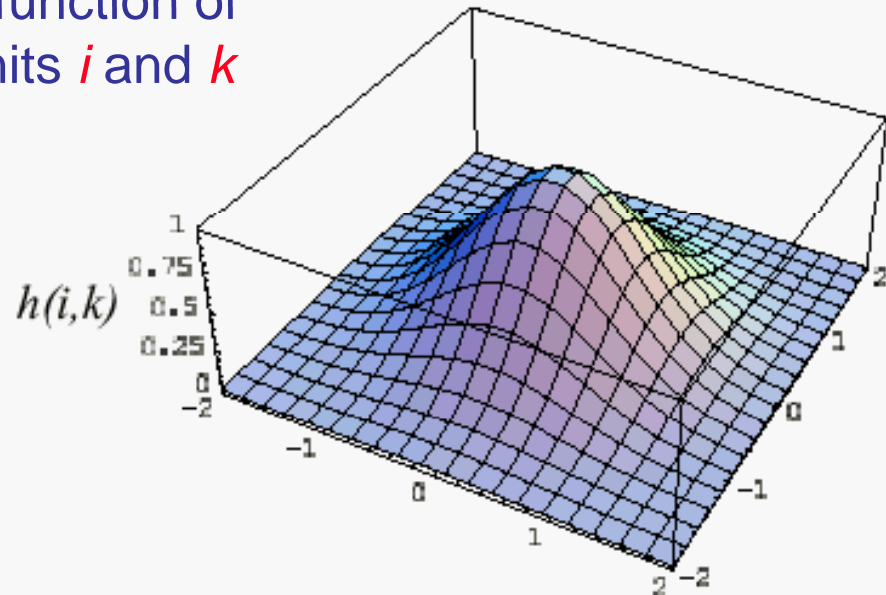


Self-Organizing Maps (SOM)

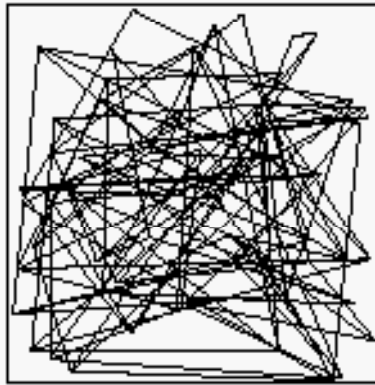
- output units *ordered* in some fashion, usually a 2D grid
- weights to output units are adapted so that order present in input space \mathbb{R}^N is preserved in output
- learning patterns near each other in input space are mapped to neighbouring output units

Kohonen Weight Update

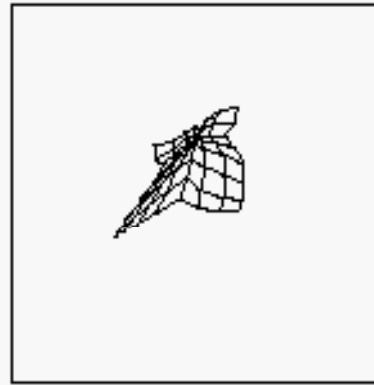
- using same formulae as before for vector quantization, winning unit is selected
- weights to winning unit **as well as its neighbours** are adapted using
$$w_i(t+1) = w_i(t) + \gamma h(i,k)(x(t) - w_i(t)) \quad \forall i \in S$$
- where $h(i,k)$ is a decreasing function of the grid-distance between units i and k



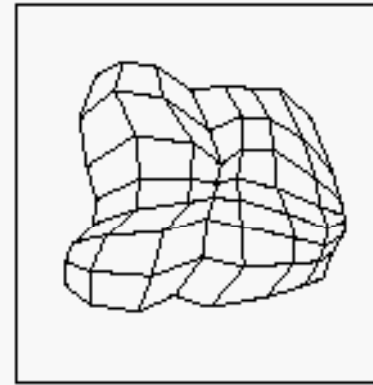
Topology-conserving map



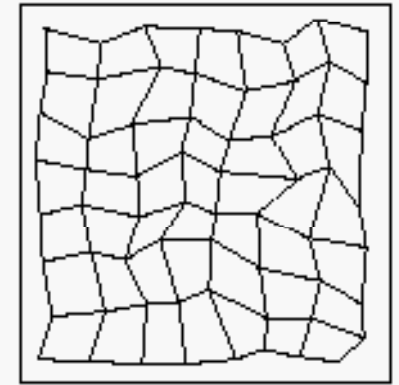
Iteration 0



Iteration 200



Iteration 600



Iteration 1900

- network consists of 2 inputs & 8x8 outputs in planar grid
- vertices indicate weight vectors
- lines are drawn (for illustration) to connect weight $w_{i,(j_1,j_2)}$ with weights $w_{i,(j_1+1,j_2)}$ and $w_{i,(j_1,j_2+1)}$

Elastic Networks

- given N cities, X_k
- construct network of M units or “beads” Y_j
- each Y_j connected to its two neighbours Y_{j-1} and Y_{j+1}
- distribute beads to form shortest cost circuit through some collection of target points X_k

- minimize total energy:

$$E(\{Y_j\}, K) = -\alpha K \sum_i \log \sum_j e^{-(X_i - Y_j)^2 / 2K^2} + \beta \sum_j (Y_j - Y_{j+1})^2$$

How do we minimize E?

- adjust positions of the M beads, Y_j by:

$$\Delta Y_j = \alpha \sum_i w_{ij} (X_i - Y_j) + \beta K (Y_{j+1} + Y_{j-1} - 2Y_j)$$

where

$$w_{ij} = \frac{e^{-(X_i - Y_j)^2 / 2K^2}}{\sum_k e^{-(X_i - Y_k)^2 / 2K^2}}$$