Neuronale Netze (SS 2002), 8.7.

Self-organizing maps (contd.)

- **Standard SOM:**

  Neurons $\{1, \ldots, N\}$ are additionally equipped with a neighborhood structure $d(i, j)$.

  Often: $d(i, j)$ is given by a lattice, distance of neurons i and j in the lattice.

  - Training:
    init $\vec{w}^i$
    repeat
    choose $\vec{x}$
    compute the winner i_0
    adapt all neurons $w^i += \eta \cdot \exp(-d(i, i_0)/\sigma^2)(\vec{x} - \vec{w}^i)$
    $\eta* = \eta^-$, $\sigma* = \sigma^-$
  - Application areas: visualization, function approximation, initialization of VQ/LVQ, robotics, web-mining, ...
  - SOM is not a gradient descent, but a variation thereof is:

    $$\sum_{i,\vec{x}} 1_{\vec{w}^i \text{ winner for } \vec{x}} \sum_k \exp(-d(i, k))(\vec{w}^k - \vec{x})^2$$

    (= minimization of the overall quantization for each neuron averaged over the neighborhood) where the winner is the neuron with minimum averaged quantization error.
  - There exist various measures for topology preservation: roughly, the topology of the weigths in the data space should coincide with the topology of the neurons in the lattice.

    Measured: via measuring the relative distances of weights and indices (problem: 'Frankfurter Würstchen'), or the respective neighborhood structure given by the receptive fields and the lattice.

- **Growing SOM:** start with minimum lattice and add rows or dimensions of the lattice at appropriate places until the quantization error is small enough. Appropriate places: the direction where the quantization error is maximum.

- **Alternative lattices:**
  - Hexagonal lattices (like honeycombs, they are more stable for small neighborhood range)
  - Hyperbolic SOM - regular lattices in the hyperbolic space, advantage: expinentially increasing number of neighbors
- **Neural gas:** no prior lattice is given, in each step, neurons are adapted according to their distance from the data point

  Algorithm:
  init $\vec{w}^i$
  repeat
  choose $\vec{x}$
  compute $|\vec{x} - \vec{w}^i|$ for all i
  $r_i :=$ number of neurons closer to $\vec{x}$ than $\vec{w}_i$
  adapt: $\vec{w}^i += \eta \cdot \exp(-\lambda r_i)(\vec{x} - \vec{w}^i)$

  This is a gradient descent on the function

  $$\sum_{i,\vec{x}} \exp(-\lambda r_i)(\vec{x} - \vec{w}^i)^2$$

  Posterior lattice: neurons are neighbored if they are the first and second winner for a data point.

One self-organizing neuron:
neuron with weight $\vec{w}$ computes for input $\vec{x}$ the output $y = \vec{w}^t\vec{x}$

Hebbian learning: $\vec{w} += \eta y \vec{x}$

- Assume $Ex_i = 0$, correlation matrix $E(\vec{x}\vec{x}^t)$, this is positive semidefinite and symmetric, can be diagonalized with an orthonormalbase of eigenvectors, say $e_1, \ldots, e_n$ with eigenvalues $\lambda_1 > \ldots > \lambda_n$. We assume for simplicity that they are pairwise different! E_1 is the first **principal component**, e_i the ith principal component. Finding these directions is called principal component analysis.

  The first principal component is the direction of maximum information, the next components the directions of maximum information in the respective orthogonal space. e_i optimizes the function $\vec{w}^t C \vec{w}$ where $|\vec{w}| = 1$.

- Hebbian learning explodes into the direction of e_1, it is a gradient descent of $\vec{w}^t C \vec{w}$ in the mean.
- The update $\vec{w} = (\vec{w}\eta y\vec{x})/|\vec{w}\eta y\vec{x}|$ converges in the mean to the first principal component.
- **Oja-rule:** $\vec{w}+ = \eta(y\vec{x} - y^2\vec{w})$
  Idea behind the rule: Taylor-expansion of normalized Hebb

  Mean: $E(\triangle\vec{w}) = \eta(C\vec{w} - \vec{w}^t C\vec{w}\vec{w})$

  Fixed points: principal components (via $E(\triangle\vec{w}) = 0$)

  Stable fixed point: first principal component (via negative definiteness of Jacobian)
- **Yuille:** $\triangle vecw = \eta(\vec{w}^t\vec{x}\vec{x} - |\vec{w}|^2\vec{w})$

  Idea behind the rule: mean is gradient ascent for $1/2\vec{w}^t C\vec{w} - 1/4|\vec{w}|^4$

  stable fixed point: $\sqrt{\lambda_1}e_1$
- **Hassoun:** $\triangle vecw = \eta(\vec{w}^t\vec{x}\vec{x} - \lambda(1 - 1/|\vec{w}|)\vec{w})$

  Idea behind the rule: mean is gradient ascent for $1/2\vec{w}^t C\vec{w} - \lambda 2(|\vec{w}| - 1)^2$

  stable fixed point: $\lambda/(\lambda - \lambda_1)e_1$ for $\lambda > \lambda_1$
- **Sanger:** n neurons for n independent components

  $\triangle\vec{w}_j = \eta(\vec{w}_j^t\vec{x}(\vec{x} - \sum_{i=1}^{j}\vec{w}_i^t\vec{x}\vec{w}_i)$

  Idea behind the rule: Oja + subtracting the parts of the inputs $\vec{x}$ which ly into the directions of already computed eigenvectors.

  stable fixed points: e_i for neuron number i