**Neuronale Netze (SS 2002), 8.7.**

**Fully recurrent neural networks (contd.)**

- **Alternative training:**

  A training problem for a Hopfieldnetwork can be equivalently formulated as perceptron training problem:

  Hopfield pattern $\vec{x}^p$ and weights $w_{ij}$, biases as on-neurons
  $\rightarrow$ perceptron with weights $w_{ij}$, for $i < j$, no bias, and pattern $(\vec{X}(I)^p, y(I)^p)$, $I = 1, \ldots, N$ where $y(I)^p = x_I^p$, $X(I)_{ij}^p = x_i^p$ if $j = I$ and $i \neq I$, $X(I)_{ij}^p = x_j^p$ if $i = I$ and $j \neq I$, $X(I)_{ij}^p = 0$ otherwise.

  - this yields a solution if existing,
  - this demonstrates the in-principle restriction of Hopfield networks without hidden neurons.

- **Hopfield networks for minimization:**

  Idea: write an arbitrary polynomial of degree two as energy function, the Hopfield network will then find (local) minima.

  - Minimize $\sum_{i \leq j} a_{ij} X_i X_j + \sum_i b_i X_i + c$ with $X_i \in \{0, 1\}$ becomes minimize $-\sum_{i<j} -a_{ij} X_i X_j + \sum_i (b_i + a_{ii}) X_i$ which is the energy function of a Hopfield network with weights $w_{ij} = -a_{ij}$ and biases $b_i + a_{ii}$.
  - The TSP can be solved finding global optima in the polynomial
    $\sum_{j=1}^{n-1} \sum_{kl} d_{kl} X_{kj} X_{lj+1} + \sum_{kl} d_{kl} X_{kn} X_{l1}$ (minimal tour length)
    $+\alpha \sum_k \left(\sum_i X_{ki} - 1\right)^2$ (each city precisely once)
    $+\beta \sum_i \left(\sum_k X_{ki} - 1\right)^2$ (one city at each time step)
    where $X_{ij}$ is 1 iff city $i$ is the $j$th city of the tour, $\alpha$ and $\beta$ are large values.

    Hence Hopfield networks could solve TSP if they always relaxed to global optima.
  - Various different problems can be written in a similar form.
  - Polynomials of arbitrary degree would be possible with hidden neurons.

- **The Boltzman machine:**

  This is a stochastic version of a Hopfield network with two benefits: local minima can better be avoided, training with hidden neurons is possible

  – Dynamics: if $i_0$ is chosen, it becomes 1 with probability $\left(1 + \mathrm{e}^{-\mathrm{net}_i/T}\right)^{-1}$ where $T > 0$ is the temperature.

  – $T \to 0$: Hopfield network
  $T \to \infty$: random updates
  fixed $T$: the probability of a state $\vec{o}$ can be computed as (Boltzmann distribution)

  $$p^*(\vec{o}) = \frac{\mathrm{e}^{-E(\vec{o})/T}}{\sum_{\vec{o}} \mathrm{e}^{-E(\vec{o})/T}} \,.$$

  hence minima of the energy function have a higher probability.

  – Possibility of simulated annealing in order to avoid local optima: start with large $T$ and decrease the temperature - there exist appropriate schedules, borrowed from physics.

  – Training (with hidden neurons) starts at the assumption $p^* \approx p^+$ for some desired distribution $p^+$, e.g. uniform distribution of the training patterns.

  Minimize the cross-entropy:
  $E(p^*, p^+) = -\sum_{\vec{o}} p^+(\vec{o}) \cdot \ln \frac{p^*(\vec{o})}{p^+(\vec{o})}$ with gradient descent finally yields (time consuming) Boltzmann training:

    * init weights at random
    * iteratively update $w_{ij} += \epsilon($ Hebb $-$ Anti-Hebb $)$ where
    * Hebb is the correlation of $o_i$ and $o_j$ for desired patterns, thereby hidden neurons are relaxed clamping the visible units,
    * Anti-Hebb is the already existing correlation of $o_i$ and $o_j$ for the network, which is estimated via relaxation of the network (free running).

**Unsupervised/self-organizing networks:**

- These are FNNs with only two layers and a winner-takes-all output; given the output neurons $\vec{w}^1, \ldots, \vec{w}^N$, input $\vec{x}$, the neuron $\vec{w}^i$ with smallest $|\vec{x}-\vec{w}^i|$ or largest $(\vec{w}^i)^t\vec{x}$, respectively, declares itself as the **winner**.

  Often: alternative notation via the neurons/prototypes/codebook vectors and their respective receptive fields.

- **Vector quantization:**

  init $\vec{w}^i$ at random
  iteratively choose data point $\vec{x}$,
  compute the winner $\vec{w}^{i_0}$,
  adapt $\vec{w}^{i_0} += \epsilon(\vec{x} - \vec{w}^{i_0})$

  This is a stochastic gradient descent on the error surface

  $$\frac{1}{2}\sum_p (\vec{x}^p - \vec{w}^i)^2 M_i(\vec{x}^p)$$

  where $M_i(\vec{x}^p) \in \{0, 1\}$ is the characteristic function of the receptive field of $\vec{w}^i$.

  This update is very sensitive w.r.t initialisation.