Machine Learning

Michael M. Richter

Neural Networks (2)

Email: mrichter@ucalgary.ca
Neural Nets (2)

Part 5

Unsupervised Learning
Unsupervised Learning

Idea:
- The neural network gets only some input patterns and has to arrange them in (hopefully) meaningful clusters
- See clustering
- Example

![Diagram showing circle, triangle, and square connected to a neural network]]
A Method: Competitive Learning (1)

- The teacher has to be replaced by an internal mechanism for making a decision.
- One way to do it is an internal competition
- Two neurons compete:
Competitive Learning (2)

Remark: There can be more than two levels
Competitive Learning (3)

**Assumptions:**
- We have McCulloch-Pitts neurons
- Normalized weights $\sum w_{ij} = 1 \quad \forall j, \ j < w_{ij} < 1$
- Input, output values are binary

**Two possibilities:**
- **one** neuron is the winner and can learn ("winner takes all")
- several neurons can learn
- Basic idea to define winning: Getting highest input.
Competitive Learning (4)

The learning algorithm (with learning rate \( a \)):

\[
\text{for all training data } In = (in_1, in_2, \ldots in_m) \text{ do }
\]

\[
\text{for all neurons } j \text{ in the competition layer do }
\]

\[
\text{compute } \sum_i w_{ij} \cdot in_i;
\]

\[
\text{determine the winning neuron } S \text{ as the one for which } \sum_i w_{is} \cdot in_i \text{ is maximal;}
\]

\[
\text{for all neurons } j \neq S \text{ and all } i \text{ do } w_{ij} := w_{ij}
\]

\[
\text{for all } i \text{ do } w_{is} := w_{is} + a \cdot \left( \frac{in_i}{m} - w_{is} \right)
\]

Here \( m \) is the number of active input neurons. The winning neuron gets a weight bonus for the active neurons on the cost of a weight loss for inactive neurons.
Competitive Learning (5)

- Analysis:
  - If $\text{in}_i = 0$ then $w_{is}$ decreases
  - If $\text{in}_i = 1$ and previous $w_{is} < 1/m$ then $w_{is}$ increases

- Consequences:
  - The weights $w_{is}$ are now closer to $\text{in}_i$
  - $\sum w_{is} = 1$, because
    - $\sum$ weight changes of $w_{is}$ is:
      $\Delta = \sum_i a \times (\text{in}_i/m - w_{is}) = a \left( 1/m\sum_i \text{in}_i - \sum_i w_{is} \right)$
      $= a(1 - 1) = 0$
    - That means, the weights are still normalized. If the same input is presented again, the same neuron will also win again.
Termination

- Does competitive learning terminate?
- In general not (it has this in common with clustering methods).
- What often can be reached is an equilibrium.
- This means that for randomly presented inputs the expected change of the weights is zero.
- But even an equilibrium cannot always be reached.
- However, when one stops at an arbitrary point one can get useful information.
A Geometric View (1)

- Consider a training vector set whose vectors all have the same length, and suppose, without loss of generality, that this is one. Recall that the length \( |x^2| \) of a vector \( x \) is given by

\[
|x^2| = \sum x_i^2
\]

- A vector set for which \( |x^2| = 1 \) for all \( x \) is said to be normalized. If the components are all positive or zero, then this is approximately equivalent to the condition \( \sum x_i = 1 \)

- Since the vectors all have unit length, they may be represented by arrows from the origin to the surface of the unit (hyper)sphere.
Geometric View (2)

Vectors on Unit Hyper sphere
Suppose now that a competitive layer has also normalized weight vectors. Then these vectors may also be represented on the same sphere.

Blue: Input vector
the neuron that wins
is the one that is
closest to the
input vector.
It moves even closer
to the input vector.
Geometric View (4)

Winner move

Result: Clustering
Example

• Suppose the inputs corresponds to the coordinates of a world map and the distribution favors cities around Calgary and in South America.
• Suppose furthermore all the weights correspond to the city of Calgary and the input are the coordinates of Buenos Aires.
• The winner would be a small city close to and south of Calgary, say De Winton.
• As a bonus, De Winton would move south, may be as far as to Spokane.
• For new inputs from the area of Calgary only cities close to Calgary would win.
Relation to k-Means

• All patterns that cause to fire the same nodes are clustered together.
• The weights wij of the i\textsuperscript{th} node correspond to the i\textsuperscript{th} mean.
• Consequence:
  – Competitive learning is incremental k-means learning
  – k = number of top nodes.
Part 6

Topological Mappings
Winner Take All versus Topology

• In the winner-take-all network, there is no spatial relationship between the clusters identified by the neurons.
• What would be more useful is the situation where the output space of the network was arranged in some meaningful manner because then some neurons “close” to the winner could also benefit.
• This could provide other benefits in terms of robustness, etc. The output space of a unsupervised or self-organizing network is composed of the statistically common features present in the input data.
Kohonen Maps

• Kohonen maps and self-organizing maps (SOMs) are synonyms.
• The basic idea behind the Kohonen network is to set up a structure of interconnected neurons which compete for the signal (as in competitive learning). While the structure of the map may be quite arbitrary, most implementations support only rectangular and linear maps.
Idea of Kohonen Maps (1)

What do we mean by "close" and "far"? We can think of organizing the output nodes in a line or in a planar configuration:

Input and output space may have different dimension: Outpace has mostly a reduced dimension. The goal is to train the net so that nearby outputs correspond to nearby inputs.
Kohonen Map (3)

Input space $V$

Output space $A$ usually of lower dimension

Input vector $x_1 x_2 \ldots x_n$
Kohonen Maps and Topology (1)

- Kohonen's SOMs are a type of unsupervised learning. The goal is to discover some underlying structure of the data.
- Kohonen's SOM is called a topology-preserving map because there is a topological structure imposed on the nodes in the network. A topological map is simply a mapping that preserves neighborhood relations.
- A discontinuity is called a topological defect.
- In the nets we have studied so far, we have ignored the geometrical arrangements of output nodes. Each node in a given layer has been identical in that each is connected with all of the nodes in the upper and/or lower layer.
Kohonen Maps and Topology (2)

- The SOM defines a mapping from the input data space spanned by $x_1..x_n$ onto a one- or two-dimensional array of nodes.
- The mapping is performed in a way that the topological relationships in the n-dimensional input space are maintained when mapped to the SOM.
- In addition, the local density of data is also reflected by the map: areas of the input data space which are represented by more data are mapped to a larger area of the SOM.
Feature Maps (1)

- The orientation tuning over the surface forms a kind of map with similar tunings being found close to each other. These maps are called *topographic feature maps*.

- A map is a way of representing on a two-dimensional surface, any real-world location or object. Many maps only deal with the two-dimensional location of an object.

- Topographic maps have on the other hand do deal with the third dimension by using contour lines to show elevation change on the surface of the earth, (or below the surface of the ocean).
Feature Maps (2)

- These nets consist of a layer of nodes each of which is connected to all the inputs and which is connected to some neighbourhood of surrounding nodes.
- It is possible to train a network using methods based on activity competition in a such a way as to create such maps automatically.
Feature Maps

- The reduced dimensionality of the output space can be called the feature space; and an additional advantage would be if this space was organized topologically.
- That is similar features or clusters were spatially close and dissimilar ones far apart. The formation of distinct ordered clusters in the feature space implies the existence of two opposing effects. Illustration:

\[
\begin{align*}
\text{X: input vector} \\
o: weight vector
\end{align*}
\]
Meaning of the Topologies

• Input space V: Signal similarity
• Network Topology: Given by the network connections
• Goal: Correspondence:
• Neurons with similar signal input can communicate via short connections.
• That means, we need a map: Input → Network that preserves neighborhoods
• Therefore: The map has to have knowledge about the task.
Kohonen Map Detailed (1)

- Suppose the input vectors are \( x = (x_i) \) and the weights are \( w = (w_j) \) connecting the input vector to output node \( j \). The Kohonen maps are (signal space to network):
  \[ \Psi : V \rightarrow A. \]

- For each neuron there is an associated weight vector \( (w_j) \) connecting the inputs to neuron \( i \).

- All weight vectors are initially set to small random values. The only strict condition is that all weights are different.

- The winner neuron \( s(x) \) is determined by
  \[ s(x) = \text{argmin}_{i \in A} \| (x - w_i) \| \]
  as in competitive learning.
Kohonen Map Detailed (2)

• The set

\[ R_i = \{ x \mid \| (x - w_i) \| \leq \| (x - w_j) \|, j \in A \} \]

forms the receptive field (also called Voronoi region) of the neuron i.

• In a self organizing map (SOM) the neurons are arranged a priori in a fixed grid where in some dynamic models the topological structure is an object of learning in itself.
Kohonen Map Detailed (3)

- Objective of the learning process in the SOM is a distribution of weight vectors across the input space by adaptation of weight vectors.
- A stimulus (input) $x$ is randomly presented to the network according to a certain data distribution $P$.
- There are different ways of giving rewards to the winners, i.e. different ways of learning.
Kohonen Map Detailed (4)

• A common way of learning in the SOM has the adaptation rule

$$\Delta w_i = \eta h_{SOM}(x, \sigma)(x - w_i)$$

where the neighborhood function is usually the Gaussian shape and \( s(x) \) is the winner for stimulus \( x \):

$$h_{SOM}(x, \sigma) = \eta \exp(-\frac{||i-s(x)||^2}{2\sigma^2})$$

evaluated in the output space \( A \).

• \( \eta \) is a learning parameter
• \( \sigma \) describes the range of the neighborhood.
• Both parameters will be varied.
The black neuron in the center is the winner. The squares describe the shrinking neighborhoods.

In general: The forms of the neighborhoods may be arbitrary geometric objects.
Shrinking Neighborhoods (2)

- Decreasing the neighbourhood ensures that progressively finer features or differences are encoded.

- The gradual lowering of the learn rate ensures stability (otherwise the net may exhibit oscillation of weight vectors between two clusters).

- Both methods have to ensure that the learning process has some kind of convergence or stability.
The Learning Process (1)

Learning of Kohonen nets:

As for competitive learning:

\[
\text{for all training data IN} = (in_1, in_2, \ldots) \text{ do}
\]

\[
\text{for all Kohonen neurons } j \text{ do}
\]

\[
\text{Compute } \sqrt{\sum_{i=1}^{n} (in_j - w_{ij})^2}
\]

Determine neuron S for which this value is minimal

\[
\text{for all Kohonen neurons } j \text{ from the neighborhood of S do}
\]

\[
w_{ij} := w_{ij} + a \ (in_j - w_{ij})
\]
Discussion

- Learning of Kohonen nets means:
  - All Kohonen neuron from the chosen neighborhood of S adapt their weights in the direction of the learned vector IN
  - The size of the neighborhood is varying, in general the will shrink over time
  - After successful learning there will be groups of Kohonen neurons that have similar weight vectors

⇒ The net finds categories of learning data, i.e. some clustering.
Inhibition

Learning by inhibition

- Idea:
  - Neurons are sending each other signals for lowering the activities
  - This means:

- States of neurons change over time...
Inhibition: Mexican Hat Function

This function is popular but the success is doubtful

\[ 2e^{-ad^2} - e^{-bd^2} \]

\[ d = \text{Dist}(\text{Unit}_i, \text{Unit}_j) \]

\[ a > b \]
Dimension 1

• In this case both, the input space and the topology of the network are both of dimension one.

• In this case the neighbor function \( h \) can be simplified as a step function:

\[
h_{\text{step}} = 1 \text{ for } \|i-s(x)\| = 0 \text{ or } i = s(x).
\]

• In this case a mathematical proof of the convergence for learning weights can be given.
Topology Preservation

- Task: We present data to the input space as query q and search for similar data, i.e. the k nearest neighbors to q. But: We want to perform the search in the network!
- For complete topology preservation: Retrieval is correct and complete.
  - But: Costs of topology generation is high.
- For incomplete topology preservation: Adaptation of the search depth of the algorithm.
  - But: Dependency from the degree of the topology preservation.
- A good topology preservation allows efficient nearest neighbor search in net instead search in the whole data set.
SOM: Geometric Interpretation

- Topological relations between clusters do almost remain.
- Neighbor units correspond to neighbor clusters.
- Data space is mapped to the 2-dim. structure ("map")
- Provides visualisation of high dimensional data
- The 2-dim. structure is projected into the high dimensional space
A General Aspect: Dimension Reduction

In reality we encounter n-dimensional situations (e.g. n = 3)

This means:

We have n data v = (v_1,..., v_n)

These data are in a topological space and neighborhoods contain dependencies.

Wanted: Replacement by

r = (r_1),...,r_m), m < n

Motivation: There are dependencies between data.

Reduction allows visualization.
Linear Approach (Error Analysis)

Search for m-dimensional approximating hyper plane

Representation from data point $v$:

$$v = w_0 + \sum w_i r_i + d(v)$$

$d(v)$: Error

Dimension reduction is not simply to omit coordinates. The principal axis is one-dimensional but a composition of the two axes.
Principal Curves (1)

The idea of the principal curve $f$ (principal area) is to minimize

$$\int \| x - \tau_f \|^2 \, \text{Prob}(x) \, dx$$

where $\tau_f(x)$ denotes the projection index of the sample $x$ onto the curve $f$.

This curve is no longer a linear function!

Kohonen Algorithm:
Approximation of the principle curve (area)

$$\|(x - \lambda_f) \|^2$$
Principal Curves (2)

- This means the principal curve runs through the “middle” of the data distribution.
- If the curve is discretized, we get a sequence of points, i.e., the neurons of a 1-dimensional Kohonen chain.
- The position of each neuron must be in the center of its receptive field (i.e., the x where the neuron is the winner).
- But this is exactly what the Kohonen algorithm does:
  - Moving the neuron positions towards the average of all samples in the receptive field of the respective neurons.
Partitioning

The weight vectors in $A$ partition $A$ into convex polygons $R_i$, such that they describe the NN-regions.

(Voronoi-Regions, Receptive Fields)

Topological defect:

- $R_i$ and $R_j$ are neighbors
- The mapping $A \rightarrow V$ is discontinuous at $w_i$ and $w_j$. 
SplitNet: Construction (1)

Main principle: Splitting of Kohonen chains
E.g.: Data from $[0;1]^2$ and 1-d-chain: Dimension conflict

Splitting Criteria:
- Topological defects
- Deletion of neurons
- Local variation of edge length/quantization error
SplitNet: Construction (2)

(a) T_1 | C = \{1, \ldots, 7\}
\[ c = (0.4; 0.5) \]
\[ s = 7 \]

(b) T_1 | C = \{\}
\[ c = (0.4; 0.5) \]
\[ s = 7 \]

T_2 | C = \{1, 2, 3, 4\}
\[ c = (0.31; 0.68) \]
\[ s = 4 \]

T_3 | C = \{5, 6, 7\}
\[ c = (0.51; 0.26) \]
\[ s = 3 \]
The Topological Defect

- The chain is a one-dimensional topology on the neurons.
- The chain in a) is folded into the two-dimensional input space $[0,1]^2$. For each neuron $N_k$ the region $R_k$ is given by dotted lines.
- The main directions of the chain correspond to the principal curves of the input data.
- This leads to a topological defect: The neurons $N_1$ and $N_7$ are adjacent in the input space but not in the chain.
Splitting the Net

- We cut the chain in a first half \( C_1 \) and a second half \( C_2 \).
- For pairs \((c^1_i, c^2_j)\) with \( c^1_i \in C_1, c^2_j \in C_2 \) we compute the midpoint \( c^\text{mid} = \frac{1}{2}(c^1_i + c^2_j) \) and present it as input vector to the net.
- If the first and second winner for \( c^\text{mid} \) just the points \( c^1_i \) and \( c^2_j \) we have discovered a topological defect.
- Then the conflict is resolved by splitting the chain at the midpoint, thus creating two new Voronoi regions and chain.
- This is shown in picture b).
- Remark: This is only a heuristics.
- This gives also rise to a tree structure.
SplitNet Formally

Insertion of neurons. bmu denotes the best matching unit, i.e., the winner. Goal: Reduction of the quantization error

$$E_{VQ_j} = \sum p(x)$$

$p(x)$: probability

$$\Rightarrow$$ Idea: Insert a new neuron where the local error is high:

$$E_{bmu}^{t+1} = E_{bmu}^t - \epsilon_{bmu}$$

with

$$d_{bmu} = ||x_i - w_{bmu}||$$

One can show:

$$\langle d_j^2 \rangle$$ approximates $E_{VQ,j}$

$$\Rightarrow \langle E_j \rangle$$ approximates $E_{VQ,j}$

Estimation of the chain error:

$$E_{rel}^c = \frac{h_L \sum_{j \in C^F_j} E_j}{s_c}$$

Insert where

$$e_{insert} = \frac{E_{rel}^c}{1 + S_c}$$

is maximal

$$\Rightarrow$$ Reduction of the total error is maximal
Criteria:
- static/dynamic network
- dimensions preserving/-reducing

Topology preserving
Network Models

Static Models
- Dimension reducing
  - SOM ($D_V > D_A$)
- Dimension preserving
  - SOM ($D_V = D_A$)

Dynamic Models
- Dimension reducing
  - GCS
  - GSOM
  - TS-SOM
- Topology constructing
  - Size adapting
    - Growing Neural Gas
  - Hierarchy learning
    - SplitNet
  - NeuralGas
  - TRN
Applications of Kohonen Nets

- There are a huge number of applications. Some examples:
  - Increasing contrasts in image processing
  - Separating words in speech recognition
  - Visualization of satellite data in color plots (i.e. mapping of high dimensional data to a color cube)
  - Learning of control data for robot arms (mapping onto a 3-dimensional network)
  - Quantization of input signals
Part 7

Adaptive Resonance
The Plasticity-Stability Problem

• Dilemma: How can one learn new associations without forgetting old ones?
• Plastic: Remain adaptive to new significant events:
  – Learn new associations
• Stable: Remain irrelevant to insignificant events:
  – Do not forget something important
Adaptive Resonance Theory (ART) (1)

- Adaptive Resonance Theory (ART) was developed by Grossberg as a theory of human cognitive information processing. It was the result of an attempt to understand how biological systems are capable of retaining plasticity throughout life, without compromising the stability of previously learned patterns. Somehow biologically based learning mechanisms must be able to guard stored memories against transient changes, while retaining plasticity to learn novel events in the environment. This tradeoff between continued learning and buffering of old memories has been called by Grossberg the stability-plasticity dilemma.
Adaptive Resonance Theory (ART) (2)

• The simplest version of ART is quite similar to Kohonen networks (indeed, ART is a bit simpler)
• ART can get extended into ARTMAP which adds in an interesting system of other components. This allows it to perform the sort of training as in backprop networks, but in a very different manner.
• The vital difference is that ART is capable of learning based on a single exposure to an event, and does not show the catastrophic forgetting that backpropagation networks do.
Example

• Suppose a backpropagation network is shown one example of, say, a poisonous mushroom, and 400 examples of non-poisonous mushrooms. Trained on this data set, it will not be able to identify poisonous mushrooms, even if it sees that same mushroom again.

• Even if it does manage to learn a distinction between poisonous and non-poisonous, if it then gets trained on 4000 new examples of non-poisonous mushrooms, it is likely to 'forget' about the poisonous ones. That is, once the data set it is being trained on changes, it is unlikely to continue to work on the old problems.
Types of ART (1)

- Art 1:
  - Unsupervised learning (clustering)
  - Binary inputs
  - Encodes new inputs by adjusting weights
  - Bottom up weights are an adaptive filter that generate classifications
  - Top down weights are used for testing: They compare inputs and actual network representation
  - It is stable
Types of ART (2)

• ART 2:
  – Accepts continuous input patterns
  – Has many variations

• Fuzzy ART:
  – Like ART 1 but replaces input operations by fuzzy logic and allows in this way continuous input patterns.

• ARTMAP and FUZZY-ARTMAP:
  – Combine ART 1 and Fuzzy ART with associative memory resulting in a supervised learning system.
ART: Principal Architecture (1)

• Three levels: F0, F1, F2 where
  – F0 takes input and its complement
  – F1 receives input from F0 (bottom up) and is modified by the active cluster in F2 (top down)
  – F2 stores the active cluster in a single active node

• F1 and F2 are completely connected
ART: Principal Architecture (2)

- **F0**: (input, input<br>\(c\))
  - \(\text{Input} = (x_1, ..., x_n, 1-x_1, ..., 1-x_n)\)
  - Complemented inputs avoid normalization

- **F1**: \(\sum w_{ij} = 1\)
  - \(2n\) nodes

- **F2**: \(m\) nodes

- **Summation**
  - \(\sum w_{ij} = 1\)
Processing in Fuzzy ART (1)

- Activation in node $j$ of F2 for input $I$:

$$a_j(I) = \frac{I \wedge w_j}{\alpha + w_j}$$

$$\lvert I \wedge w_j \rvert = \lvert \sum I_i \wedge w_{ij} \rvert$$

$$\lvert w_j \rvert = \lvert \sum w_{ij} \rvert$$

$\wedge$ is the fuzzy AND (e.g. min); $\alpha$ is a constant, usually a small number.

Therefore:

The activation at node $j$ is the combination of the input $I$ and the weights leading to $j$. 
Processing in Fuzzy ART (2)

• The node with the largest $a_j$ value (the winner) best represents the input pattern.

• When the winner node $j$ is determined:
  – activation $a_j := 1$ and $a_i = 0$ for all other $i$.

• Resonance means: The winning nodes $j$ influence the lower nodes in $F1$

$$ I \land w_j \geq p, \text{ where } p \text{ is a threshold.}$$

• The parameter $p$ (called the vigilance parameter) determines how similar patterns must be in order to be in the same cluster.
Processing in Fuzzy ART (3)

- Learning:
  \[ w_j^{new} = \beta (I \wedge w_j^{old}) + (1 - \beta) w_j^{old} \]
- \( \beta \) is a learning rate, \( \beta = 1 \) means fast learning; small values mean the old weights \( w_j^{old} \) survive.
- Slow learning is advisable if noise is present.
ART: Geometric Interpretation

• „Vigilance“ corresponds to circle type areas around prototypes
• If a new point is outside: a new cluster is introduced
• Clusters remain stable (Grossberg: „stability-plasticity dilemma“ is solved)
• Number of clusters can grow
Vector Quantization (1)

- Vector quantization is a classical quantization technique from signal processing which allows the modeling of probability density functions by the distribution of prototype vectors.
- It was originally used for data compression.
- It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by some “prototype”.
Vector Quantization (2)

• Technically, the basic idea is to code or replace with a key, values from a multidimensional \textit{vector space} into values from a discrete \textit{subspace} of lower dimension.

• In vector quantization, we assume there is a codebook which is defined by a set of $M$ prototype vectors. ($M$ is chosen by the user and the initial prototype vectors are chosen arbitrarily).
Vector Quantization (3)

An input belongs to cluster $i$ if $i$ is the index of the closest prototype (closest in the sense of the normal euclidean distance). This has the effect of dividing up the input space into a **Voronoi tessellation**.
Vector Quantization (4)

- Vector quantization can be done by different methods:
  - Clustering a k-means
  - ART
  - competitive learning.
  - Kohonen nets

- The goal always is to "discover" structure in the data by finding how the data is clustered.
Vector Quantization (3)

Voronoi tessellation as vector quantization

- Vector quantization is one example of competitive learning.
- The **Kohonen** algorithm realizes a VQ but for that purpose it has several disadvantages.
Part 8

Hopfield Nets
and
Boltzmann Machines
Pattern Recognition

- First task: Store an image in a neural net. That is easily possible if the activities of the neurons correspond to the values of the pixels.

- A simple problem: Retrieve the image. Trivial answer: Give the image to the network and hope that the activities terminate and you get the original image back.

- Much more refined: You give an inexact copy to the network, will it also terminate with the original image? Can you do such things with several images? That means: What is the storage capacity?
Examples

- There is a database of images of criminals. There was a robbery in a supermarket, the person was recorded on a video tape. Is this person one of the persons in the data base?

- We want to identify handwritten texts and have examples of each letter in a data base of letters.

- We have images of standard objects on a street or in a landscape. We have also images from a satellite. What are the images on the photos?
The Hopfield Net Idea

- The activations of the neurons of the net correspond to the values of pixels of the image.
- We store such an image in the net, i.e. fix the activations.
- Then we give another image (a noisy version of the first image) as input to the net.
- The McCulloch-Pit neurons start to work sequentially and change their activities.
- It can be proved that this process always terminates.
- But at which image??
The Hopfield Net

- Characteristics of the Hopfield net:
  - McCP neurons with activities +1, -1.
  - All neurons are input as well as output neurons (there are no hidden neurons) and they are fully connected but $w_{ii} = 0$ (no self influence).
  - Weights are symmetric: $w_{ij} = w_{ji}$
  - Neurons are updated asynchronously, i.e. at each time only one neuron $m$ is active if $\sum a_i \cdot w_{im} \geq \theta_m$
  - The objects of learning are activity states, not weights; the learning process terminates if no neuron can be active.
  - The process is controlled by a so-called energy function.
The Energy Function

• The *energy* function is:

$$E := -\frac{1}{2} \sum_n \sum_{n' \neq n} a_n a_{n'} w_{nn'} + \sum_n a_n \theta_n$$

• We call the energy function also the *error* function.

• One can show:
  
  – a) The energy function has a lower bound:

$$E \geq \sum_n \sum_{n' \neq n} w_{nn'}$$

  – b) If there is one neuron active and $E_0$ is the energy before the activity and $E_1$ is the energy after the activity then $E_0 - E_1 \geq 0$ and $E_0 - E_1 = 0$ can only happen if the neuron in question became active, what can happen only finitely often.
Convergence Theorem

• Consequence (Convergence Theorem):
• For any initial activity state the net activity stops eventually, i.e. the process terminates.
• Proof: The energy function can only be a finite number of times at the same level; it can go down only a finite number of times.
• Remark: The energy function plays here the same role as the counting parameter when one want to prove the termination of a while loop.
Pattern Recognition with the Hopfield Net (1)

• Pattern (input patterns) $x$ are stored in a Hopfield net not as in a data base but in a certain way in the weights.

• Simple form of pattern recognition:
  – An input pattern $x$ is given as an input to the net. The retrieval is successful if the net activity terminates with $x$.

• More interesting is the non-trivial pattern recognition:
  – A disturbed version $x'$ of an input pattern is given as input to the net. The retrieval is successful if the activity terminates with $x$ (correctness).

• The convergence theorem guarantees the termination but not the correctness.
Pattern Recognition with the Hopfield Net (2)

- N = Number of neurons
- M = number of patterns, we assume M << N
- Patterns $x^k$ (k=1...M) : $x^k=(x_1^k,...,x_M^k)$
- Binary activations, but from \{+1,-1\}
- Determination of the weights (storage of patterns):

$$w_{ij} := \begin{cases} 
\sum_{k=1}^{M} x_i^k \cdot x_j^k & \text{if } i \neq j \\
0 & \text{otherwise}
\end{cases}$$

- Increase for same activity
- Decrease for different activity
- symmetric
- $w_{ii}=0$

We get:

$$w_{ij} := w_{ij} + x_i^{M+1} \cdot x_j^{M+1} \quad \text{for } i \neq j$$
Recognition Algorithm

- Put threshold $\theta := 0$
- Algorithm:
  1. Initially: Activities $a_j(0)$ at time $t=0$
  2. Iteration step: $a_j(t+1) = f \left( \sum_{i=1}^{N} a_i(t) \cdot w_{ij} \right)$
  3. Termination criterion: $a(t) = a(t+1)$

- We have a convergence theorem:
  For each initial values of the activities the iterations terminate

- Question: What is the final pattern??
Attractors

- Attractor states $a$ are fixed point of the net activity, i.e. they have no successor states, no neuron can be active anymore.
- Such states can be the terminating states for other input states $x$. The set of such states is called the attraction domain of the attractor $a$.
- The role of the attractors for pattern a recognition is that one wants that each stored pattern is the attractor for exactly all noisy versions of $a$.
- Unfortunately, this is not always the case. This will be analyzed next.
Attractors
Input of a Learning Pattern $x^k$

$$\text{net}_j := \sum_{i=1}^{N} x^k_i \cdot w_{ij} = \sum_{i \neq j}^{M} \sum_{l=1}^{x^l} x^k_i \cdot x^l_j = \sum_{l=1}^{x^l} \sum_{i \neq j}^{M} x^k_i \cdot x^l_j$$

- Consider the input of neuron $j$:

  $$\text{Signal}_j := x^k_j \cdot \sum_{i \neq j}^{M} x^k_i \cdot x^k_i = x^k_j \cdot (N - 1)$$

  Separate input into a signal term and a noise term:

  $$\text{net}_j = \text{Signal}_j + \text{Noise}_j$$

- **Signal term** = Part that comes from $x^k$ itself
- **Noise term** = Influences of other patterns

- Therefore:

  $$\text{Noise}_j := \sum_{l \neq k}^{M} x^l_j \cdot \sum_{i \neq j}^{M} x^k_i \cdot x^l_i$$
The Signal Term

- Assume: All noise terms disappear !!!
- If $N>1$, then
  - $x_j^k$ increases positively.
  - Consequence $x^k = a(0) = a(1)$.
  - Consequence: Each fixed point is recognized.
- If the input is pattern $y$ that differs at $r$ positions from $x^k$ then

\[
 net_j := \sum_{i=1}^{N} y_i \cdot w_{ij} = \sum_{i \neq j} y_i \cdot x'_i \cdot x'_j = \sum_{i=1}^{M} x'_j \cdot \sum_{i \neq j} y_i \cdot x'_i
\]

\[
 Signal_j := x_j^k \cdot \sum_{i \neq j} y_i \cdot x_i^k = x_j^k \cdot (N - 1 - 2r)
\]

- Therefore for $r < \frac{N-1}{2}$ the net is stable after the 1. step : $a(1) = a(2)$
The Noise Term

\[ \text{Noise}_j := \sum_{l \neq k} x^l_j \cdot \sum_{i \neq j} x^k_i \cdot x^l_i \]

- Up to an error with \(|\text{error}| \leq 1\) each summand corresponds to the scalar product of \(x^k\) and \(x^l\):

\[ \sum_{i \neq j} x^k_i \cdot x^l_i \]

- If all patterns are pair wise orthogonal we get:

\[ |\text{noise}_j| \leq M-1 \]

- Therefore: If \(M<<N\) and all patterns are pair wise orthogonal then the noise term is small compared with the signal term.

- Consequence: The attraction domain gets smaller
Convergence (1)

- The convergence theorem says that procedure not only approximates $a$ in the limit but in fact always stops.
- But where?
- Definitely at a minimum of the error function. This is in general a local minimum.
  - For each such minimum $M$ the set of all inputs that converge to this input is called the domain of attraction of $M$.
  - The fixed points of net are exactly the attractors.
Convergence (2)

- Unfortunately there are very many minima and an input may end up at a minimum that was not an input.
- E.g. for each attractor a the state -a is also an attractor: Because of the square in the energy function.
- In general, other attractors than the stored patterns are called *ghost states*. These are very hard to detect.
- If more patterns are stored in the net the chance that they are correct attractors decreases.
Interpretation for Images (1)

- Suppose each pixel in an image is either black or white (0 or 1).
- Each pixel is represented by some neuron, i.e. by its activation state.
- The assumption that two images are orthogonal is rather strange.
- The situation is: We have a data set of images (e.g. persons) and these images are stored in the net.
- Suppose now we present a slightly different image of one of these persons: The net operates and stops at an attractor.
Interpretation for Images (2)

• This can be:
  – The original image (wanted)
  – One of the other images
  – Some other image that happens to be an attractor.

• If we present exactly one of the original images then we get it back because it is a fixed point.

• For a varied image we get it back if and only if the variation is in the attractor domain of the original image.

• That means, the variation has to be sufficiently small.
Example

Suppose you have trained the Hopfield Net on these patterns:

```
1  2  3
```

Now we present the net a corrupted version of the digit “1”. The net works step by step until the attractor (the original) is reached:

```
;  i  I  1
```
How Many Images Can We Store?

• Notation:
  – Cor = the number of patterns the can be correctly retrieved
  – N = number of neurons
  – Cap = Cor/N (the capacity)

• Theorem: The maximum storage capacity of a Hopfield net is bounded above by
  \[ \frac{1}{4} \ln(N). \]
Storage in a Hopfield Net and in a Data Base

• In a data base each image is stored in a “different box” that can be located. Retrieval is by keys.
• In a Hopfield net each image is stored in “the whole net”. Retrieval is called “content addressable”.
• In a data base it is known how many images you can store, not in a Hopfield net.
• In a data base you can query only those images for retrieval that are stored.
• Hopfield net also “approximate queries”.
• If a Hopfield net is overloaded you can retrieve nothing.
The Boltzmann Machine (1)

- The Boltzmann machine is an extension of the Hopfield net.
- It can have hidden and non-hidden neurons. The latter ones can be input or output neurons or both.
- Weights are still symmetric: $w_{ij} = w_{ji}$ and may be 0; $w_{ii} = 0$.
- In contrast to the Hopfield net the Boltzmann machine
  - Is devoted to improving weights
  - Learning of weights is supervised
- The change of activities is stochastic, i.e. the weights describe probabilities and what is learned is probabilities (for the intended behavior of the net).
- This can be applied for optimization tasks.
The Boltzmann Machine (2)

- Basic idea: Escape a local minimum. Suppose the ball has a certain energy, then it will sooner or later come over the hill (with a certain probability) and reach the left valley.

- If the energy is then reduced it will not be able to go back.
The Boltzmann Machine (1)

- The net input for neuron m is $\text{net}_m = \sum x_i \cdot w_{im} - \theta_m$
- The state of $x_i$ is
  - +1 with probability $p$
  - -1 with probability $1-p$

where the probability distribution is of the form

$$p(x) = \frac{1}{1 + e^{\frac{x-\theta}{T}}}$$

- $T$ is called the temperature parameter. During the activity it can be changed. The probability function is a sigmoid function that approximates the threshold function for small $T$. 

-
Learning in a Boltzmann Machine (1)

• The neurons in the Boltzmann Machine are divided into
  – *visible* neurons $V$, and *hidden* neurons $H$.
• The visible neurons are those which receive information from the environment.
• The training set obtains vectors in $V$. These are *fixed states* describing the intended input-output relations. During the training these states are in fact fixed. The probability distribution over the training set is denoted $P^0(V)$.
• If the Boltzmann machine now runs with visible states free then another probability distribution $P(V)$ is obtained.
Learning in a Boltzmann Machine (2)

- The fixed states are the teaching input.
- There is no comparison with „output“ and „teaching input“, learning takes place as in the Hopfield net by minimizing an energy function.
- What is learned is a probability, namely that the teaching input is approximated well.
- It is **not** a causal generative model in which we first pick the hidden states and then pick the visible states given the hidden ones.
- Instead, everything is defined in terms of energies of joint configurations of the visible and hidden units.
Learning in a Boltzmann Machine (3)

Goal: Approximate the distribution $P_0(V)$ of the fixed states by the distribution $P(V)$ obtained after learning.

As an analog to the energy function we use the Kullback-Leibler distance $G$:

$$G = \sum P(v) \ln \frac{P_0(v)}{P(v)}$$

This distance depends on the weights and should be minimized.
Learning in a Boltzmann Machine (4)

- That means: Maximize the product of the probabilities that the Boltzmann machine assigns to the vectors in the training set.
  - This is equivalent to maximizing the sum of the log probabilities of the training vectors.
  - It is also equivalent to maximizing the probabilities that we will observe those vectors on the visible units if we take random samples after the whole network has reached an equilibrium with no external input.
Learning in a Boltzmann Machine (5)

• We simplify the energy function by taking the threshold as 0 to $E := -\frac{1}{2} \sum_i \sum_{j \neq i} a_i a_j w_{ij}$

• In the same way as we treated the backpropagation we consider the partial derivates of the energy function from the weight where we obtain:

$$\frac{\partial \exp(-E/T)}{\partial w_{i,j}} = -\frac{1}{2} \cdot \frac{1}{T} \cdot a(i) \cdot a(j) \cdot \exp(-E/T)$$

• One can show for a global state $a$ where $P(a)$ is the probability of being in state $a$ and $p(n,n’)$ is the probability that both neurons are active:

$$\frac{\partial \ln P(a)}{\partial w_{i,j}} = \frac{1}{T} \cdot (a(i) \cdot a(j) - p(i, j))$$
Learning in a Boltzmann Machine (6)

• One can show:

\[
\frac{\partial \ln P(a)}{\partial w_{i,j}} = \frac{1}{T} \cdot (a(i) \cdot a(j) - p(i, j)) = \frac{\partial G}{\partial w_{i,j}}
\]

• Again using the gradient method one can show that this leads to the rule:

\[
\Delta w_{ij} = \eta \cdot (p^0(i, j) - p(i, j))
\]

• This rule leads to a (local) optimum for the probability of being in the intended state.
The Derivative is Simple

• The probability of a global configuration at thermal equilibrium is an exponential function of its energy.
  – So settling to equilibrium makes the log probability a linear function of the energy
• The energy is a linear function of the weights and states
• The process of settling to thermal equilibrium propagates information about the weights
Working of the Boltzmann Machine (1)

- Fix two numbers \( L_1 > 0 \) and \( L_2 > 0 \).
- The Boltzmann machine has an inner loop and an outer loop. **Inner loop:**
  - Choose a neuron \( n \) with some fixed probability \( p \) and accept its activity with probability

\[
Pa(i)(T) = \frac{1}{1 + e^{-\frac{\Delta E_{a(i)}}{T}}}
\]

- Here \( a(n) \) is the resulting net activity and \( \Delta E_{a(n)} \) is the difference between the new and the old energy.
Working of the Boltzmann Machine (2)

• **Outer loop**:  
  – Choose an initial value $T_0$ for $T$  
  – Apply the inner loop $L_1$ many times  
  – Decrease $T$ by a fixed increment  
  – Terminate if the state does not change $L_2$ many times.

• The outer loop is controlled from outside: The decrease of $T$ is not done by the net itself.

• There are different heuristics for choosing $T_0$, the increment and the values of $L_1$ and $L_2$. 
Working of the Boltzmann Machine (3)

• After sufficiently many learning steps the probability of a global state of the network will depend only upon that global state's energy.
• This means that log-probabilities of global states become linear in their energies.
• This relationship is true when the machine is "at thermal equilibrium", meaning that the probability distribution of global states has converged.
• This distribution is known from thermodynamics as the Boltzmann distribution; it gave the machine its name.
Discussion (1)

- It seems surprising that the steepest gradient can be expressed by local rules.
- Everything that one weight needs to know about the other weights and the data is contained in the difference of two correlations.
- Biologically, such rules are plausible because there one always learns locally.
Discussion (2)

• Advantages:
  – The performance is for many statistical applications extremely useful and it often outperforms simple backpropagation nets.

• Disadvantages: Training is time consuming
  – T
  – The time the machine must be run for in order to collect equilibrium statistics grows exponentially with the machine's size, and with the magnitude of the connection strengths.
Restricted Boltzmann Machines

- We restrict the connectivity to make inference and learning easier.
  - Only one layer of hidden units.
  - No connections between hidden units.
- In an RBM it only takes one step to reach thermal equilibrium when the visible units are fixed.
Summary

• Competitive learning, winner takes all, geometric interpretation
• Kohonen maps: Topologies, more than one winner neurons, Kohonen algorithms, principal curves, topology preservation
• Hopfield nets, energy function, convergence
• Boltzmann machines
Recommended References