Simulations in Statistical Physics

Course for MSc physics students

János Kertész

Lecture 8

How to find the ground state in a rugged energy landscape?

No simple optimization scheme works. NP-complete optimization problem: The computational complexity (roughly the CPU time needed to solve the problem) grows faster than any power of the system size (practically exponentially).
Many important problems
- Traveling salesman
- Graph partitioning
- Graph coloring

…
If one NP complete can be reduced to polynomial computing time, then all can be.

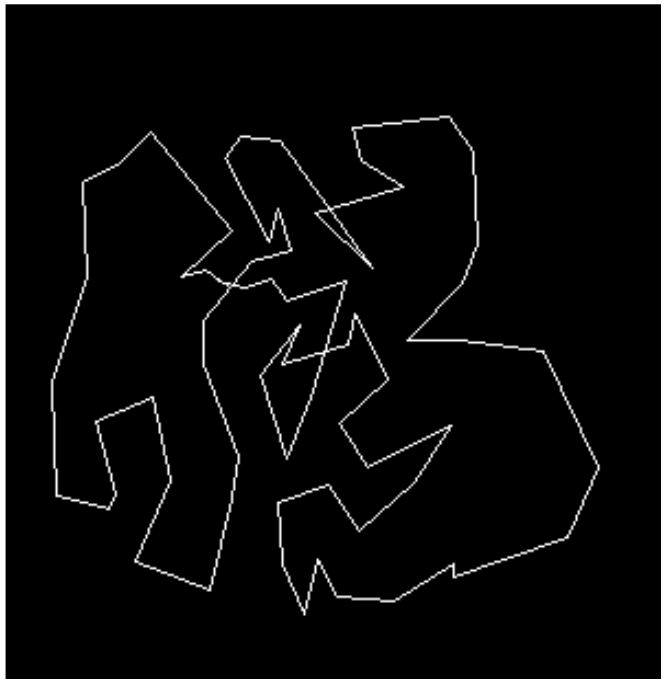
This is math. In reality we need good approximations, which work fast.

Simulated annealing

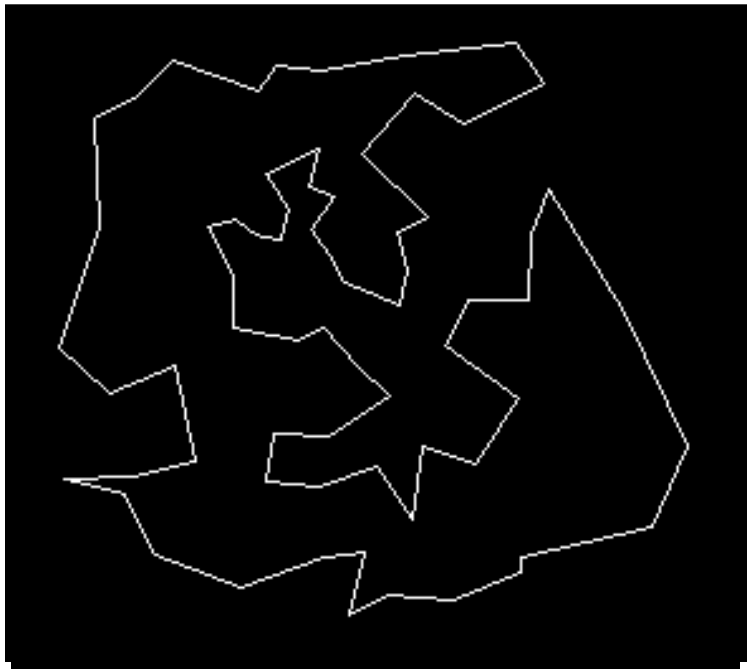
In a simple landscape the ground state can be found quickly by putting T=0 in a MC simulation. This method leads to spurious result in a rugged landscape as the system gets stuck in an improper minimum.

Simulated annealing uses the trick that a) it lets the system cool down slowly and b) it allows warming up again.

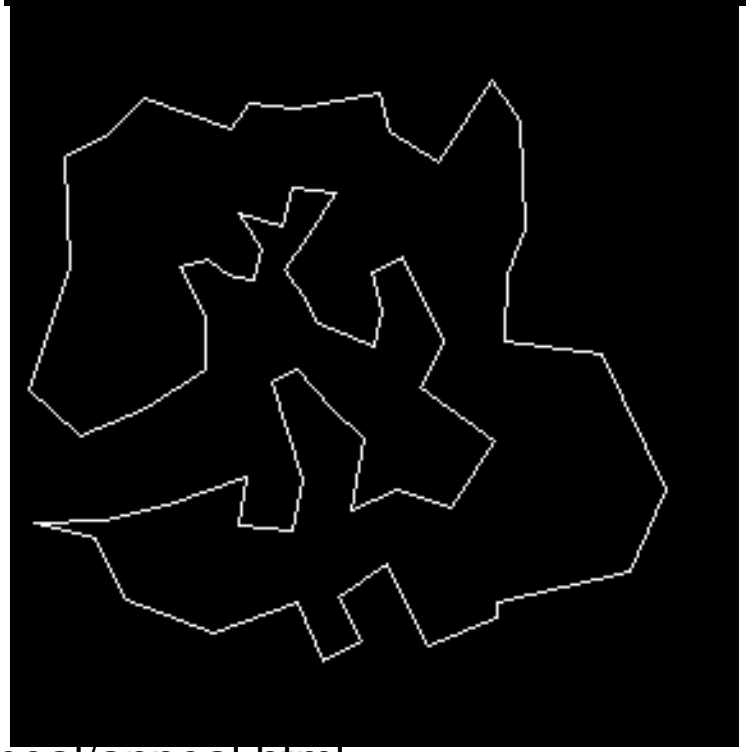
Traveling salesman problem: The salesman has to visit N cities randomly positioned in the plane such that it returns and the length of the path is minimum. (Basic problem e.g., in circuit design.)
Hamiltonian = path length
Elementary move = exchenge two cities in the path
Use Metropolis simulated annealing. (T ~ alcohol ☺)



Bad solution
„high temperature”



Good (low temp) sol’n



Even better sol’n obt’d after many thermal cycles

http://www.math.uu.nl/people/beukers/anneal/anneal.html

Another heuristic optimization method: **Genetic algorithms**

Let's learn from nature: Darwinian natural selection leads to the survival of the fittest – an optimization problem.

How is it „implemented":
- genetic code (genotype)
- copying by sharing the genetic code
- mutations
- those, who don't fit extinguish

Natural selection is very complicated due to changing environment – self-organization, adaption. The species themselves represent the environment at the same time.

Most optimization problems are simpler, there is constant criterion to fulfill.

Genetic algorithm:

The playground:

- Genetic representation
- Fitness function

1. Genetic representation.

In biology the sequence of the 4 bases provide the code. (4-letter alphabet).

In computer science we like the 2-letter alphabets. The genetic representation of the problem means that the states consists of a sequence *s* of bits of a given length. (E.g., the ground state of a spin glass.) Other genetic representations also exist.

2. Fitness function.

This defines the task: Find *s* such that $f(s) = \min$. *f* can be a complicated procedure (e.g., assign to the bits in *s* some attributes and operate on them.

The game:

Reproduction

An offspring has two parents. Create the genotype from the parents' genetic pool by recombination following a simple rule.

Chromosome s_1	111011 | 00100110110
Chromosome s_2	011011 | 11000011110
Offspring s_3	111011 | 11000011110
Offspring s_4	011011 | 00100110110

random crossover point (could be several)

Compare the fitness of the 4 different chromosomes, keep the two better ones and remove the other two ones.

Or wait until the whole generation is born and select afterward.

Mutation
Chromosomes are altered by random changes in the bits.

Important terms and hints

Chromosome: Carrier of the genetic representation
Gene: Smallest units in the chromosome with individual meaning
Parents: Pair of chromosomes, wich produce offsprings
Population: Set of chromosomes from which the parents are selected. Its size should be larger than the length of the chromosome
Selection principle: The way parents are selected (random, elitistic)
Crossover: Recombination of the genes of the parents by mixing
Crossover rate: The rate by which crossover takes place (~90%)
Mutatation: Random change of genes
Mutation rate: The rate by which mutation takes place (~1%)
Generation: The pool after one sweep.

The knapsack problem:

Given a pool of items with weights and values.
Put items into a „knapsack” sucht that the value is maximal but the knapsack has limited weight capacity. NP-complete problem.

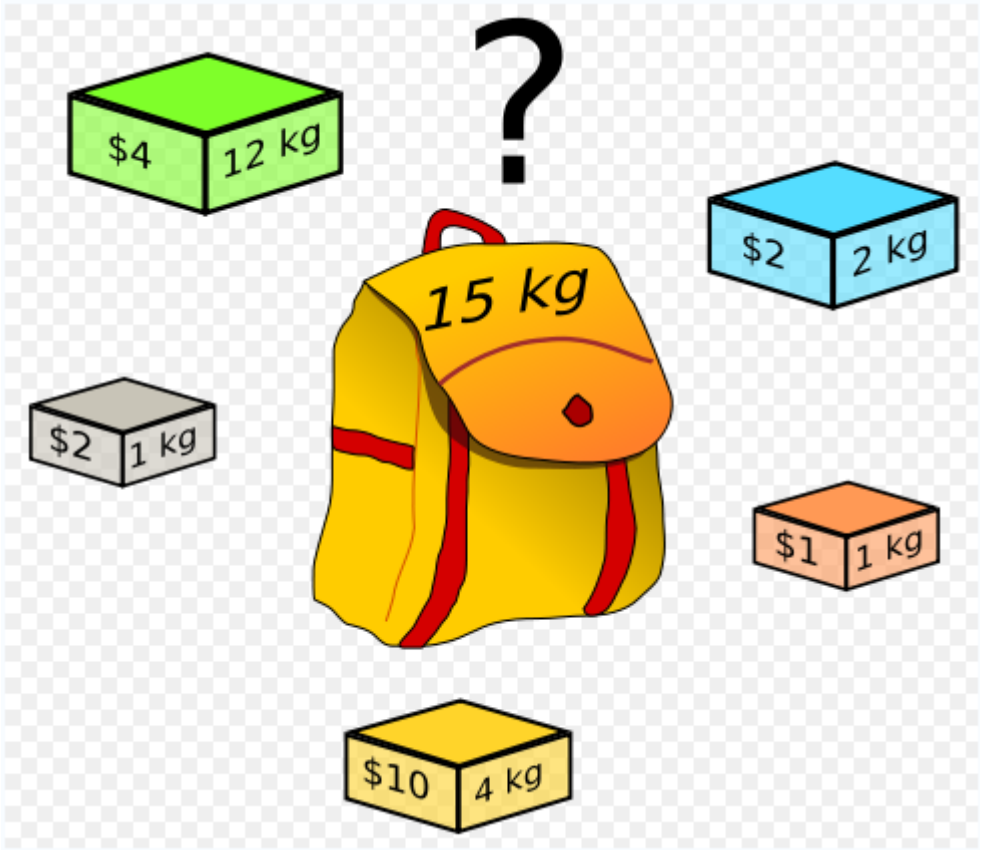
p_i value
w_i weight

Chromosome:
string of bits signalizing, which item is taken

Fitness function:

$$\sum_i p_i = \max \qquad \text{such that}$$

$$\sum_i w_i < w_{\max}$$

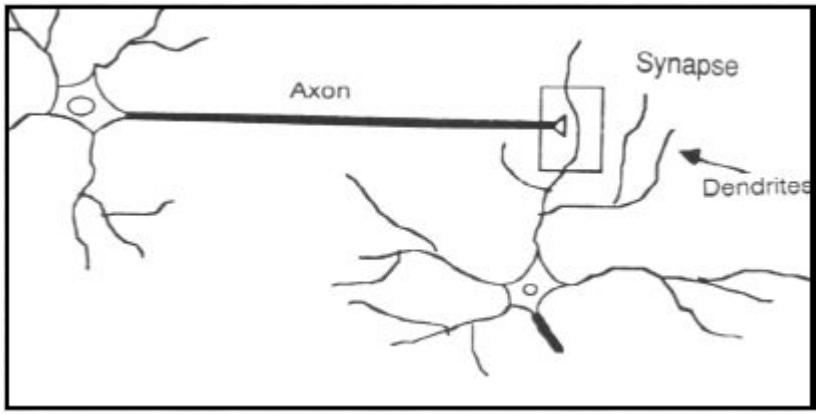


Machine Learning – Artificial Intelligence

How do we learn? A simple model: We see many times a pattern and then we are able to recognize it.

Content addressed memory

The information is not stored in one neuron but in a combination of them – neural network



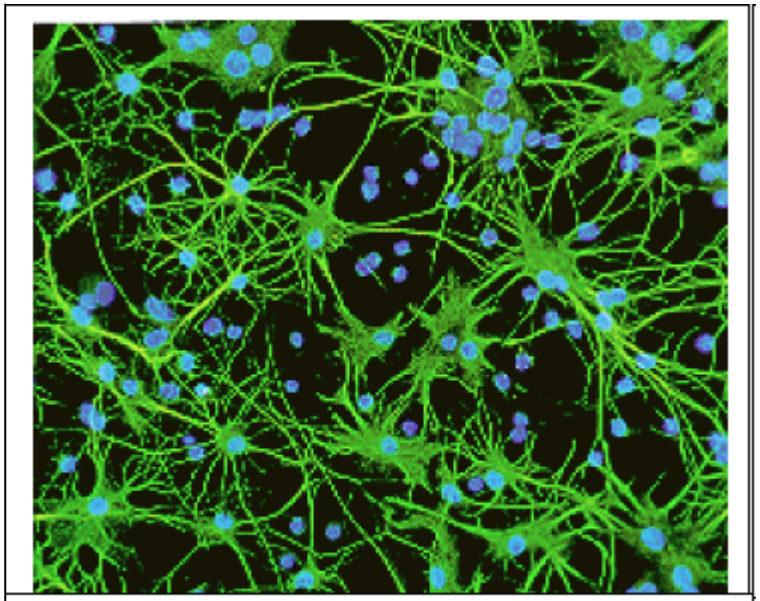
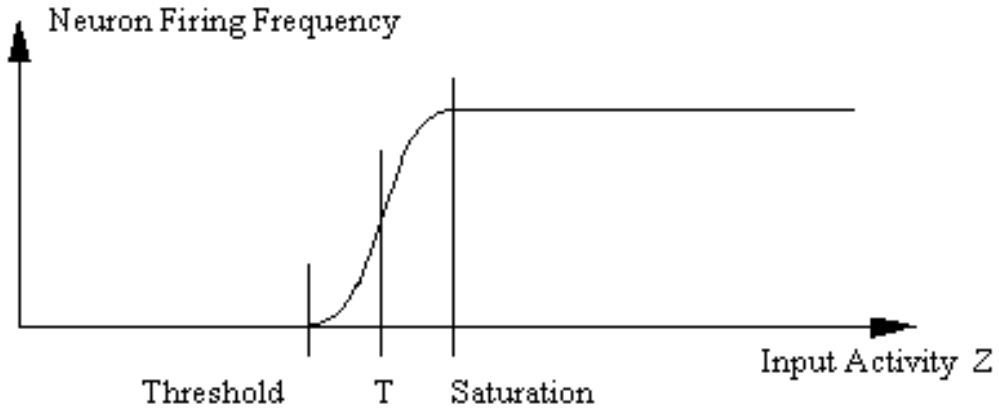


Figure 1 A view of the brain, with cells and axons showing in green and synapses showing

Neurons can be in an excited state (high firing frequency) or an a blocked state (low firing frequency). This depends on the input from the other neurons and is almost threshold-like.

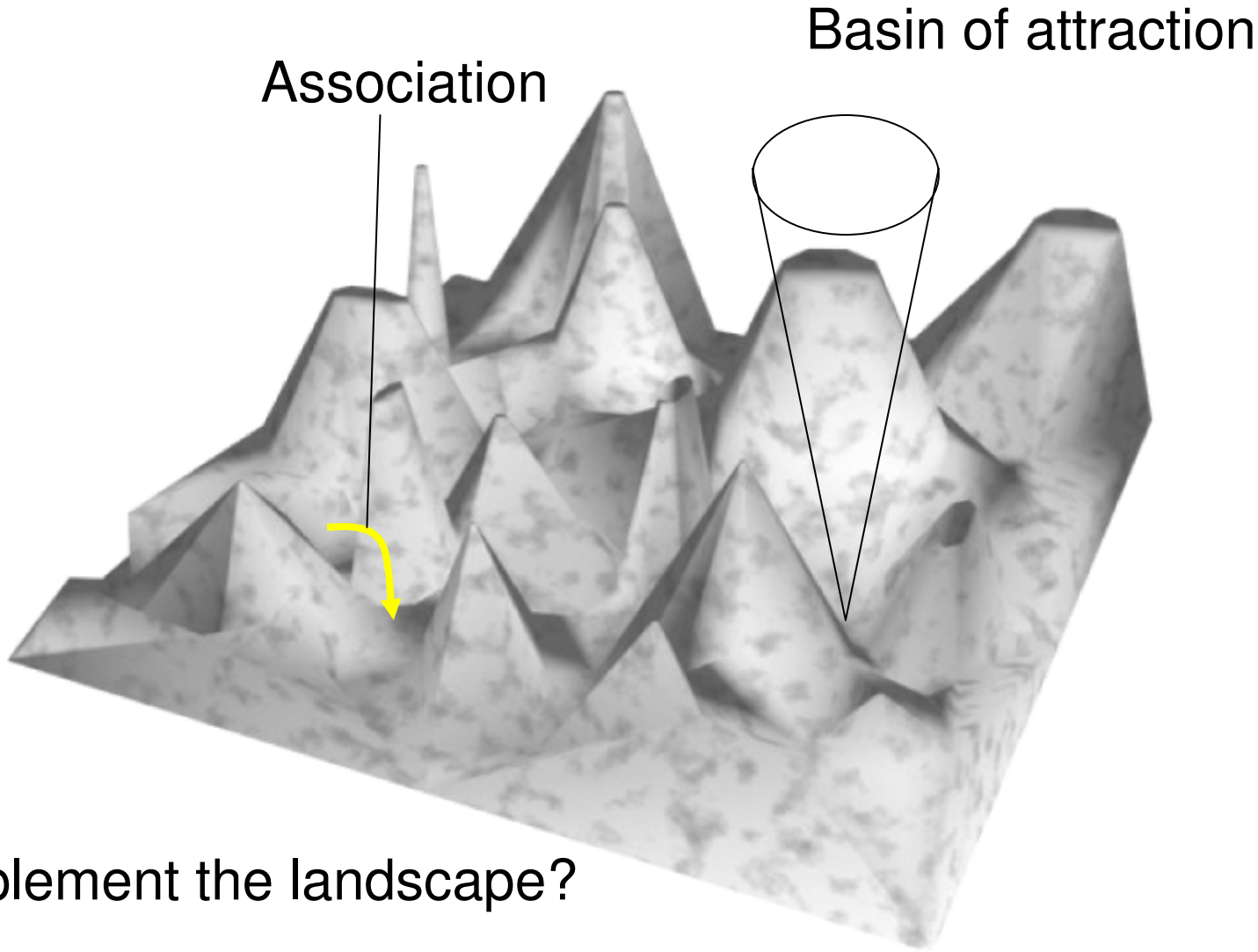


The pattern corresponds to a specific combination of excited and blocked neurons in the network.

Recognition: The learned pattern has a basin of attraction. Patterns similar to it lead to the learn pattern in a dynamic process.

Association: Transition between similar learned patterns

Identify the learned patterns with the minima of a rugged landscape!



How to implement the landscape?
Learning

A simple physics approach: **The Hopfield-model**

Neurons are Ising spins! (Two state variables)

They can block or stimulate each other corresponding to antiferromagnetic interactions ferromagnetic

Hebbian learning rule: When an axon is repeatedly used (for stimulating B by A) it gets stronger.

Define an „Ising spin glass”: $\mathcal{H} = -\sum J_{ij} s_i s_j$ where the individual couplings are at the beginning undefined.

There is a **learning** and a **retrieving phase**.

In the learning phase patterns are defined and „thought" to the machine. A pattern is a spin configuration $\{\xi_i\}$. We define for this pattern the ΔJ_{ij} such that the bonds are all satisfied: $\Delta J_{ij} = \eta \xi_i \xi_j$, which is the Hebbian learnign rule. If we want to teach N patterns and choose $\eta = 1/N$ we have finally:

$$J_{ij} = \sum_{\alpha} \Delta J_{ij}^{(\alpha)} = \frac{1}{N} \sum_{\alpha} \xi_i^{(\alpha)} \xi_j^{(\alpha)}$$

α : pattern index

In the retrieving phase we start from a configuration $\{s_i\}$, which we want to get recognized. Implementing a MC search for the ground state (does not have to be slow!) leads to a pattern for which the initial cofiguration is in the basin of attraction.

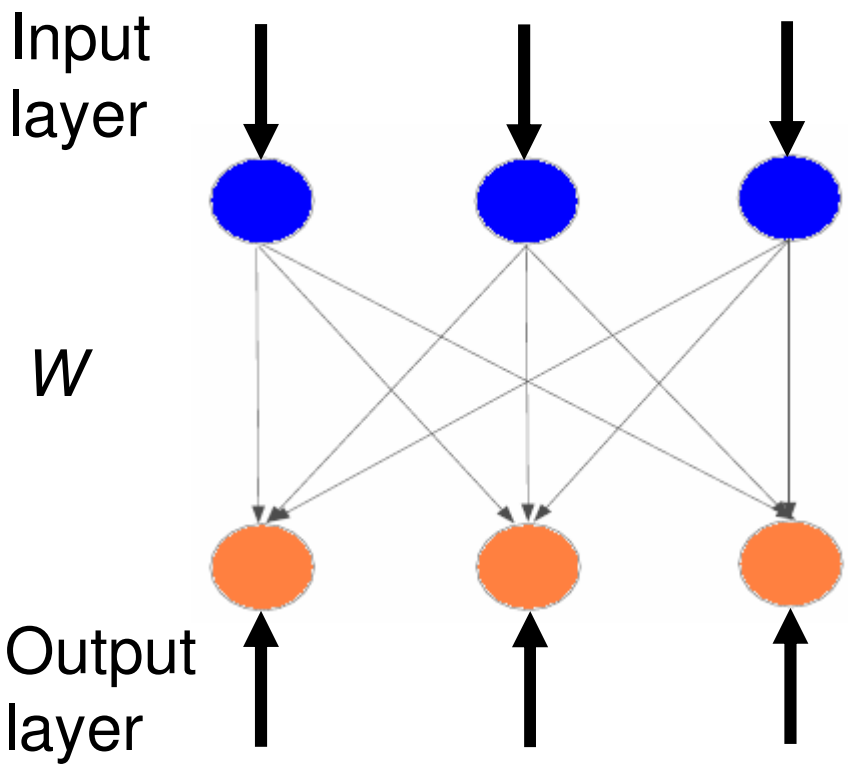
Patterns may interfer, this limits capacity. How different are two patterns? **Hemming distance**: ICOUNT(XOR(IS1,IS2))

Capacity max: the dimension of the space (L^d) (orthogonality) in reality much smaller. (# of neurons in human brain 10^{11})

http://www.theparticle.com/applets/ml/hopfield/

Other algorithms:
Feedforward network (**perceptron**)

Input vector *I* → output vector *O*, transformation matrix *W* (*i* X *o*)
$O = f(IW)$



Multi-layer perceptrons

Learning:

Initializing $W_{io} \in (-1,1)$ randomly

During training (supervised learning) these values are modified iteratively such that the difference between the guessed and the required sol'n is minimized. Define the error

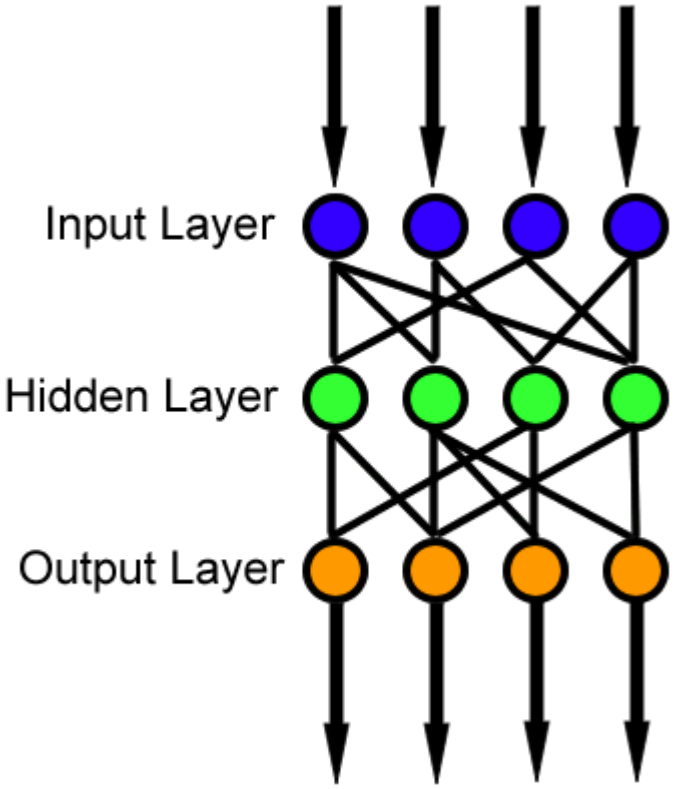
$$E(t) = T - O(t) = T - f(IW(t))$$

where *T* is the target (pattern).

Update $W_{io}(t+1) = W_{io}(t) + \varepsilon E_o(t)$

with ε being the „learning rate"

More complex tasks: **multilayer perceptrons + backpropagation**



Increasing nonlinearity, error adjustments

Broad use from evaulation of complex experiments to financial analysis

Pattern recognition
Artificial Intelligence

A further method: **Kohonen map**

Nonlinear mapping with different attractors corresponding to the patterns

Unsupervised learning: Find the inherent structure in the data

Classification/Clustering

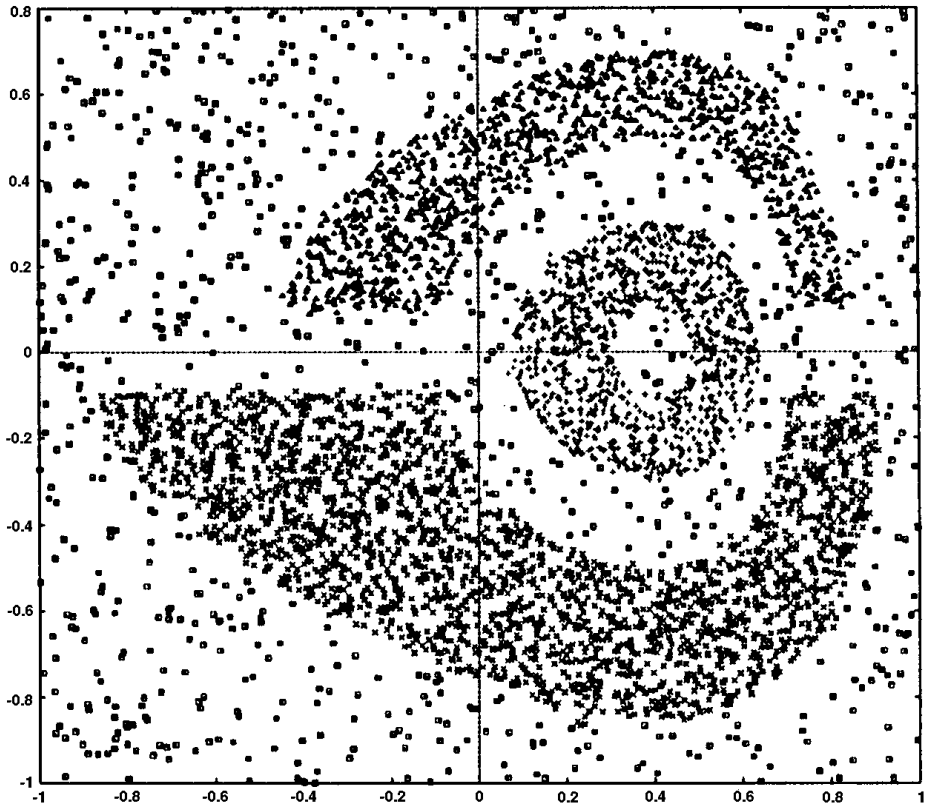
Given a set of entities, with similarity measure. Find the natural groups. Clustering, classification, taxonomy, community finding

Simple example: Set of points. Similarity: e.g., inverse Euclidean distance

The eye is a very good pattern recognizer – how to do it by computer?

$$\mathcal{H} = -\sum_{i<j} J_{ij}\delta_{s_i s_j} \qquad s_i = 1,.....,q$$

q-state potts model with inhom. ferromagnetic interactions.



The coupling is a positive monotonously increasing funtion of the similarity (decreasing fn of the distance, e.g.,

$$J_{ij} = Ke^{-\left\|\mathbf{r}_i - \mathbf{r}_j\right\|^2 / 2a^2}$$

At high temperatures the Potts spins point to random „directions”. As the temperature decreases, the close spins feel each other more than those farther away – they allign, building domains of spins in the same state. The whole sample is still paramagnetic but consists of super-spins. (The number *q* of states has to be large enough.)

The algorithm is a simple MC cooling. The points where a superspin is formed is signalized by a peak in the susceptibility (pseudotransition).