# Global Optimization



complicated multivariate potential surface
local and global minima
earlier methods go to local ones, here: trials to find global one

Many methods: random (Monte Carlo) search, stochastic gradient methods, simulated annealing, genetic algorithm...

## Simulated annealing

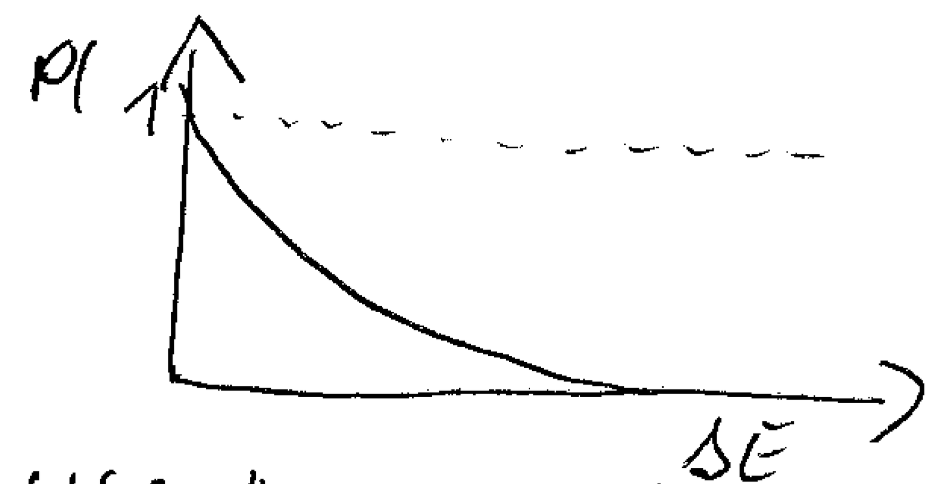
- define energy-like value $E(\underline{x})$ — $\underline{x}$ ← can be independent variable, parameter...
- suppose $P(E) \sim \exp(-E/kT)$ ≈ Boltzmann distributed
- define $\underline{x}_0$, calculate $E(\underline{x}_0)$
- define $T_0$ (≈ temperature)

→ create new $\underline{x}$ (close to $\underline{x}_0$), randomly, calculate $E(\underline{x})$ and $\Delta E = E(\underline{x}) - E(\underline{x}_0)$

if $\Delta E \leq 0$ $\underline{x}$ is the new accepted minimum point

$\Delta E > 0$ $\underline{x}$ is accepted with $P(\exp(-\Delta E/kT))$ probability

$\underline{x}_0$ remains, if $\underline{x}$ is rejected




└ repeat

- reduce sometimes $T \to 0$

$T \approx 0$ konverged

use Neumann's rejection method

≈ Monte Carlo simulation of condensed matter is very similar

# Global optimization

## Genetic algorithm (evolutionary algorithm)

(one of biologically inspired methods)

Previous min-max search: one starting point (except simplex)

Here: more starting points → result is also more points

Nomenclature: - points at a given iterative sets: population
- function to optimize: fitness (large is good)
- variables → stored binary, like genetic information

Next generation (population) is created by:
- reproduction: most fit points survive to the next
- crossing: new member is created by interchanging variables of two ones from the previous generation
- mutation: small changes in the values

Iteration stops, when fittness (average) does not ~~change~~ become better.

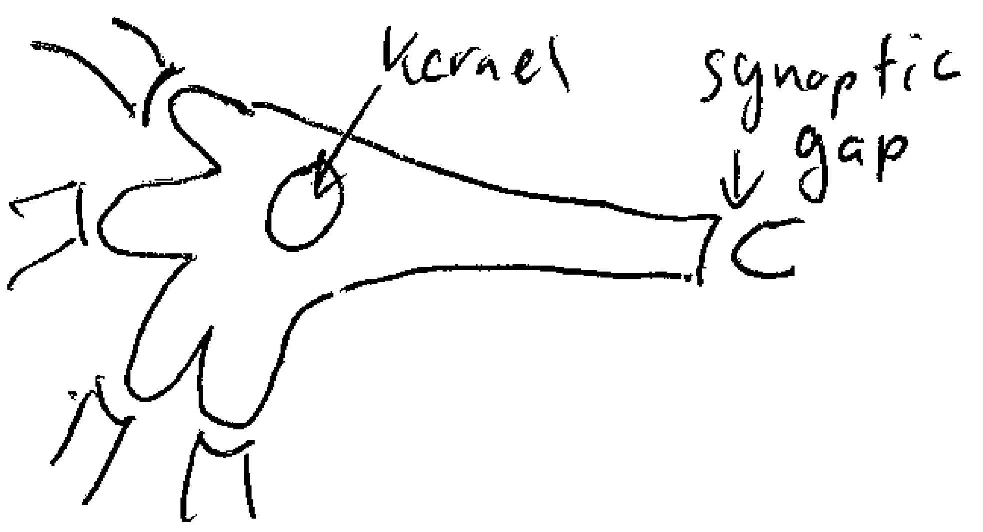
Result: ~~set po best~~ collection of elements with best fitness or elements of the best population

- Examples:
  - minimization, parameter estimation
  - function approximation
  - program writing

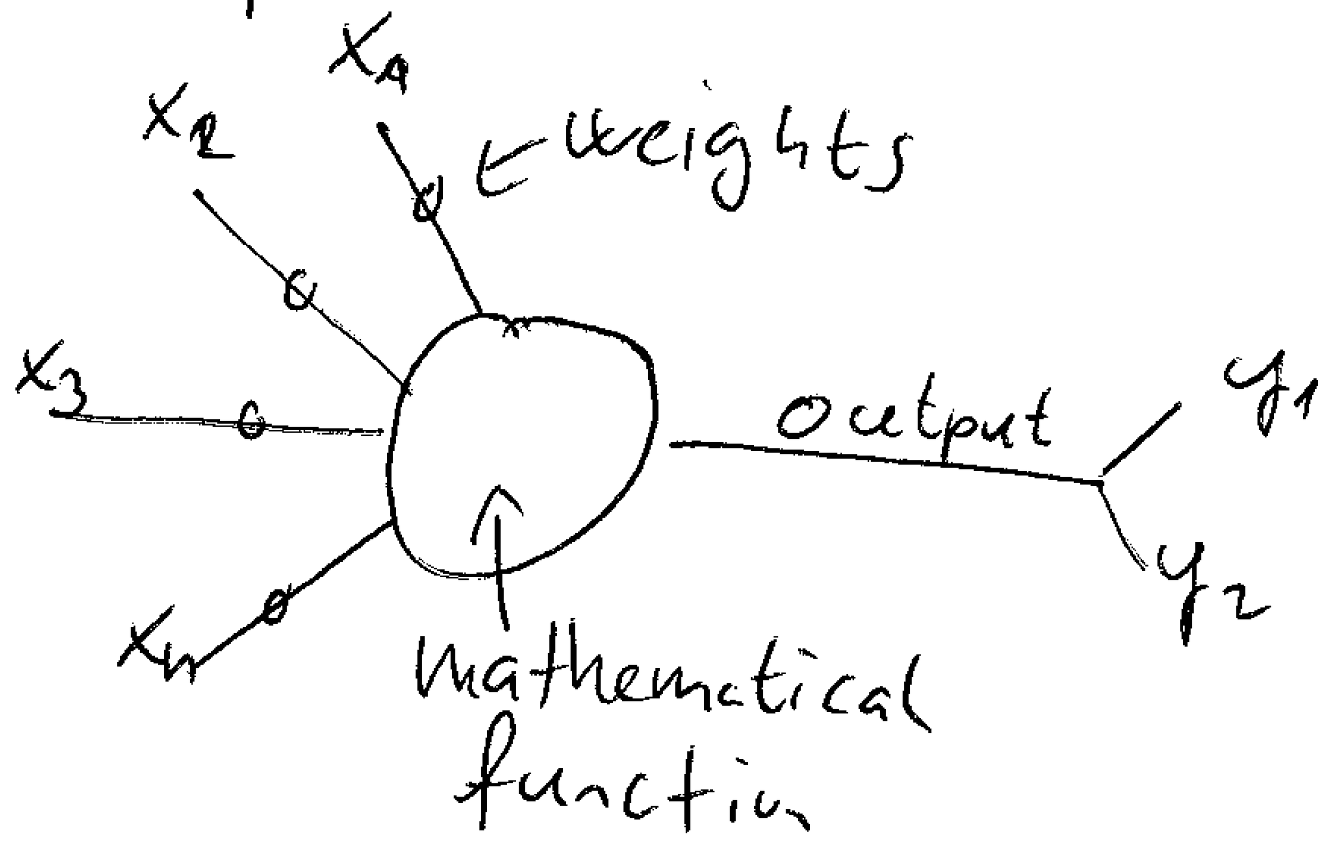
# Neural networks



biology: neural cell




$\neq$



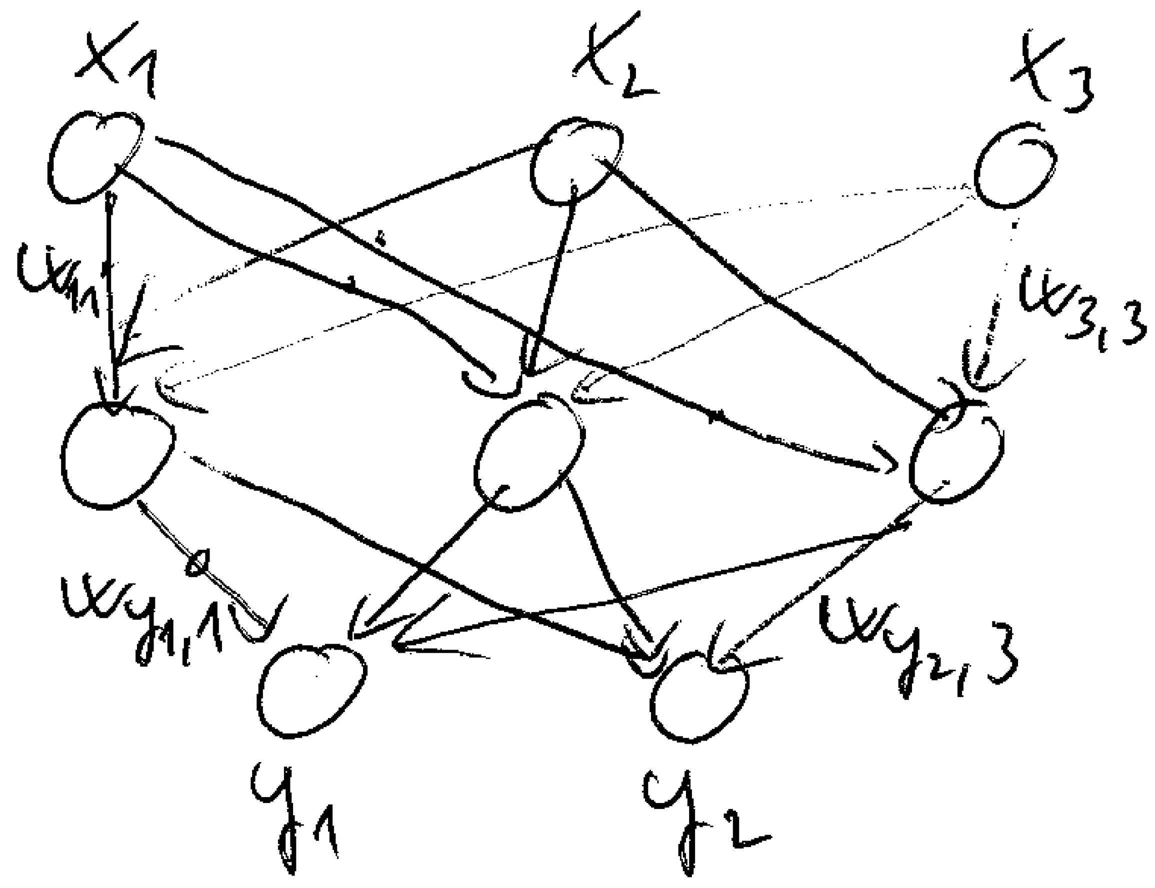

$$y_J = f_J\left(w_0 + \sum_{m=1}^{M} w_{Jm} x_m\right)$$

Simplest structure:

input layer

hidden layer

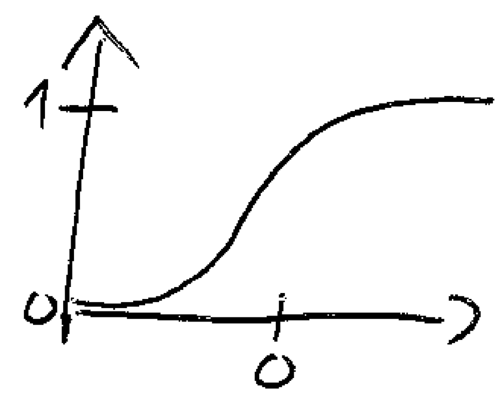
output layer




functions:

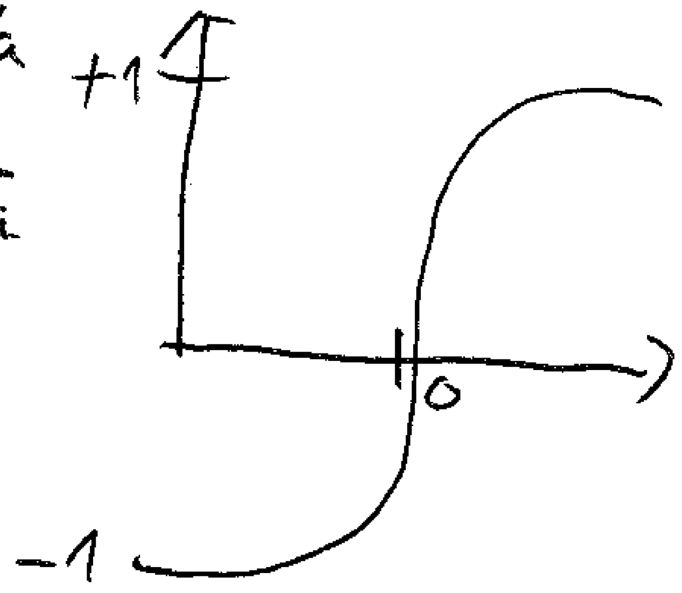
sigmoid

$$f(z) = \frac{1}{1+e^{-z/a}}$$




tangent hyperbolic

$$f(z) = \frac{e^{z/a} - e^{-z/a}}{e^{z/a} + e^{-z/a}}$$




## Supervised learning:

$N$ observation, $M$ input variables

$J$ points in the hidden layer, $R$ number of hidden layers

$K$ number of outputs

$R = 1$ all continous functions can be approximated (derivable ones)

$R = 2$ also non continouos ones!

1.) Training: $N$ cases where $\underline{x}$ and $\underline{y}$ are known $\Rightarrow$ determine all weights

2.) Use it when only $\underline{x}$-s are known, to get $\underline{y}$

Dataset size for training:

$$\frac{J(M+K)+1}{N} < 0.1$$

to avoid overfitting

Deep-learing: more hidden layers

How to find W-s?

$X$ input matrix $(N \times M)$

$y$ output matrix $(N \times K)$

weight matrix

minimise: $E = \| Y - F(W, X) \|$

- gradient method by layers — $W \to \underline{w} \in$ row vector in $W$

iterative

$$\underline{w}(t+1) = \underline{w}(t) - \beta \frac{\partial E^2(W, t)}{\partial \underline{w}\, t} = \underline{w}(t) - \beta\, d(t)$$

constant

mostly by least squares:

$$\left(\frac{\partial E^2}{\partial \underline{w}}\right) = 2E \frac{\partial E}{\partial \underline{w}} = -2E \frac{\partial F(W, X)}{\partial \underline{w}} X$$

- back propagation algorithm

for $y$ answer on $\underline{X}$ input

target; min } $E = \frac{1}{2} \cdot \left(y - f(z(\underline{x}, \underline{w}))\right)^2$

$$\frac{\partial E}{\partial w_i} = \frac{dE}{df} \cdot \frac{df}{dz} \cdot \frac{\partial z}{\partial w_i}$$

$\left( \text{if } z = \sum w_i x_i \quad \frac{\partial z}{\partial w_i} = x_i \quad \frac{\partial z}{\partial x_i} = w_i \right)$

$f(z) = \frac{1}{1+e^{-z}}$ (sigmoid)

$$\frac{df}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} \cdot \left(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right) = f \cdot (1-f)$$

$$\frac{\partial E}{df} = f - y$$

$$\frac{\partial E}{\partial w_i} = (f-y) \cdot f \cdot (1-f) \cdot x_i$$

$\Delta w_i = -\frac{\partial E}{\partial w_i}$ ← step in gradient direction, for all $i$

$\eta$ - learning coefficient ← effect of new

$\alpha$ - suspension ← old

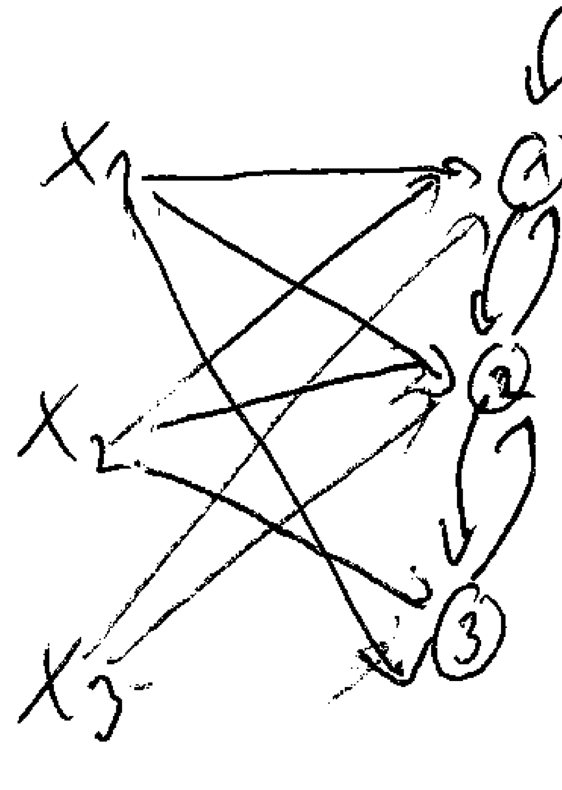
# Unsupervised learning

output is not important

mapping of input onto a lower dimension → clustering of objects
clustering of variables
data size reduction

## Kohonen network

≈ memory in brain



functions on a grid, e.g. 2D grid $\underline{r}_i, \underline{r}_j, \dots$

change $\underline{w}_i$-s

– select an $\underline{x}_i$ → search the most similar $\underline{w}_n \Rightarrow \underline{w}_n^*$

$$\underline{w}_n(t+1) = \underline{w}_n + \eta\, \phi(\underline{r}_n, r_{n^*})(\underline{x} - \underline{w}_n(t))$$

$\eta$: learning speed; $\phi$: physical distance, eg. gaussian

⇒ objects are mapped on a grid

similar objects on the same/neighboring grid

Many different versions for NN

– telecommunication connection

– ...