Chapter 7

Stochastic methods

Introduction

- In most high-dimensional and complicated models, there are multiple maxima and we must use a variety of tricks:
  - searching multiple times from different starting conditions
- a naïve approach – exhaustive search through solution space – get out of hand and is completely impractical for real-world problems.
- stochastic methods for finding parameters
  - *randomness* plays a crucial role in search and learning
  - to bias the search toward regions where we expect the solution to be and allow randomness to help find good parameters.

Introduction

- Two general classes of such methods
  - Boltzman learning
    - physics based – statistical mechanics
    - highly developed and rigorous theory and has many successes in pattern recognition.
  - Genetic algorithms
    - biology based – mathematical theory of evolution
    - more heuristic yet affords flexibility and can be attractive when adequate computational resources are available.

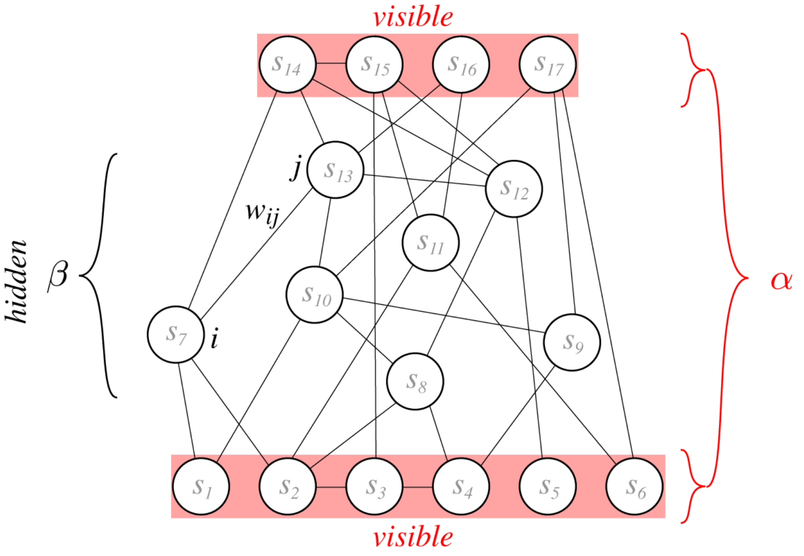
- due to high computational burden, they would be of little use without computers.

Stochastic search

- suppose we have a large number of variables, where each variable can take one of two discrete values.
  - The optimization problem is *to find the values of the s_i so as to minimize the cost or energy*

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} s_i s_j$$

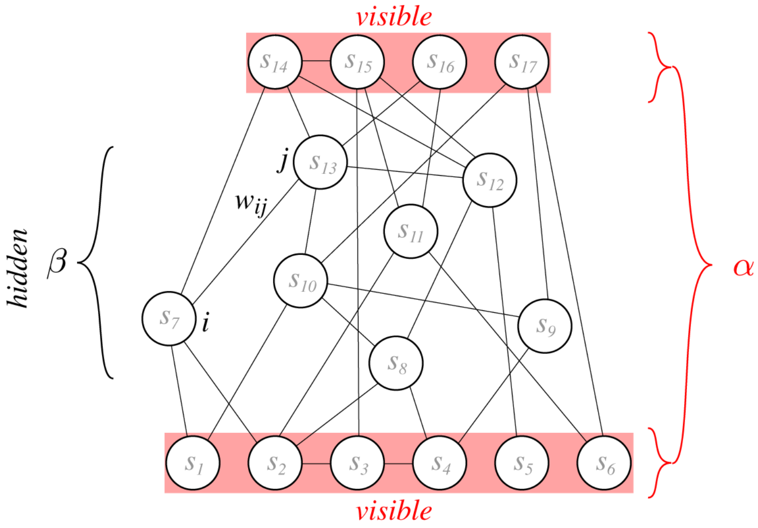
  - Except for very small problems or few connections, it is infeasible to solve directly for the N values s_i that give the minimum energy.
    - the space has 2^N possible configurations.



Stochastic search

- the network represents N physical magnets
- the w_{ij} are functions of the physical separations between the magnets.
- each pair of magnets has an associated interaction energy which depends on their state, separation and other physical properties: $E_{ij} = -\frac{1}{2} w_{ij} s_i s_j$
  - the energy of the full system is the sum of all interaction energies.
- *Greedy search*
  - consider node in turn and calculate the energy with $s_i = +1$ and the in the $s_i = -1$. Choose the one giving the lower energy.
    - usually gets caught in local minima or never converge.

$$E = -\frac{1}{2} \sum_{i,j=1}^{N} w_{ij} s_i s_j$$



Stochastic search

Simulated annealing

- annealing
  - in physics, the method for allowing a system such as many magnets or atoms in an alloy to find a low-energy configuration.
  - the system is heated, thereby conferring randomness to each component (magnet).
    - the full system explores configurations that have high energy.
  - the process proceeds by gradually lowering the temperature of the system so as to allow the system to relax into a low-energy configuration.
  - as the temperature is lowered, the system has increased probability of finding the optimum configuration.
  - at moderately high temperatures the system slightly favors regions in the configuration space that are overall lower in energy, and hence are more likely to contain the global minimum.

Stochastic search

Simulated annealing

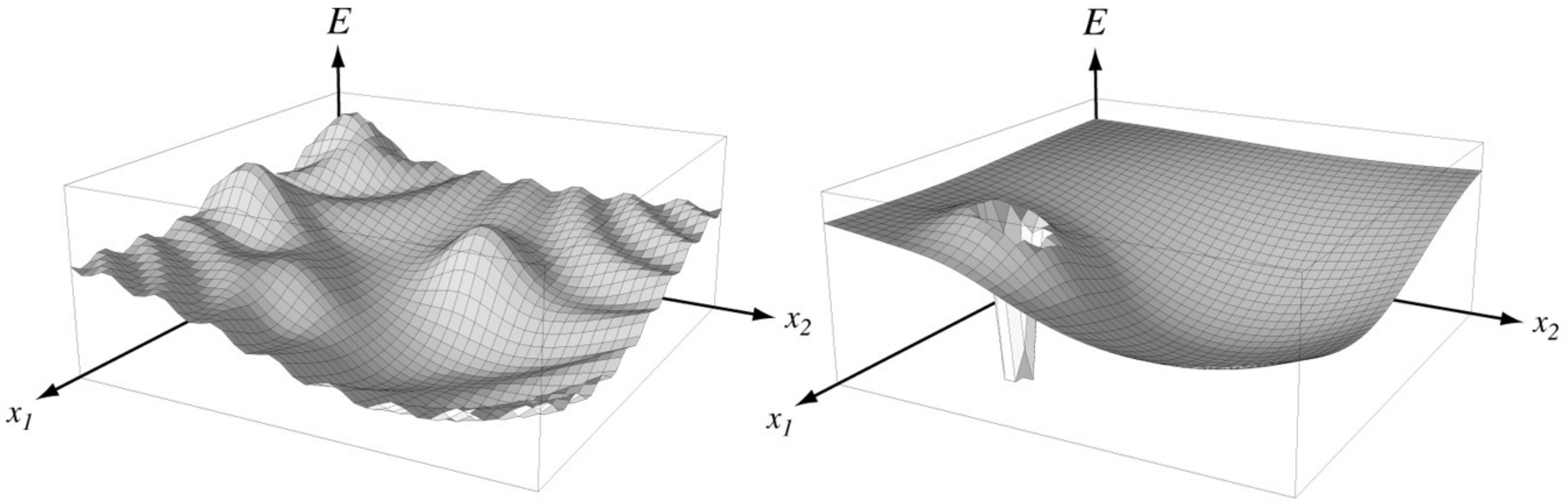


FIGURE 7.2. The energy function or energy “landscape” on the left is meant to suggest the types of optimization problems addressed by simulated annealing. The method uses randomness, governed by a control parameter or “temperature” T to avoid getting stuck in local energy minima and thus to find the global minimum, like a small ball rolling in the landscape as it is shaken. The pathological “golf course” landscape at the right is, generally speaking, not amenable to solution via simulated annealing because the region of lowest energy is so small and is surrounded by energetically unfavorable configurations. The configuration spaces of the problems we shall address are discrete and are more accurately displayed in Fig. 7.6. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Stochastic search

The Boltzmann factor

- the probability the system is in a (discrete) configuration indexed by γ having energy E_γ is given by

$$P(\gamma) = \frac{e^{-E_\gamma / T}}{Z(T)}$$

Blotzmann factor

partition function

$$Z(T) = \sum_{\gamma'} e^{-E_{\gamma'} / T}$$

- understanding the factor via a slight detouring
  - noninteracting magnets (without interconnecting weights) in a uniform external magnetic field.
    - the probability that the system has a particular total energy is related to the number of configurations that have that energy.
      - □ the highest-energy configuration: $\binom{N}{N} = 1$
      - □ the next-to-highest energy configurations: $\binom{N}{1} = N$
      - □ the next-lower-energy configurations: $\binom{N}{2} = N(N-1)/2$

Stochastic search

The Boltzmann factor (cont.)

- the exponential form of the Boltzmann factor is due to the exponential decrease in the number of accessible configurations with increasing energy.
- T in the Boltzmann factor: $e^{-E_\gamma/T}$
  - at high T, the probability is distributed roughly evenly among all configurations.

    $$e^{-E_\gamma/T} \approx e^0 = 1$$
  - at low T, it is concentrated at the lowest-energy configurations.
- in the case of magnets are interconnected by weights, the situation is a bit more complicated.
  - the energy associated with a magnet pointing up or down depends upon the states of others.

Stochastic search

simulated annealing algorithm (*algorithm 1 in p355*)

- start with randomized states and select a high initial $T(1)$.
- choose a node i randomly.
    - suppose s_i=+1. calculate the system energy, E_a.
    - recalculate E_b for a candidate new state s_i=-1.
- if $E_b < E_a$, accept the change.
- if $E_a < E_b$, accept the change with a probability equal to $e^{-(E_b - E_a)/T}$
    - the occasional acceptance of a state that is energetically less favorable is crucial to the success of SA.
        - □ it allows the system to jump out of unacceptable local energy minima.
- the algorithm continues *polling* the node randomly several times and setting their state in this way.
- the temperature is lowered and the polling repeated.
- SA terminates when the temperature is very low.
    - if the cooling has been sufficiently slow, the system then has a high probability of being in a low-energy state – hopefully the global minimum.
- we need only consider nodes connected to the one being polled.

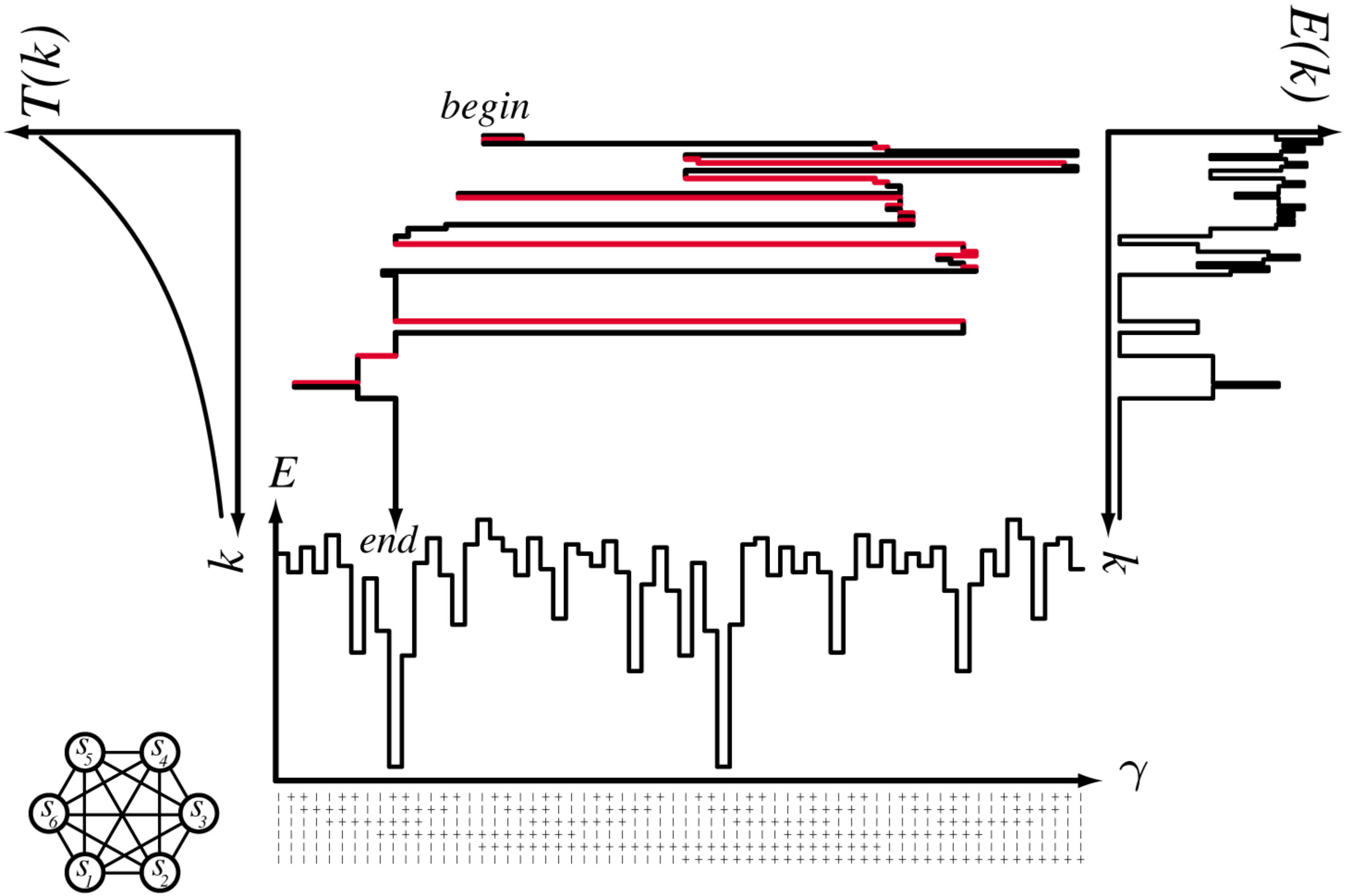
Stochastic search

simulated annealing algorithm (*cont.*)

- several aspects of the algorithm that must be considered
  - starting temperature
    - $T(k)$: the cooling schedule
    - we demand $T(1)$ to be sufficiently high that all configurations have roughly equal probability.
  - the rate of temperature decreasing
    - the decrease must be both *gradual* and *slow* enough that the system can move to any part of the state space before being trapped in an unacceptable local minimum.
    - $T(k+1) = cT(k), 0.8 < c < 0.99.$
  - the final temperature
    - it must be low enough that there is negligible probability that if the system is in a global minimum will move out.

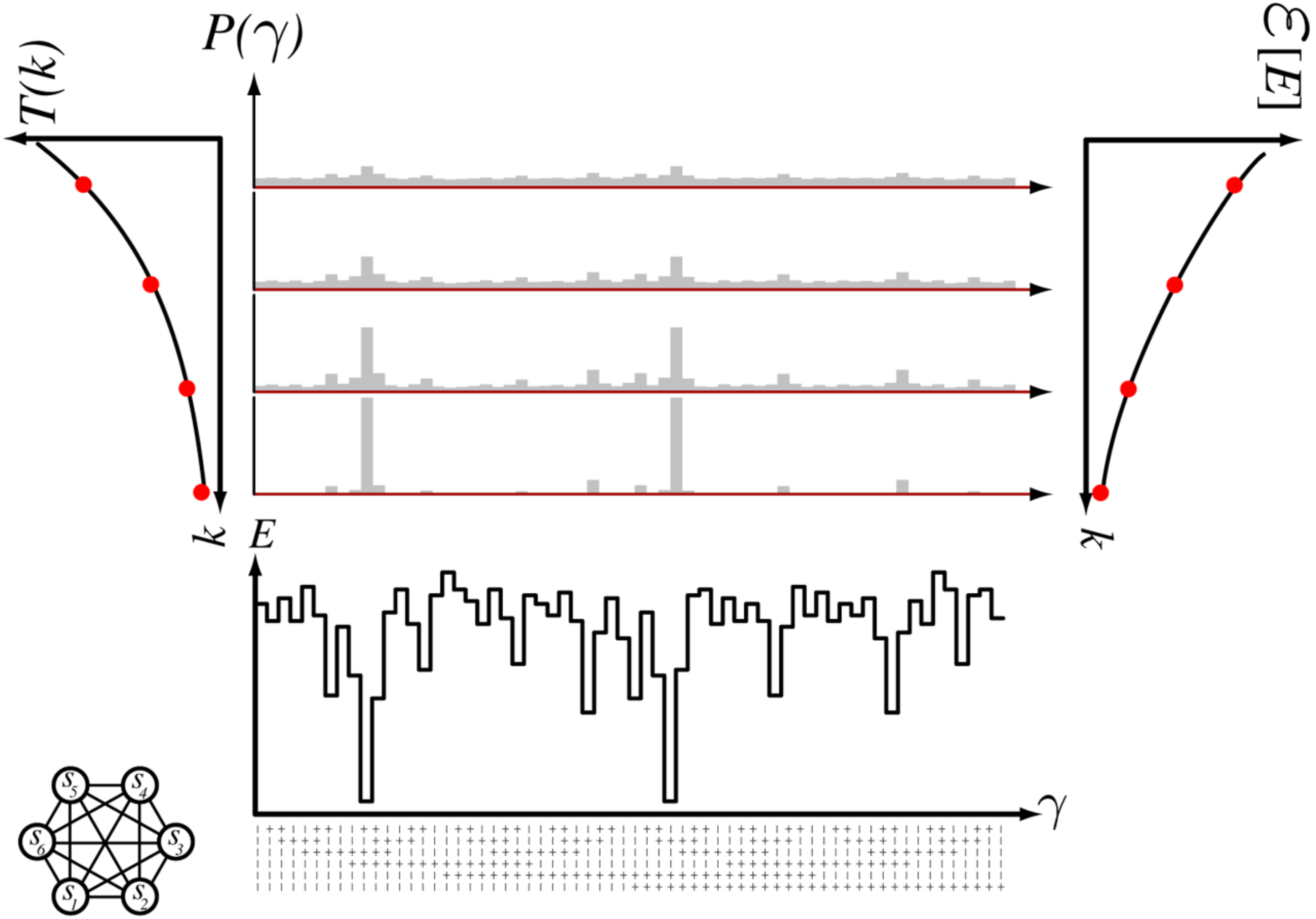
Stochastic search

simulated annealing algorithm



Stochastic search

simulated annealing algorithm



Stochastic search

Deterministic simulated annealing

- stochastic simulated annealing is slow because of the discrete nature of the search through the space of all configurations.
- a faster, alternative method is to allow each node to take on analog values during search;
  - physical analogy
    - consider a single node i connected to several others; each exerts a force tending to point node i up or down.
    - if there is a large "positive" force, then $s_i \approx +1$;
    - if a large negative force, $s_i \approx -1$.
    - in general, s_i will be between these limits.
  - at the end of the search the values are forced to be $s_i = \pm 1$, as required to by the optimization problem.

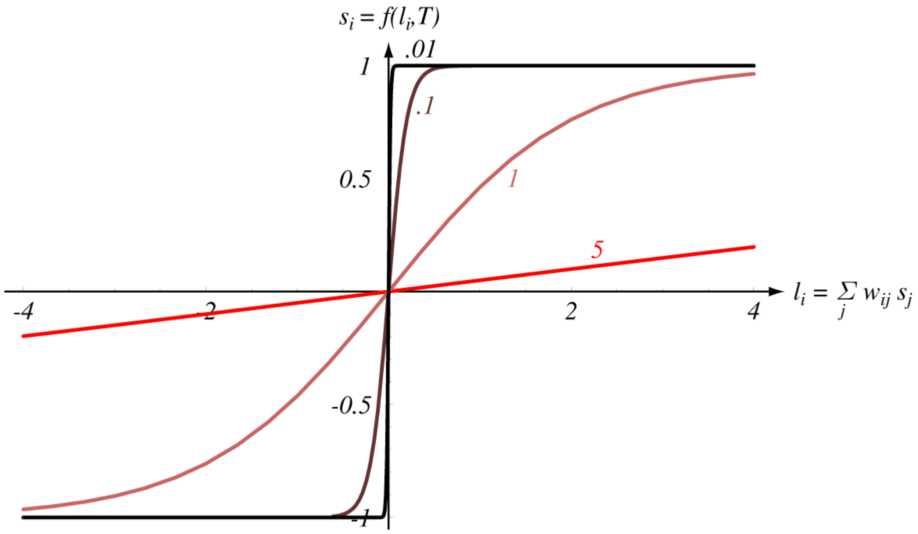
Stochastic search

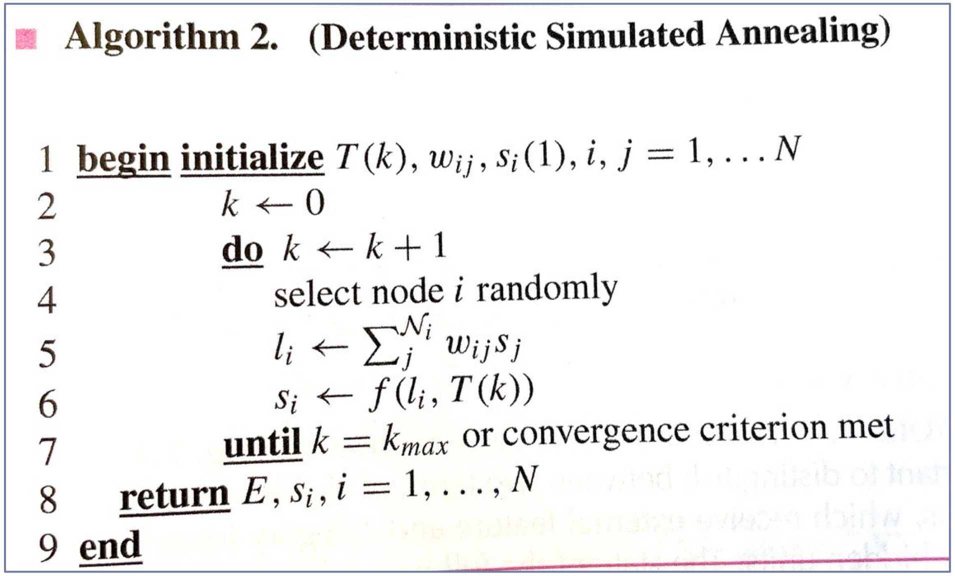
Deterministic simulated annealing (cont.)

- s_i also depend on T.

$$l_i = \sum_j w_{ij} s_j$$

$$s_i = f(l_i, T) = \tanh[l_i / T]$$





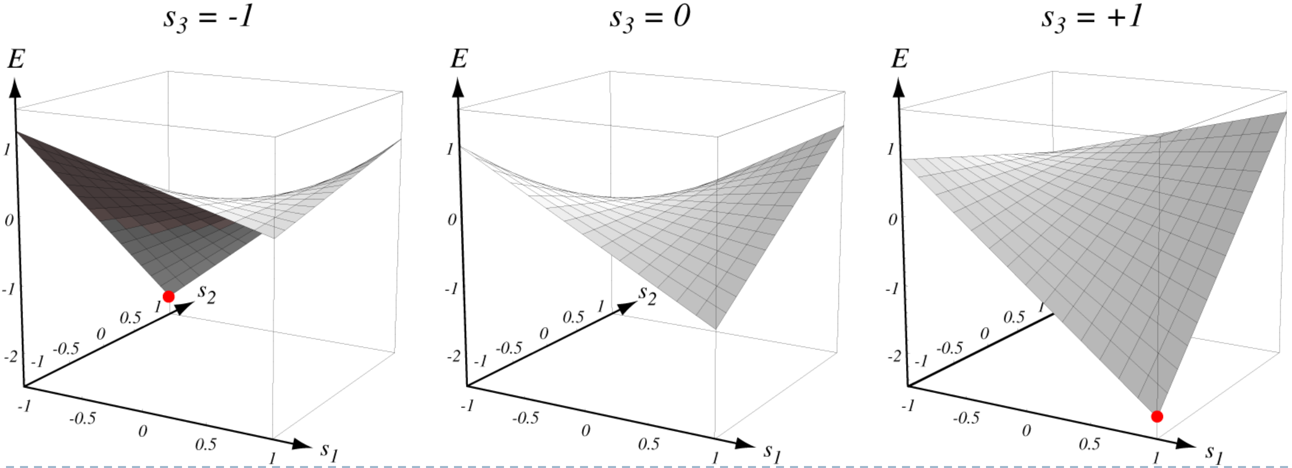
Algorithm 2. (**Deterministic Simulated Annealing**)

```
begin initialize $T(k)$, $w_{ij}$, $s_i(1)$, $i, j = 1, \ldots N$
        $k \leftarrow 0$
        do $k \leftarrow k + 1$
            select node $i$ randomly
            $l_i \leftarrow \sum_j^{N_i} w_{ij} s_j$
            $s_i \leftarrow f(l_i, T(k))$
        until $k = k_{max}$ or convergence criterion met
    return $E, s_i, i = 1, \ldots, N$
end
```

Stochastic search

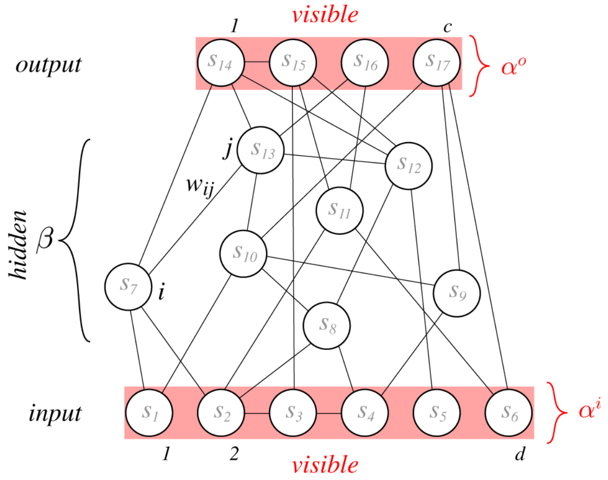
Deterministic simulated annealing (cont.)

- the search method is sometimes called *mean-field annealing* because each node responds to the average or mean of the forces due to the nodes connected to it.
  - deterministic because in principle we could deterministically solve the simultaneous equations governing the s_i as the temperature is lowered.



Boltzmann learning

- the network
  - the input units accept binary feature information; the output units represent the output categories (1-of-c)
  - during classification, the input units are held fixed (or clamped) to the feature values of the input pattern; the remaining units are annealed to find the lowest-energy.
- Accurate recognition requires proper weights; a method for learning weights from training patterns.



Boltzmann learning

Stochastic Boltzmann learning of visible states

- learning categories from training patterns
- an alternative learning problem
  - we have a set of desired probabilities for all the visible units, $Q(\alpha)$, and seek weights so that the actual probability $P(\alpha)$, achieve in random simulations, matches these probabilities over a given set of patterns.

$$P(\alpha) = \sum_{\beta} P(\alpha, \beta)$$

$$= \frac{\sum_{\beta} e^{-E_{\alpha\beta}/T}}{Z}$$

the system energy in the configuration defined by the visual and hidden parts.

  - a natural measure of the difference between the actual and the desired probability distributions is the relative entropy.

$$D_{KL}(Q(a), P(\alpha)) = \sum_{\alpha} Q(\alpha) \log \frac{Q(\alpha)}{P(\alpha)} \qquad \text{(Kullback-Leibler distance)}$$

- □ Nonnegative
- □ Zero iff $P(\alpha) = Q(\alpha)$
- □ Depends on only the visible units

Boltzmann learning

Stochastic Boltzmann learning of visible states

- learning is based on gradient descent in the relative entropy.
  - a set of training patterns defines $Q(\alpha)$
  - we seek weights so that at some temperature *T* the actual distribution $P(\alpha)$ matches $Q(\alpha)$ as closely as possible.

$$\Delta w_{ij} = -\eta \frac{\partial D_{KL}}{\partial w_{ij}} = \eta \sum_{\alpha} \frac{Q(a)}{P(\alpha)} \frac{\partial P(\alpha)}{\partial w_{ij}}$$

$$\frac{\partial P(\alpha)}{\partial w_{ij}} = \frac{\sum_{\beta} e^{-E_{\alpha\beta}/T} s_i(\alpha\beta) s_j(\alpha\beta)}{TZ} - \frac{\left(\sum_{\beta} e^{E_{\alpha\beta}/T}\right) \sum_{\lambda\mu} e^{-E_{\lambda\mu}/T} s_i(\lambda\mu) s_j(\lambda\mu)}{TZ^2}$$

$$= \frac{1}{T}\left[\sum_{\beta} s_i(\alpha\beta) s_j(\alpha\beta) P(\alpha,\beta) - P(\alpha)\mathrm{E}\left[s_i s_j\right]\right]$$

  - $s_i(\alpha\beta)$ is that state of node i in the full configuration specified by α and β.

□ If node i is a visible one: only α is relevant.

□ If node i is a hidden one: only β is relevant.

Boltzmann learning

Stochastic Boltzmann learning of visible states

$$\Delta w_{ij} = \frac{\eta}{T}\left[\sum_{\alpha}\frac{Q(\alpha)}{P(\alpha)}\sum_{\beta} s_i(\alpha\beta)s_j(\alpha\beta)P(\alpha,\beta) - \sum_{\alpha} Q(\alpha)\mathrm{E}\left[s_i s_j\right]\right]$$

$$= \frac{\eta}{T}\left[\sum_{\alpha\beta} Q(\alpha)P(\beta\,|\,\alpha)s_i(\alpha\beta)s_j(\alpha\beta) - \mathrm{E}\left[s_i s_j\right]\right]$$

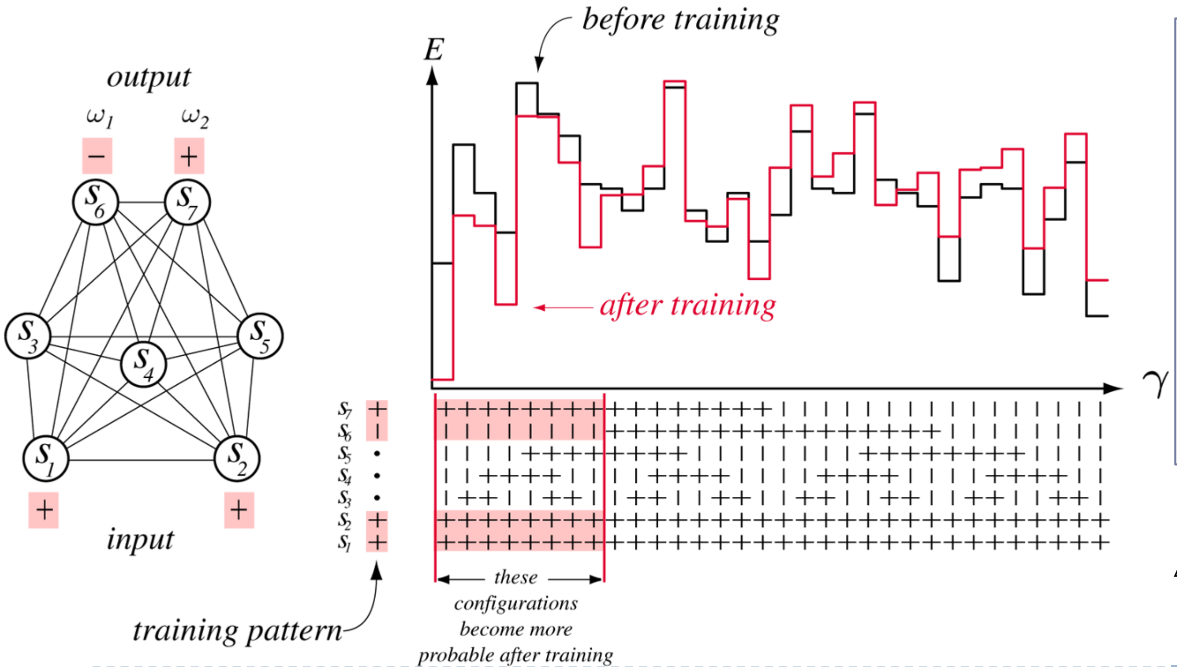
$$= \frac{\eta}{T}\left[\underbrace{\mathrm{E}_{\mathrm{Q}}\left[s_i s_j\right]_{\alpha\ clamped}}_{learning} - \underbrace{\mathrm{E}\left[s_i s_j\right]_{free}}_{unlearning}\right]$$

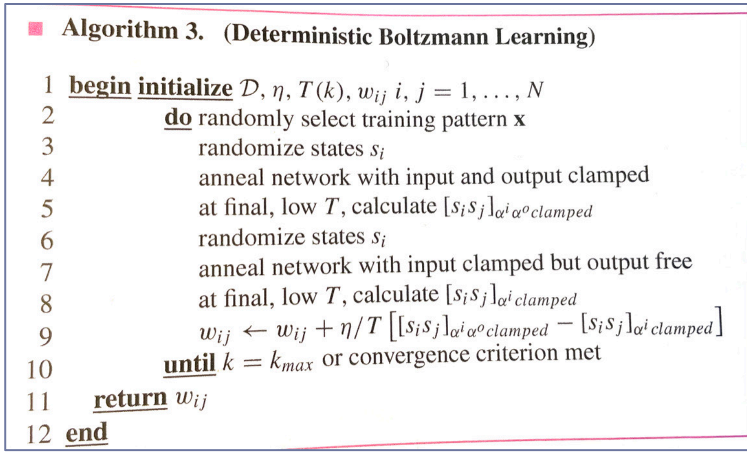
- $\mathrm{E}_{\mathrm{Q}}\left[s_i s_j\right]_{\alpha\ clamped} = \sum_{\alpha\beta} Q(\alpha)P(\beta\,|\,\alpha)s_i(\alpha\beta)s_j(\alpha\beta)$ *the correlation of the variable s_i and* s_j when visible units are held fixed – in visible configuration α, averaged according to the probabilities of the training patterns, $Q(\alpha)$.
- if learning components becomes equal to unlearning component, we have achieved the desired weights.

Boltzmann learning

Stochastic learning of input-output associations

- the problem of learning mappings from input to output
- we want the network to learn associations between the (visible) state on the input units and states on the output units.
  - we want $P(\alpha^o|\alpha^i)$ to match $Q(\alpha^o|\alpha^i)$ as closely as possible.





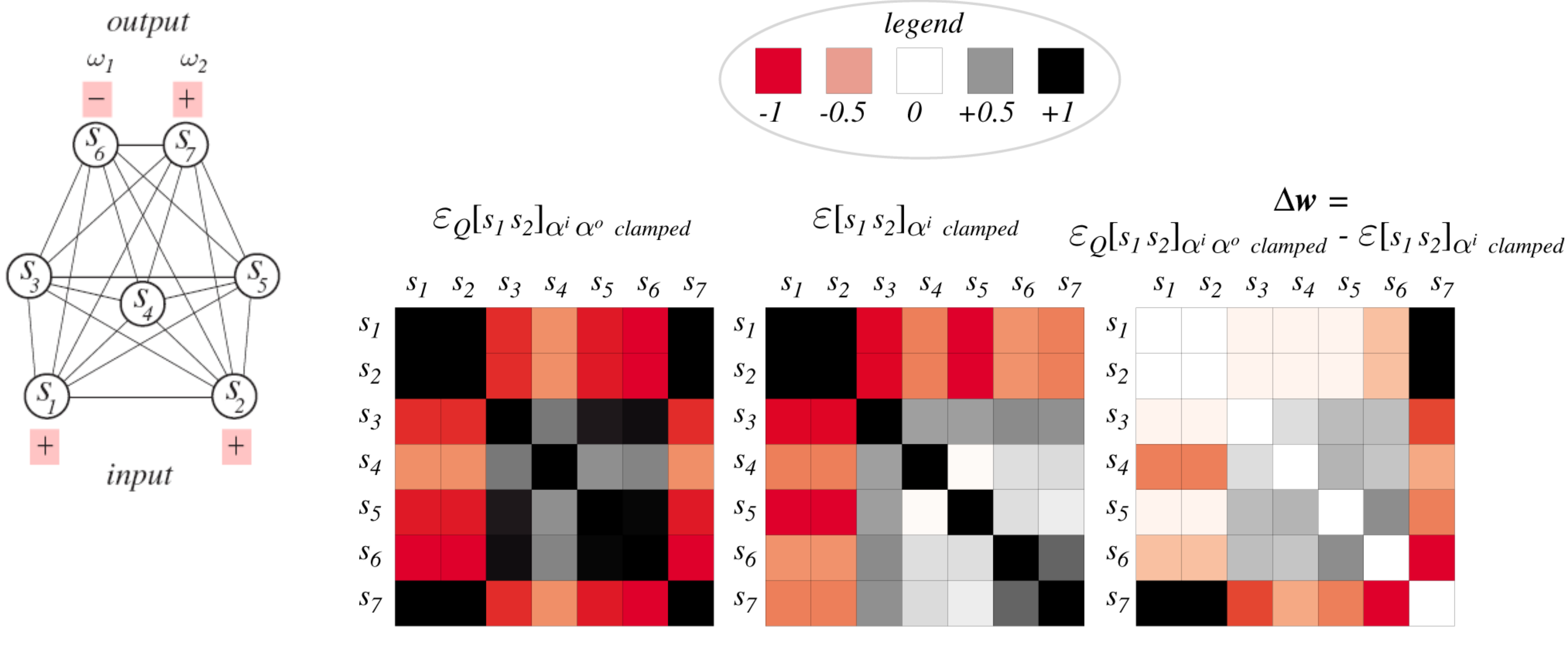
Algorithm 3. (Deterministic Boltzmann Learning)

```
begin initialize $\mathcal{D}, \eta, T(k), w_{ij}\ i, j = 1, \ldots, N$
    do randomly select training pattern $\mathbf{x}$
        randomize states $s_i$
        anneal network with input and output clamped
        at final, low $T$, calculate $[s_i s_j]_{\alpha^i \alpha^o clamped}$
        randomize states $s_i$
        anneal network with input clamped but output free
        at final, low $T$, calculate $[s_i s_j]_{\alpha^i clamped}$
        $w_{ij} \leftarrow w_{ij} + \eta/T\,\big[[s_i s_j]_{\alpha^i \alpha^o clamped} - [s_i s_j]_{\alpha^i clamped}\big]$
    until $k = k_{max}$ or convergence criterion met
    return $w_{ij}$
end
```

$$\Delta w_{ij} = \frac{\eta}{T}\left[\underbrace{\mathrm{E}_{\mathrm{Q}}\left[s_i s_j\right]_{\alpha^i\alpha^o\ clamped}}_{learning} - \underbrace{\mathrm{E}\left[s_i s_j\right]_{\alpha^i\ clamped}}_{unlearning}\right]$$

Boltzmann learning

Stochastic learning of input-output associations


output
ω_1
ω_2
−
+
S_6
S_7
S_3
S_4
S_5
S_1
S_2
+
+
input
legend
-1
-0.5
0
+0.5
+1
$\mathcal{E}_Q[s_1 s_2]_{\alpha^i \alpha^o}$ clamped
$\mathcal{E}[s_1 s_2]_{\alpha^i}$ clamped
$\Delta \mathbf{w} =$
$\mathcal{E}_Q[s_1 s_2]_{\alpha^i \alpha^o}$ clamped - $\mathcal{E}[s_1 s_2]_{\alpha^i}$ clamped
s_1 s_2 s_3 s_4 s_5 s_6 s_7

learning *unlearning* *weight update*

$$\Delta w_{ij} = \frac{\eta}{T}\left[\underbrace{\mathrm{E}_Q\left[s_i s_j\right]_{\alpha^i \alpha^o\ clamped}}_{learning} - \underbrace{\mathrm{E}\left[s_i s_j\right]_{\alpha^i\ clamped}}_{unlearning}\right]$$

Boltzmann learning

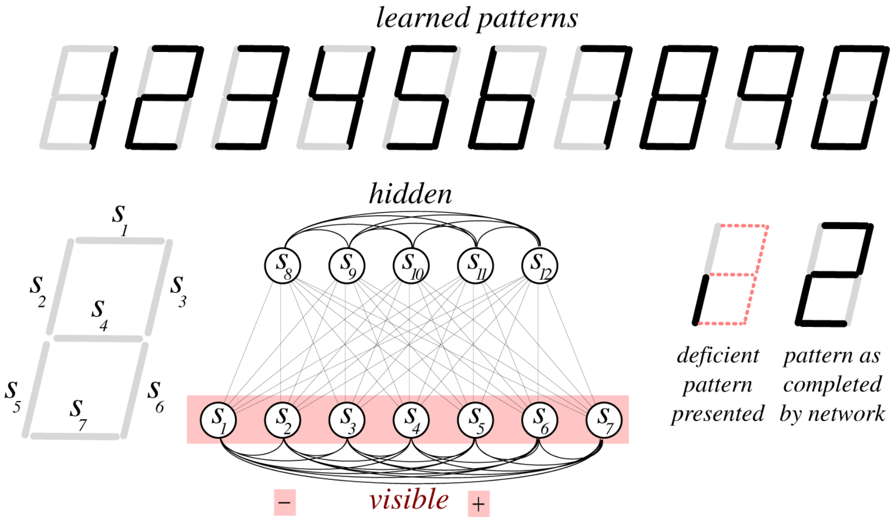
Missing features and category constraints

- if a deficient binary pattern is used for training, input units corresponding to missing features are allowed to vary.
  - they are temporarily treated as (unclamped) hidden units rather than clamped input units.
  - during annealing such units assume values most consistent with the rest of the input pattern and the current state of the network.
- when a deficient pattern is to be classified, any units corresponding to missing input features are not clamped and are allowed to assume any value.

Boltzmann learning

Missing features and category constraints

- pattern completion
  - To estimate the full pattern given just a part of that pattern
    - It is related to the problem of classification with missing features
  - Boltzmann networks without hidden or category units are closely related to Hopfield networks (associative networks).



  - pattern completion – algorithm 3 (page 367)

Boltzmann learning

Initialization and setting parameters

- several interrelated parameters that must be set in a Boltzmann network.
  - the number of visible units is determined by the dimensions of the binary feature vectors and number of categories.
  - the number of hidden units?
    - An upper bound on the minimum number of hidden units is n – one for each pattern.
    - a lower bound on the number of hidden units is $\lceil log_2 n \rceil$, where n is the number of distinct training patterns.
  - weight initialization
    - weights should be initialized randomly throughout the range

$$-\sqrt{3/N} < w_{ij} < +\sqrt{3/N}$$

  - the number of iterations

$$T(k) = T(1)e^{-k/k_0}$$

Boltzmann learning

Initialization and setting parameters

- several interrelated parameters that must be set in a Boltzmann network.
  - the learning rate

$$\eta < \frac{T^2}{N}$$

  - one heuristic that provides modest computational speedup
    - to propose changing the states of several nodes simultaneously early in an anneal.
  - two stopping criteria
    - the first determines when to stop a single anneal.
      - □ the final temperature should be so low that no energetically unfavorable transitions are accepted.
    - the second stopping criterion controls the number of times each training patterns is presented to the network.

Evolutionary methods

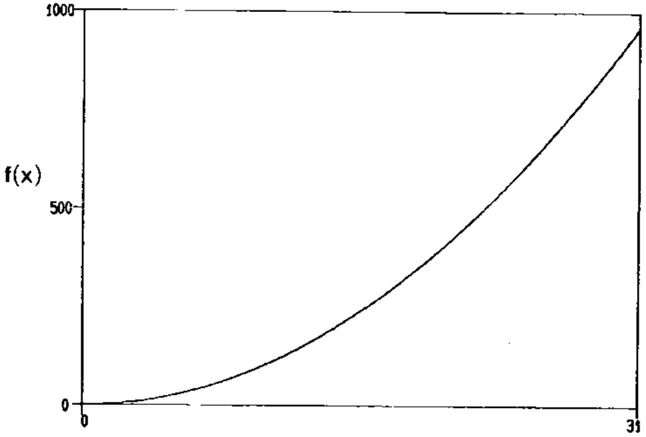
- Inspired by the biological evolution.
  - employing stochastic search for an optimal classifier
  - occasional very large changes in the classifier are introduced.
- steps
  - creating several classifiers – a population – each *varying* somewhat from the other
  - scoring each classifier on a representative version of the classification task
    - fitness
  - ranking the classifiers according to their score and retain the best classifiers
    - survival of the fitness
  - the entire process is repeated for generations

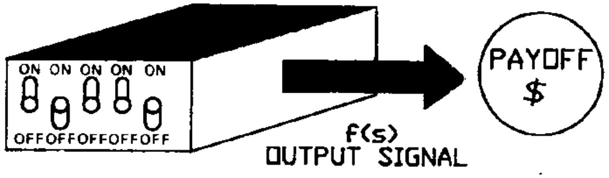
Evolutionary methods

- Genetic Algorithms
  - chromosome
    - the fundamental representation of each classifier
    - the mapping from the chromosome to the features and other aspects of the classifier depends on the problem domain.
  - Genetic operators
    - **Replications**: a chromosome is merely reproduced, unchanged.
    - **Crossover**: mixing/mating of two chromosomes. A split point is chosen randomly along the length of either chromosome. P_{co} is the probability a given pair of chromosomes will undergo crossover.
    - **Mutation**: each bit in a single chromosome is given a small chance, P_{mut}, of being changed from a 1 to a 0 or vice versa.
  - Algorithm 4 (Basic Genetic Algorithm)

A Simple Genetic Algorithm

- GAs require the natural parameter set of the optimization problem to be coded as a finite-length string over some finite alphabet.
  - $f(x)=x^2$ on the integer interval [0,31]
  - A bank of five input switches – a way to code the parameter x.
- The objective of the problem is to set the switches to obtain the maximum possible f value.

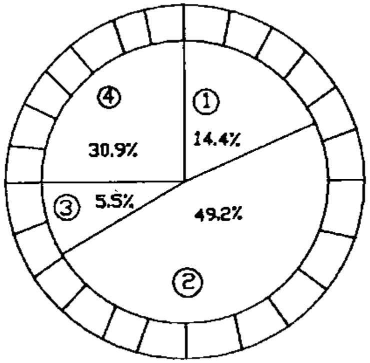




A Simple Genetic Algorithm

no	string	fitness	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

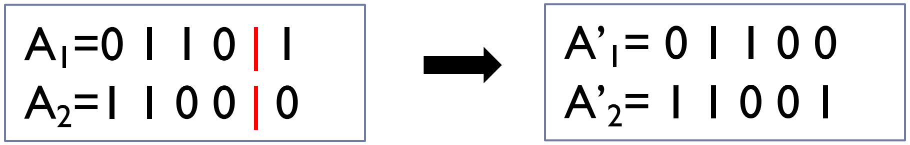
- The initial population was chosen at random through 20 successive flips of an unbiased coin.
  - We now must define a set of simple operations that take this initial population and generate successive populations that improve over time.
  - Reproduction
  - Crossover
  - Mutation



weighted roulette
wheel for reproduction

A Simple Genetic Algorithm

- Crossover
  - Member of the newly reproduced strings in the mating pool are mated at random.
  - Each pair of strings undergoes crossing over; an integer position k along the string is selected uniformly at random between 1 and the string length less one.
  - Two new strings are created by swapping.



A_1=0 1 1 0 | 1
A_2=1 1 0 0 | 0

A'_1= 0 1 1 0 0
A'_2= 1 1 0 0 1

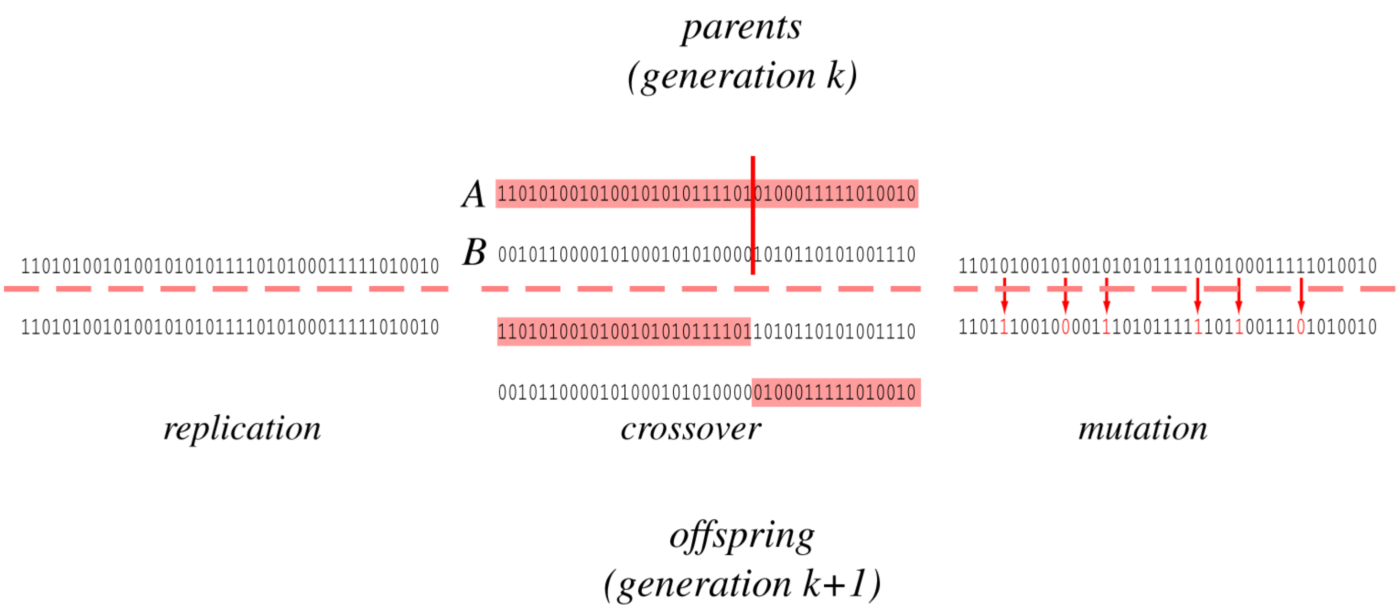
- Mutation
  - In the simple GA, mutation is the occasional (with small probability) random alteration of the value of a string position.

A Simple Genetic Algorithm

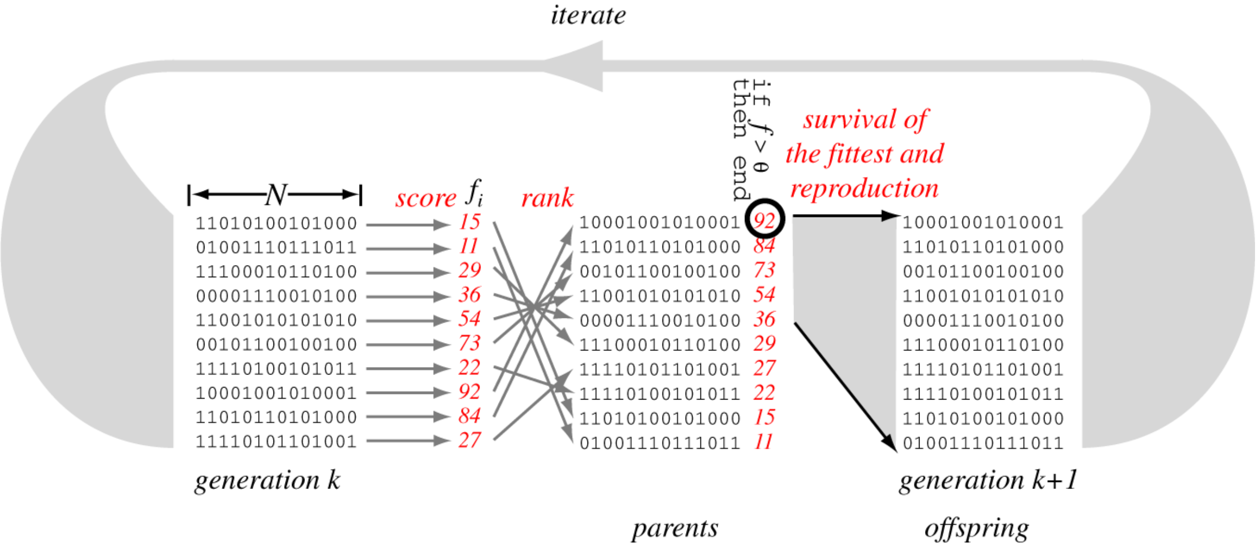
String no.	Initial population	x value	$f(x)$ x^2	pselect$_i$ $f_i / \sum$	expected count	actual count (from the wheel)
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4.0
Average			293	0.25	1.00	1.0
Max			576	0.49	1.97	2.0

String no.	Mating pool after reproduction	Mate	Crossover Site	New population	x value	$f(x)$ x^2
1	0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
3	1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	3	2	1 0 0 0 0	16	256
Sum						1754
Average						439
Max						729

Evolutionary methods



Evolutionary methods


iterate
if f > θ then end
survival of the fittest and reproduction
N
score f_i
rank
11010100101000 15
01001110111011 11
11100010110100 29
00001110010100 36
11001010101010 54
00101100100100 73
11110100101011 22
10001001010001 92
11010110101000 84
11110101101001 27
generation k
10001001010001 92
11010110101000 84
00101100100100 73
11001010101010 54
00001110010100 36
11100010110100 29
11110101101001 27
11110100101011 22
11010100101000 15
01001110111011 11
parents
10001001010001
11010110101000
00101100100100
11001010101010
00001110010100
11100010110100
11110101101001
11110100101011
11010100101000
01001110111011
generation k+1
offspring