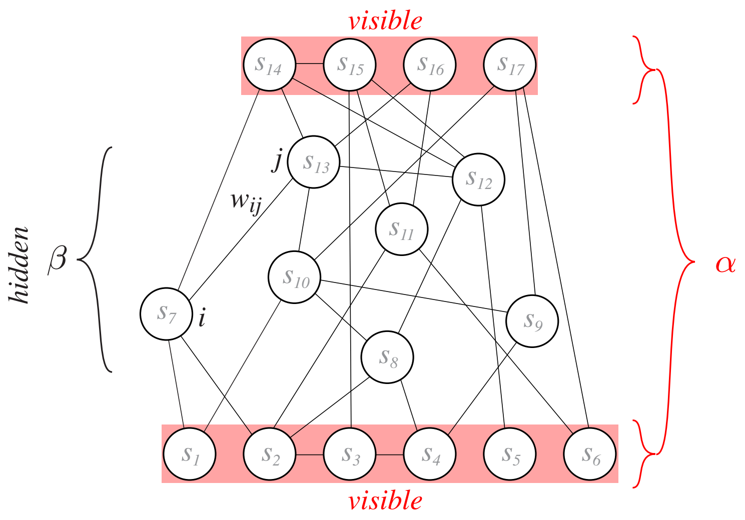


FIGURE 7.1. The class of optimization problems of Eq. 1 can be viewed in terms of a network of nodes or units, each of which can be in the $s_i = +1$ or $s_i = -1$ state. Every pair of nodes i and j is connected by bi-directional weights w_{ij}; if a weight between two nodes is zero, then no connection is drawn. (Because the networks we shall discuss can have an arbitrary interconnection, there is no notion of layers as in multilayer neural networks.) The optimization problem is to find a configuration (i.e., assignment of all s_i) that minimizes the energy described by Eq. 1. While our convention was to show *functions* inside each node's circle, our convention in so-called Boltzmann networks is to indicate the *state* of each node. The configuration of the full network is indexed by an integer γ, and because here there are 17 binary nodes, γ is bounded $0 \leq \gamma < 2^{17}$. When such a network is used for pattern recognition, the input and output nodes are said to be visible, and the remaining nodes are said to be hidden. The states of the visible nodes and hidden nodes are indexed by α and β, respectively, and in this case are bounded $0 \leq \alpha \leq 2^{10}$ and $0 \leq \beta < 2^7$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

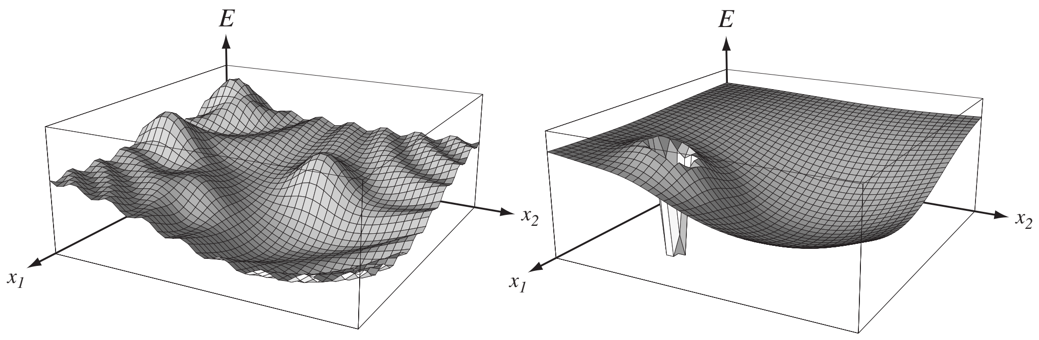


FIGURE 7.2. The energy function or energy "landscape" on the left is meant to suggest the types of optimization problems addressed by simulated annealing. The method uses randomness, governed by a control parameter or "temperature" T to avoid getting stuck in local energy minima and thus to find the 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.

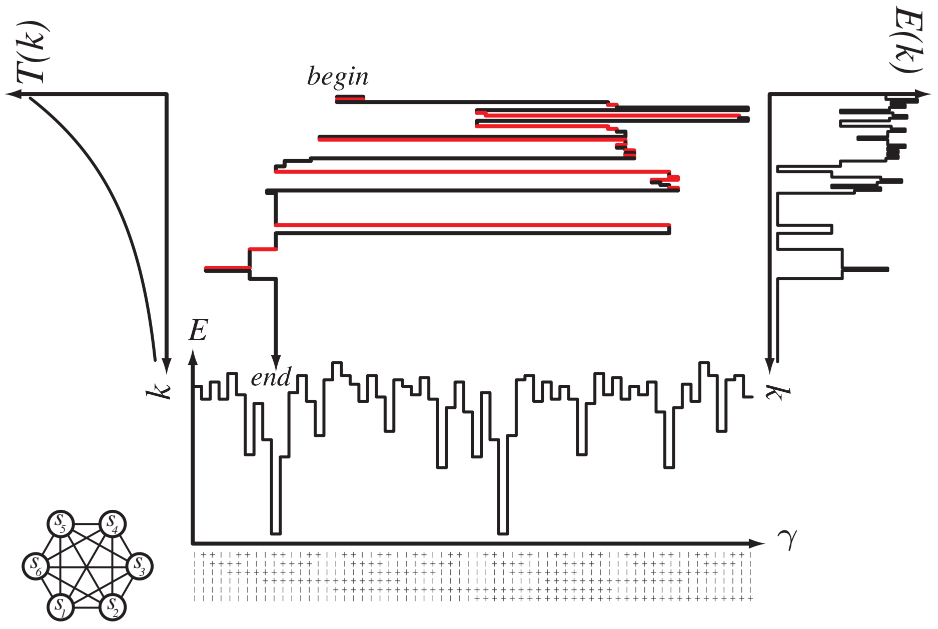


FIGURE 7.3. Stochastic simulated annealing (Algorithm 1) uses randomness, governed by a control parameter or "temperature" $T(k)$ to search through a discrete space for a minimum of an energy function. In this example there are $N = 6$ variables; the $2^6 = 64$ configurations are shown along the bottom as a column of + and − symbols. The plot of the associated energy of each configuration given by Eq. 1 for randomly chosen weights. Every transition corresponds to the change of just a single s_i. (The configurations have been arranged so that adjacent ones differ by the state of just a single node; nevertheless, most transitions corresponding to a single node appear far apart in this ordering.) Because the system energy is invariant with respect to a global interchange $s_i \leftrightarrow -s_i$, there are two "global" minima. The graph at the upper left shows the annealing schedule—the decreasing temperature versus iteration number k. The middle portion shows the configuration versus iteration number generated by Algorithm 1. The trajectory through the configuration space is colored red for transitions that increase the energy and black for those that decrease the energy. Such energetically unfavorable (red) transitions become rarer later in the anneal. The graph at the right shows the full energy $E(k)$, which decreases to the global minimum. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

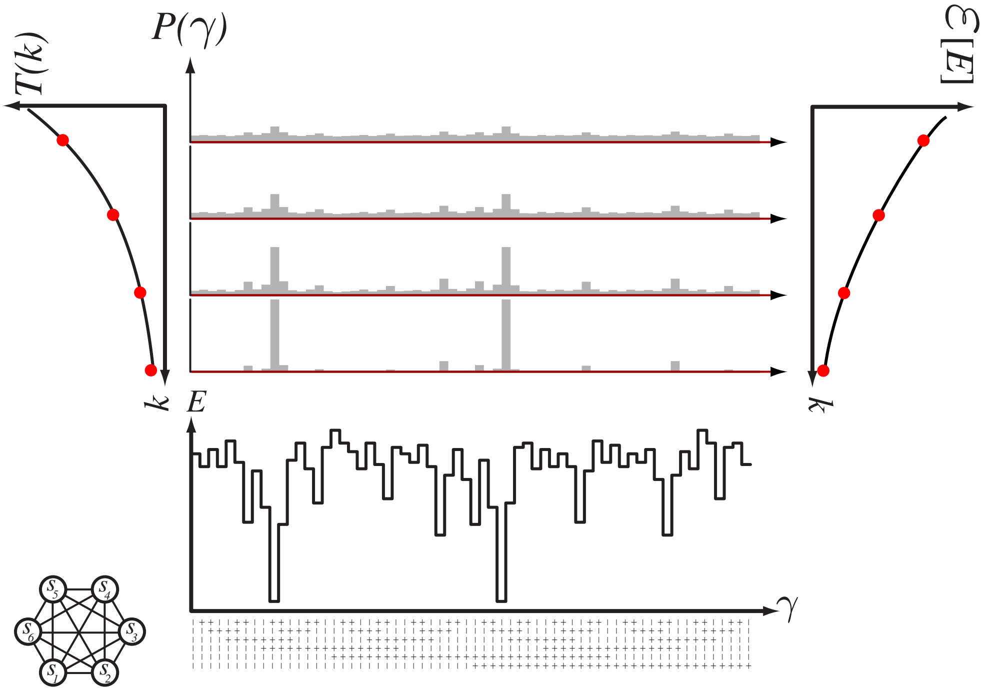


FIGURE 7.4. An estimate of the probability $P(\gamma)$ of being in a configuration denoted by γ is shown for four temperatures during a slow anneal. (These estimates, based on a large number of runs, are nearly the theoretical values e^{-E_γ}/T.) Early, at high T, each configuration is roughly equal in probability while late, at low T, the probability is strongly concentrated at the global minima. The expected value of the energy, $\mathcal{E}[E]$ (i.e., averaged at temperature T), decreases gradually during the anneal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

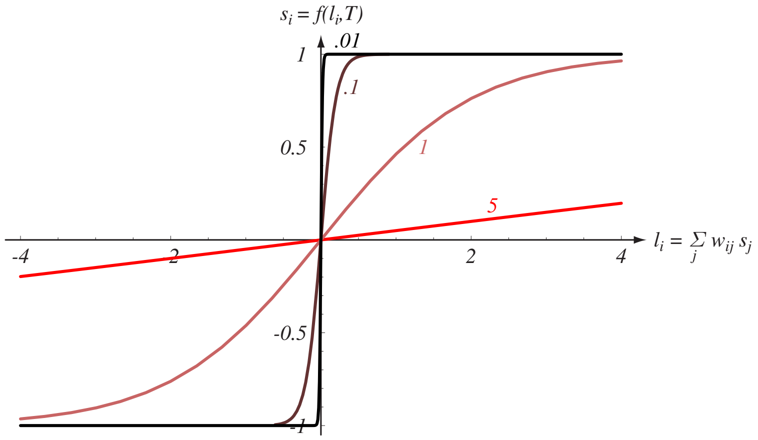


FIGURE 7.5. In deterministic annealing, each node can take on a continuous value $-1 \leq s_i \leq +1$, which equals the expected value of a binary node in the system at temperature T. In other words, the analog value s_i replaces the expectation of the discrete variable, $\mathcal{E}[s_i]$. We let l_i denote a force exerted by the nodes connected to s_i. The larger this force, the closer the analog s_i is to $+1$; the more negative this force, the closer is s_i to -1. The temperature T (marked in red) also affects s_i. If T is large, there is a great deal of randomness and even a large force will not ensure $s_i \approx +1$. At low temperature, there is little or no randomness and even a small positive force ensures that $s_i = +1$. Thus at the end of an anneal (low T), each node has value $s_i = +1$ or $s_i = -1$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

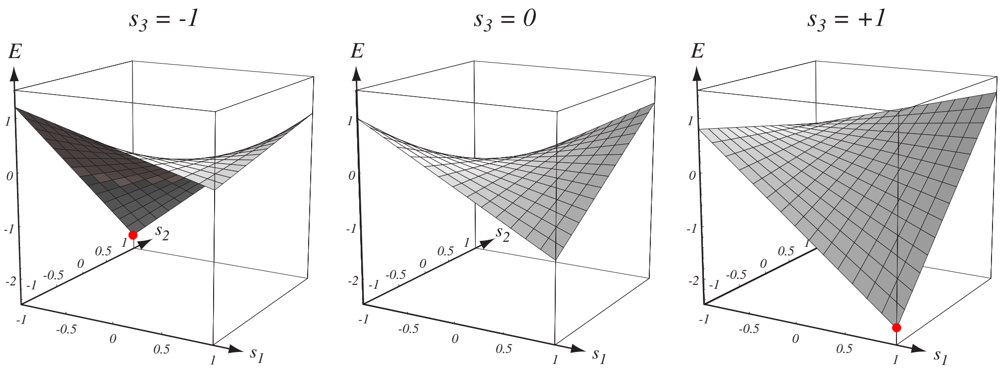


FIGURE 7.6. If the state variables s_i can assume analog values (as in mean-field annealing), the energy in Eq. 1 is a general quadratic form having minima at the extreme values $s_i = \pm 1$. In the case illustrated here $N = 3$ nodes are fully interconnected with arbitrary weights w_{ij}. While the total energy function is three-dimensional, we show two-dimensional surfaces for each of three values of s_3. The energy is linear in each variable so long as the other variables are held fixed. Furthermore, the energy is invariant with respect to the interchange of all variables $s_i \leftrightarrow -s_i$. In particular, here the global minimum occurs as $s_1 = -1$, $s_2 = +1$ and $s_3 = -1$ as well as the symmetric configuration $s_1 = +1$, $s_2 = -1$, and $s_3 = +1$ (red dots). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

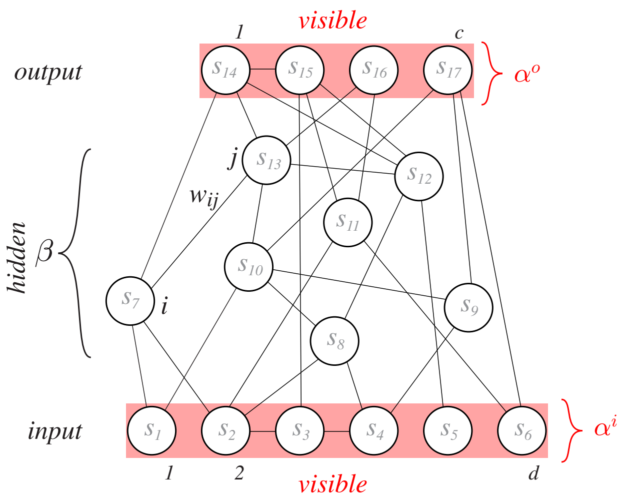


FIGURE 7.7. When a network such as shown in Fig. 7.1 is used for learning, it is important to distinguish between two types of visible units—the d input units and c output units, which receive external feature and category information—as well as the remaining, hidden units. The state of the full network is indexed by an integer γ, and because here there are 17 binary nodes, γ is bounded $0 \leq \gamma < 2^{17}$. The state of the input visible nodes is described by α^i and the output visible nodes by α^o. (The superscripts are not indexes, but merely refer to the input and output, respectively.) In the case shown, α^i is bound in the range $0 \leq \alpha^i < 2^d$ and α^o is bound in the range $0 \leq \alpha^0 < 2^c$. The state of the hidden nodes is indexed by β, which is bound in the range $0 \leq \beta < 2^{N-c-d}$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

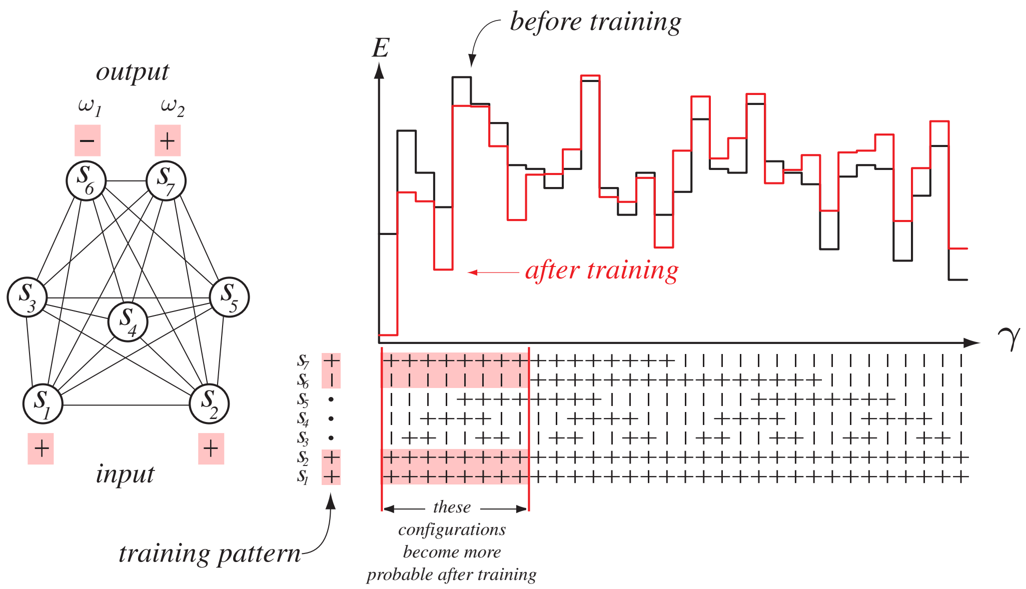


FIGURE 7.8. The fully connected seven-unit network on the left is being trained to assign the input pattern $s_1 = +1, s_2 = +1$ to class ω_2 by using the Boltzmann learning algorithm. During training, the output s_7 (corresponding to ω_2) is clamped at $+1$, and the output s_6 (corresponding to ω_1) is clamped at -1. All $2^5 = 32$ configurations with $s_1 = +1$, $s_2 = +1$ are shown at the right, along with their energy (Eq. 1). The black curve shows the energy before training; the red curve shows the energy after training. Note particularly that after training, all configurations that represent the full training pattern have been lowered in energy and hence are more probable; most other patterns become *less* probable after training. Thus, after training, if the input pattern $s_1 = +1$, $s_2 = +1$ is presented and the remaining network is annealed, there is an increased chance of yielding the outputs $s_6 = -1$ and $s_7 = +1$, as desired. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

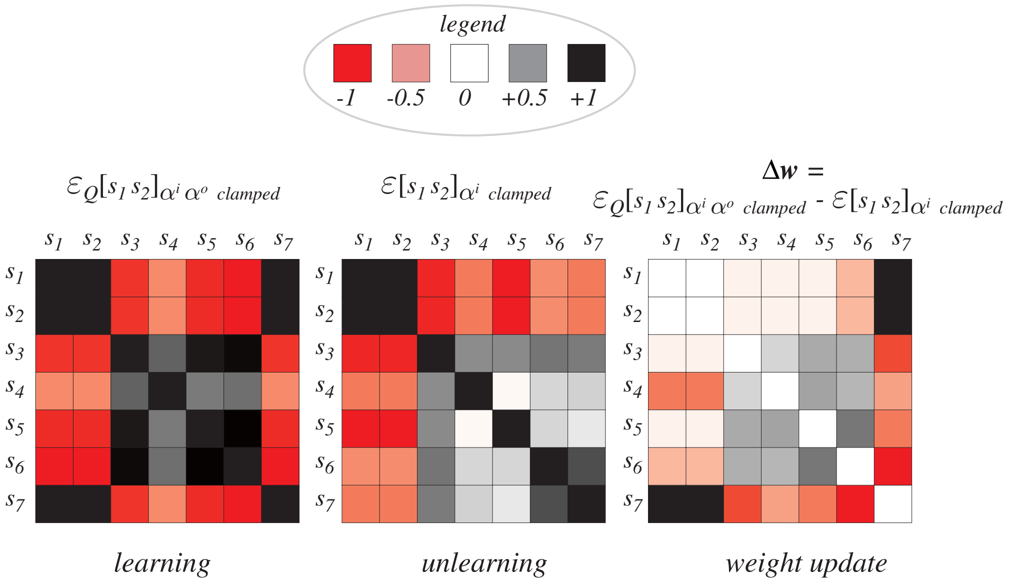


FIGURE 7.9. Boltzmann learning of a single pattern is illustrated for the seven-node network of Fig. 7.8. The (symmetric) matrix on the left shows the correlation of units for the learning component, where the input units are clamped to $s_1 = +1$, $s_2 = +1$ and the outputs are clamped to $s_6 = -1$, $s_7 = +1$. The middle matrix shows the unlearning component, where the inputs are clamped but outputs are free to vary. According to Eq. 14, the weight update should be proportional to the difference between those matrices, as indeed is shown on the right matrix. Notice, for instance, that because the correlation between s_1 and s_2 is large in both the learning and unlearning components (because those variables are clamped), there is no associated weight change, that is, $\Delta w_{12} = 0$. However, strong correlations between s_1 and s_7 in the learning but not in the unlearning component implies that the weight w_{17} should be increased, as can be seen in the weight update matrix. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

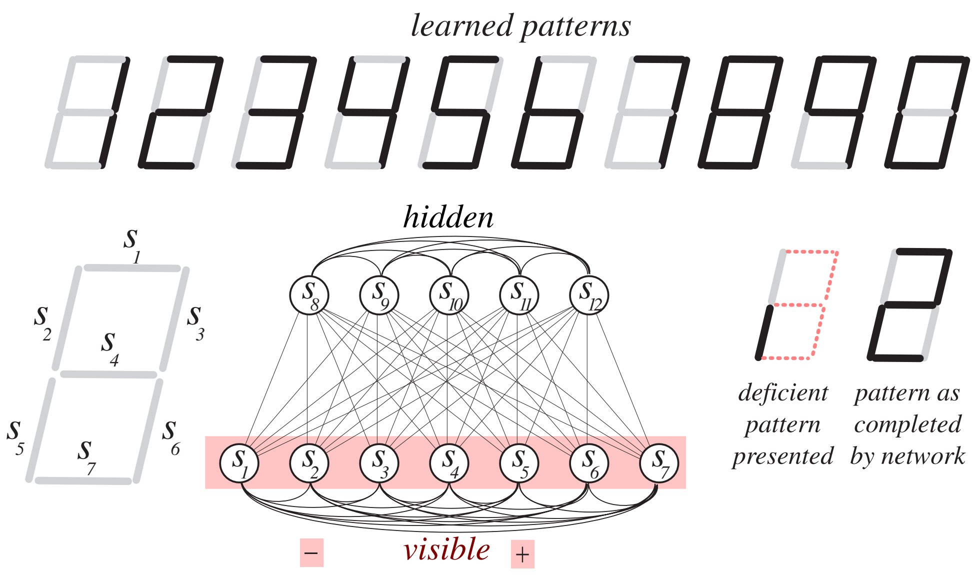


FIGURE 7.10. A Boltzmann network can be used for pattern completion—that is, filling in unknown features of a deficient pattern. Here, a 12-unit network with five hidden units has been trained with the 10 numeral patterns of a seven-segment digital display. The diagram at the lower left shows the correspondence between the display segments and nodes of the network. Along the top, a black segment is represented by a $+1$, and a light gray segment is represented as a -1. Consider the deficient pattern consisting of $s_2 = -1$, $s_5 = +1$, but with the other five inputs (shown as dotted lines in the pattern) unspecified. If these units are clamped and the full network is annealed, the remaining five visible units will assume values most probable given the clamped ones, as shown at the right. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

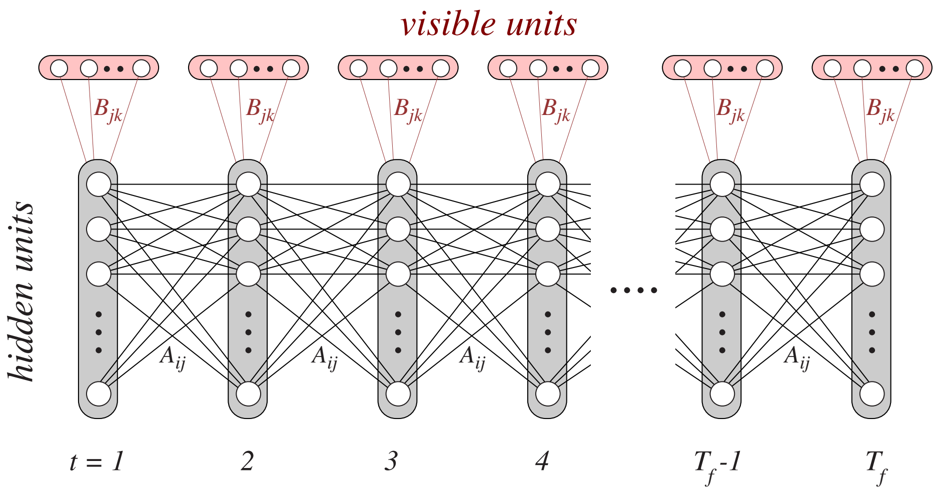


FIGURE 7.11. A Hidden Markov Model can be "unfolded" in time to show a trellis, which can be represented as a Boltzmann chain, as shown. The discrete hidden states are grouped into vertical sets, fully interconnected by weights A_{ij} (related to the Hidden Markov Model transition probabilities a_{ij}). The discrete visible states are grouped into horizontal sets, and are fully interconnected with the hidden states by weights B_{jk} (related to transition probabilities b_{jk}). Training the net with a single pattern, or list of T_f visible states, consists of clamping the visible states and performing Boltzmann learning throughout the full network, with the constraint that each of the time shifted weights labeled by a particular A_{ij} have the same numerical value. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

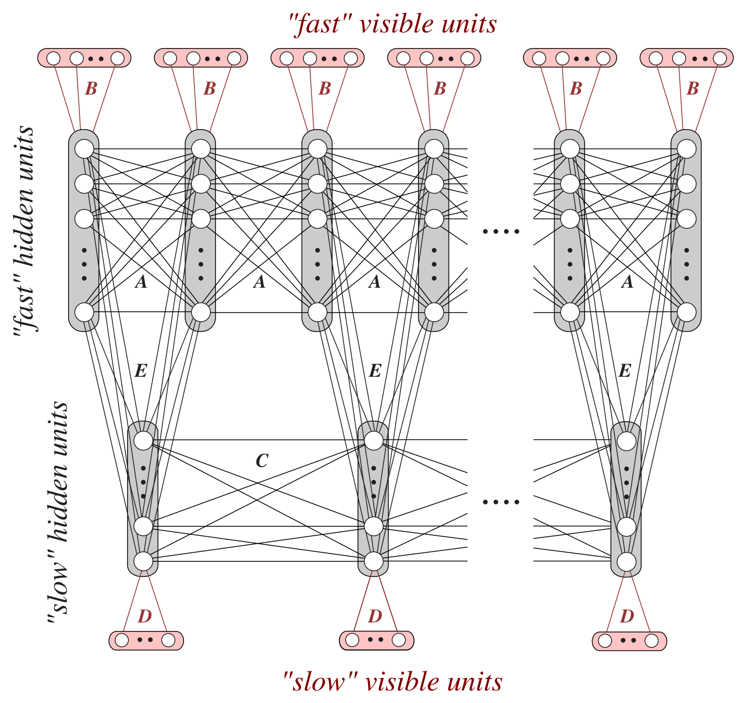


FIGURE 7.12. A Boltzmann zipper consists of two Boltzmann chains (cf. Fig. 7.11), whose hidden units are interconnected. The component chains differ in the rate at which visible features are sampled, and thus they capture structure at different temporal scales. Correlations are learned by the weights linking the hidden units, here labeled **E**. It is somewhat more difficult to train linked Hidden Markov Models to learn structure at different time scales. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

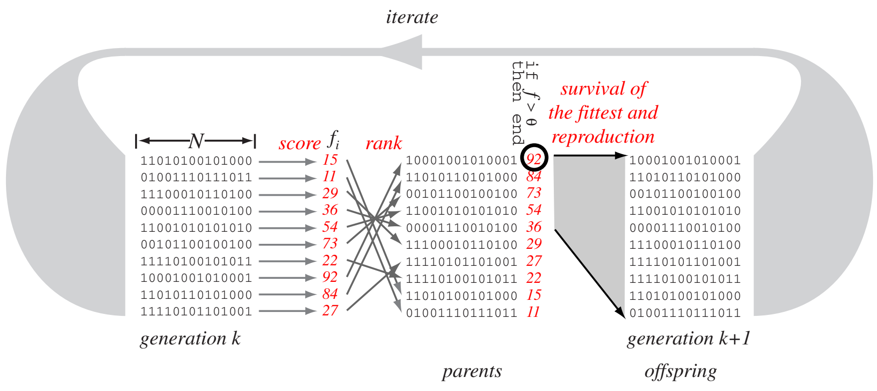
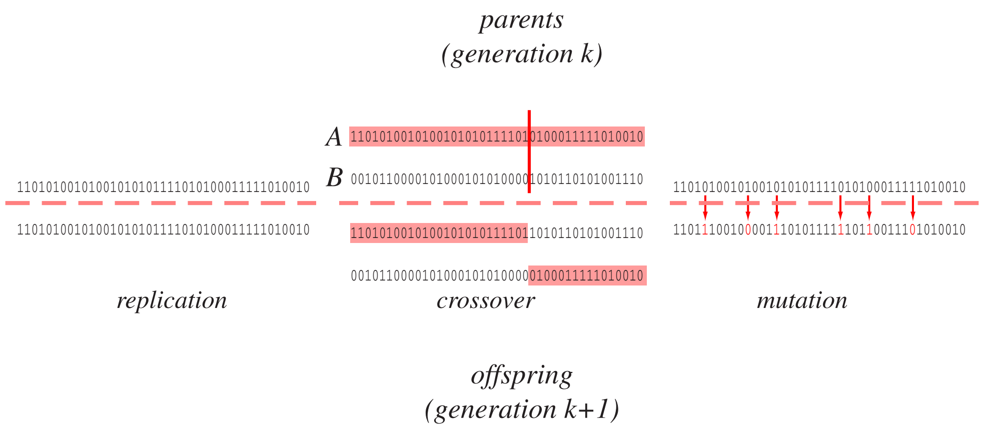


FIGURE 7.13. A basic genetic algorithm is a stochastic iterative search method. Each of the L classifiers in the population in generation k is represented by a string of bits of length N, called a chromosome (on the left). Each classifier is judged or scored according its performance on a classification task, giving L scalar values f_i. The chromosomes are then ranked according to these scores. The chromosomes are considered in descending order of score and are operated upon by the genetic operators of replication, crossover, and mutation to form the next generation of chromosomes—the offspring. The cycle repeats until a classifier exceeds the criterion score θ. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.



parents
(generation k)
A 110101001010010101011110101000111111010010
B 001011000010100010101000010101101010001110
replication
crossover
mutation
offspring
(generation k+1)

FIGURE 7.14. Three basic genetic operations are used to transform a population of chromosomes at one generation to form a new generation. In replication, the chromosome is unchanged. Crossover involves the mixing or "mating" of two chromosomes to yield two new chromosomes. A position along the chromosomes is chosen randomly (red vertical line); then the first part of chromosome *A* is linked with the last part of chromosome *B*, and vice versa. In mutation, each bit is given a small chance of being changed from a 1 to a 0 or vice versa. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

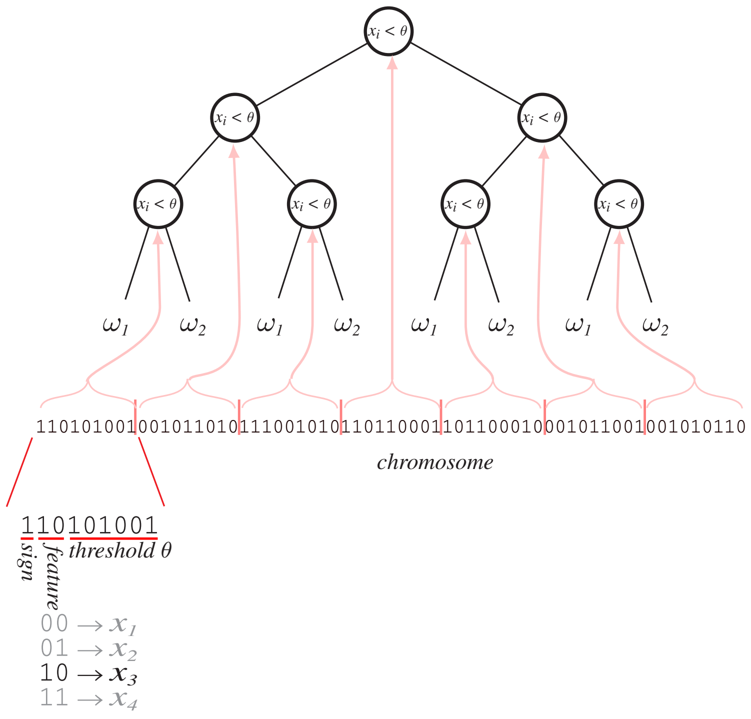


FIGURE 7.15. One natural mapping in genetic algorithms for pattern recognition is from a binary chromosome to a binary tree classifier, illustrated here for a simple binary decision tree. In this example, each of the nodes computes a query or question of the form "Is $\pm x_i < \theta$?" and is governed by nine bits in the chromosome. The first bit specifies a sign, and the next two bits specify the feature queried. The remaining six bits are a binary representation of the threshold θ. For instance, the leftmost node encodes the rule "Is $+x_3 < 41$?" (In practice, larger trees would be used for problems with four features.) During classification, a test pattern is presented to the top decision node, and depending upon the answer, passes to the right or to the left through the tree to a node at the next level. This process continues until the pattern comes to a category label (cf. Chapter 8). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

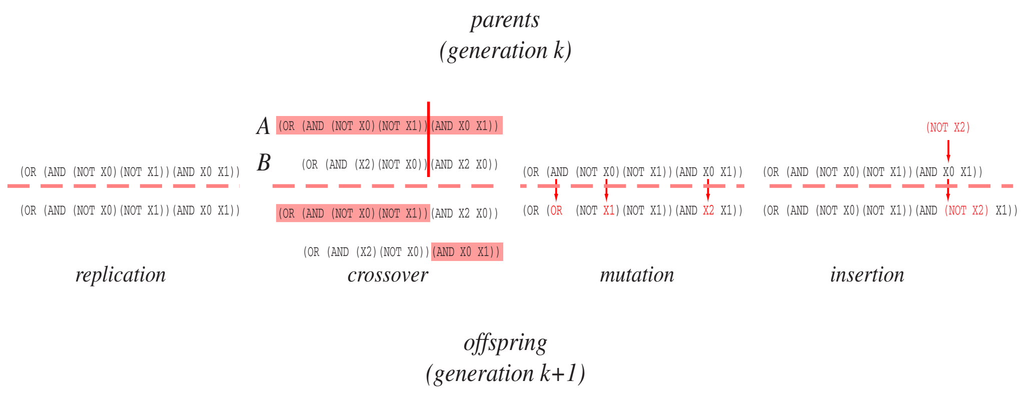

parents
(generation k)
A
(OR (AND (NOT X0)(NOT X1))(AND X0 X1))
B
(OR (AND (X2)(NOT X0))(AND X2 X0))
(OR (AND (NOT X0)(NOT X1))(AND X0 X1))
(OR (AND (NOT X0)(NOT X1))(AND X0 X1))
(OR (AND (NOT X0)(NOT X1))(AND X2 X0))
(OR (AND (X2)(NOT X0))(AND X0 X1))
(OR (AND (NOT X0)(NOT X1))(AND X0 X1))
(OR (OR (NOT X1)(NOT X1))(AND X2 X1))
(NOT X2)
(OR (AND (NOT X0)(NOT X1))(AND X0 X1))
(OR (AND (NOT X0)(NOT X1))(AND (NOT X2) X1))
replication
crossover
mutation
insertion
offspring
(generation k+1)

FIGURE 7.16. Four basic operators in genetic programming are used to transform a population of snippets of code at one generation to form a new generation. In replication, the snippet is unchanged. Crossover involves the mixing or "mating" of two snippets to yield two new snippets. A position along the snippet *A* is randomly chosen from the allowable locations (red vertical line); likewise one is chosen for snippet *B*. Then the front portion of *A* is spliced to the back portion of *B* and vice versa. In mutation, each element is given a small chance of being changed. There are several different types of elements, and replacements must be of the same type. For instance, only a number can replace another number; only a numerical operator that takes a single argument can replace a similar operator, and so on. In insertion, a randomly selected element is replaced by a compatible snippet, keeping the entire snippet grammatically well formed and meaningful. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

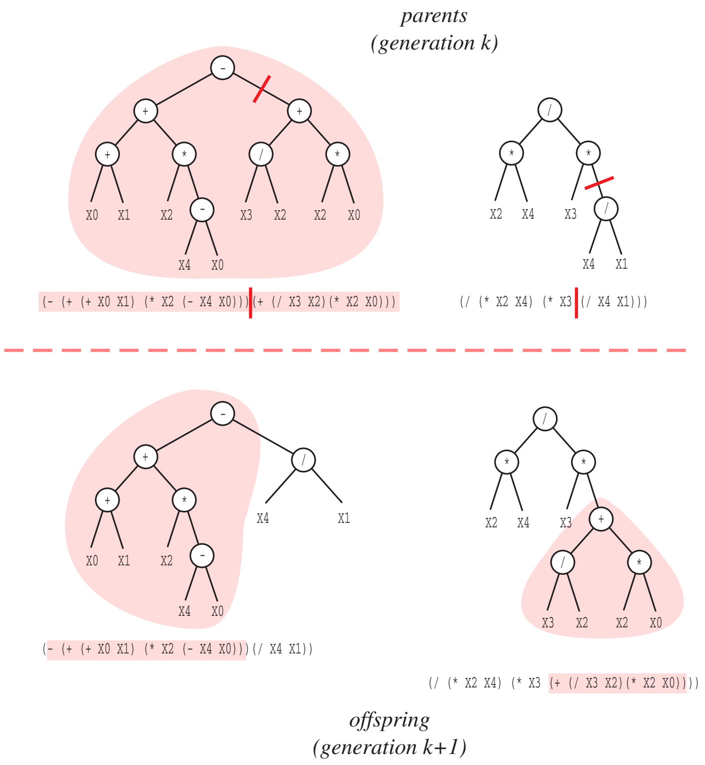

parents
(generation k)
(- (+ (+ X0 X1) (* X2 (- X4 X0))) (+ (/ X3 X2) (* X2 X0)))
(/ (* X2 X4) (* X3 (/ X4 X1)))
(- (+ (+ X0 X1) (* X2 (- X4 X0))) (/ X4 X1))
(/ (* X2 X4) (* X3 (+ (/ X3 X2) (* X2 X0))))
offspring
(generation k+1)

FIGURE 7.17. Unlike the decision trees of Fig. 7.15 and Chapter 8, the trees shown here are merely a representation using the syntax of *Lisp* expressions that implement a single function. For instance, the upper-right (parent) tree implements $(x_2 x_4)/(x_3(x_4/x_1))$. Two parent snippets and break points chosen randomly from allowable ones are shown by the red line segments. The offspring are formed by the crossover operation, where the left side of the snippet for parent 1 is concatenated to the right side of the snippet from parent 2, and vice versa. The resulting functions have an implied threshold or sign function when used for classification. Thus the function will operate on the features of a test pattern and emit category ω_i if the function is positive, and `NOT` ω_i otherwise. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.