**Deterministic methods for recognition – Methods for exact and/or elastic comparison:**

## Simulated annealing

Stochastic simulated annealing is slow, in part because of the discrete nature of the search through the space of all configurations, i.e., an *N*-dimensional hypercube. Each trajectory is along a *single* edge, thereby missing full gradient information that would be provided by analog state values in the “interior” of the hypercube. An alternate, faster method is to allow each node to take on *analog* values during search; at the end of the search the values are forced to be *si* = ±1, as required by the optimization problem. Such a *deterministic* simulated annealing algorithm also follows from the physical analogy. Consider a single node (magnet) *i* connected to several others; each exerts a force tending to point node *i* up or down. In deterministic annealing we sum the forces and give a continuous value for *si*. If there is a large “positive” force, then *si ≈* +1; if a large negative force, then *si ≈ −*1. In the general case *si* will lie between these limits.









# Stochastic Boltzmann learning of visible states

Learning categories from training patterns — consider an alternate learning problem where we have a set of desired probabilities for *all* the visible units, *Q*(*α*) (given by a training set), and seek weights so that the actual probability *P*(*α*), achieved in random simulations, matches these probabilities over all patterns as closely as possible. In this alternative learning problem the desired probabilities would be derived from training patterns containing both input (feature) and output (category) information. The actual probability describes the states of a network annealed with neither input nor output variables clamped. We now make use of the distinction between configurations of “visible” units (the input and output, denoted *α*), and the hidden states, denoted *β*, shown in Fig. 7.1. For instance, whereas *a* and *b* (c.f., Eq. 4) refered to different configurations of the full system, *α* and *β* sill specify visible and hidden configurations. The probability of a visible configuration is the sum over all possible hidden configurations:





where *Eαβ* is the system energy in the configuration defined by the visible and hidden parts, and *Z* is again the full partition function.

A natural measure of the difference between the actual and the desired probability distributions is the relative entropy, Kullback-Leibler distance or Kullback-Leibler divergence,



Learning is based on gradient descent in the relative entropy. A set of training patterns defines *Q*(*α*), and we seek weights so that at some temperature *T* the actual distribution *P*(*α*) matches *Q*(*α*) as closely as possible. Thus we take an untrained network and update each weight according to:

  
where *η* is a learning rate. While *P* depends on the weights, *Q* does not, and thus we used *∂Q*(*α*)*/∂wij* = 0.



**Mutual (cross) correlation between an image (object) and standard (example):**

**Fast computations in the frequency (Fourier) space:**