CHAPTER V

Annealing by Stochastic Neural Networks for Optimization

Two major classes of optimization techniques are the deterministic gradient methods and stochastic annealing methods. Gradient descent algorithms are greedy algorithms, which are subject to a fundamental limitation of being easily trapped in local minima of the cost function.

Hopfield networks usually converge to a local minimum of energy function. Because of its deterministic input-output relation of the units in the network, the network is not able to escape from local minima. Although such a behavior may be desirable for an associative memory application, one usually needs to obtain the global optimum or a nearly optimum points for optimization applications.

This problem is overcome by the use of stochastic annealing algorithms since they provide opportunity to escape from local minima. A highly attractive feature of the Boltzmann machine is its capability of escaping local minima through a relaxation technique based on simulated annealing [Hinton et al 83].

However, the use of simulated annealing is also responsible for an excessive computation time requirement that has hindered experimentation with the Boltzmann machine. Not only does simulated annealing require iterations at a sequence of temperatures that defines the annealing cycle but also each iteration requires many sweeps of its own. In order to overcome this major limitation of the Boltzmann machine a mean field approximation may be used, according to which the binary state stochastic neurons of the Boltzmann machine are replaced by deterministic mean values [Amit et al 85].

In this chapter, first we introduce the conventional Simulated Annealing algorithm for the solution of combinatorial optimization problems. Then two stochastic networks namely, Boltzmann machine and mean field network, are introduced and it is explained how the annealing technique can be implemented by using these networks.

The Gaussian machine [Aiker et al 91], which is a stochastic neural network developed over the continuous Hopfield model allowing escape from local minima, is also included .

5.1. Statistical Mechanics and the Simulated Annealing

The starting point of statistical mechanics is an energy function. We consider a physical system with a set of probabilistic states $\chi=\{\mathbf{x}\}$, each of which has energy $E(\mathbf{x})$. For a system at a temperature $T>0$, its state χ varies with time, and quantities such as E that depend on the state fluctuates. Although there must be some driving mechanism for these fluctuations, part of the idea of temperature involves treating them as random. When a system is first prepared, or after a change of parameters, the fluctuations has on average a definite direction such that the energy E decreases. However, sometimes later, any such trend ceases and the system just fluctuates around a constant average value. Then we say that the system is in ***thermal equilibrium***.

A fundamental result from physics tells us that in thermal equilibrium each of the possible states $\mathbf{x}$ occurs with probability, determined according to ***Boltzmann-Gibbs*** distribution,

$$P(\mathbf{x}) = \frac{1}{Z} e^{-\frac{E(\mathbf{x})}{T}} \tag{5.1.1}$$

where the normalizing factor

$$Z = \sum_{\mathbf{x}} e^{-\frac{E(\mathbf{x})}{T}} \tag{5.1.2}$$

is called the ***partition function*** and it is independent of the state **x** but temperature.

The Boltzmann-Gibbs distribution is usually derived from very general assumptions about microscopic dynamics of materials. The coefficient T is related to absolute temperature T_a of the system as

$$T = \kappa_B T_a \tag{5.1.3}$$

where coefficient κ_B is ***Boltzmann's constant*** having value 1.38×10^{-16} erg/K. Interestingly enough, the same distribution can also be achieved in the viewpoint of information theory. Although the temperature T has no physical meaning in information theory, it is interpreted as a pseudo temperature in an abstract manner.

Given a state distribution function $fd(\chi)$, let $P(\chi(k)=\mathbf{x}^i)$ be the probability of the system being at state $\mathbf{x}i$ at the present time k. Furthermore, let $P(\chi(k+1)=\mathbf{x}^j \mid \chi(k)=\mathbf{x}^i)$ represent the conditional probability of next state $\mathbf{x}^j$ given the present state is $\mathbf{x}^i$. The notation $P(\mathbf{x}^i)$ and $P(\mathbf{x}^j \mid \mathbf{x}^i)$ will be used simply to denote these probabilities respectively. In equilibrium the state distribution and the state transition reaches a balance satisfying:

$$P(\mathbf{x}^j \mid \mathbf{x}^i)P(\mathbf{x}^i) = P(\mathbf{x}^i \mid \mathbf{x}^j)P(\mathbf{x}^j) \tag{5.1.4}$$

or equivalently

$$\frac{P(\mathbf{x}^i)}{P(\mathbf{x}^j)} = \frac{P(\mathbf{x}^i \mid \mathbf{x}^j)}{P(\mathbf{x}^j \mid \mathbf{x}^i)} \tag{5.1.5}$$

Therefore, in equilibrium the Boltzmann Gibbs distribution given by equation (5.1.1) results in:

$$P(\mathbf{x}^j \mid \mathbf{x}^i) = \frac{1}{1+e^{\Delta E/T}} \tag{5.1.6}$$

where $\Delta E = E(\mathbf{x}^j) - E(\mathbf{x}^i)$.

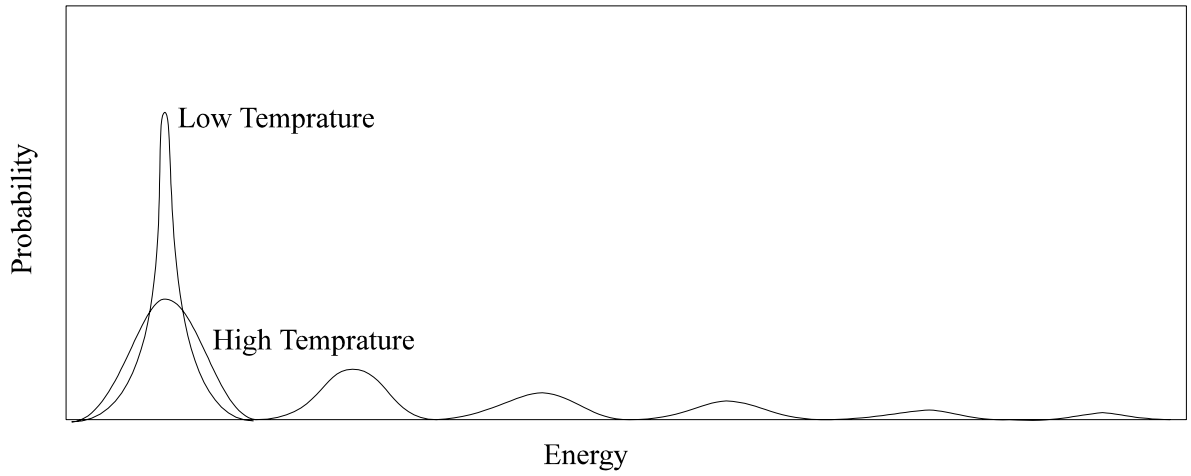

Figure 5.1 Relation between temperature and probability of the states [Kung 93]

The ***Metropolis algorithm*** provides a simple method for simulating the evolution of physical system in a heat bath to thermal equilibrium [Metropolis et al]. It is based on ***Monte Carlo Simulation*** technique, which aims to approximate the expected value $<g(\chi)>$ of some function $g(\chi)$ of a random vector χ with a given density function $f_d(\chi)$. For this purpose several χ vectors, say $\chi=\mathbf{X}^k$ $k=1..K$, are randomly generated according to the density function $f_d(\chi)$ and then $\mathbf{Y}^k$ is found as $\mathbf{Y}^k=g(\mathbf{X}^k)$. By using the strong law of large numbers:

$$\lim_{K\to\infty} \frac{1}{K} \sum_k \mathbf{Y}^k =< \mathbf{Y}^k >=< g(\chi) > \quad (5.1.7)$$

the average of generated $\mathbf{Y}$ vectors can be used as an estimate of $<g(\chi)>$ [Sheldon 1989].

In each step of the Metropolis algorithm, an atom (unit) of the system is subjected to a small random displacement, and the resulting change ΔE in the energy of the system is observed. If $\Delta E \leq 0$, then the displacement is accepted, and the new system configuration with the displaced atom is used as the starting point for the next step of the algorithm. If, on the other hand, $\Delta E > 0$, then the algorithm proceeds in a probabilistic manner so that the configuration with the displaced atom is accepted with a probability given by:

$$P(\Delta E) = e^{-\Delta E/T} \quad (5.1.8)$$

Provided enough number of transitions in the Metropolis algorithm, the system reaches thermal equilibrium. Thus, by repeating the basic steps of Metropolis algorithm, we effectively simulate the motions of the atoms of a physical system at temperature T. Moreover, the choice of $P(\Delta E)$ defined in Eq. (5.1.8) ensures that thermal equilibrium is characterized by the Boltzmann-Gibbs distribution provided in Eq. (5.1.5).

Referring to Eq. (5.1.5), notice that if $P(\mathbf{x}^i) > P(\mathbf{x}^j)$ implies $E(\mathbf{x}^i) < E(\mathbf{x}^j)$, and vice versa. So maximizing the probability function is equivalent to minimizing the energy function. Furthermore, notice that this property is independent of the temperature, although the discrimination becomes more apparent as the temperature decreases (Figure 5.1).

Therefore, the temperature parameter T provides a new free parameter for steering the step size toward the global optimum. With a high temperature, the equilibrium can be reached more rapidly. However, if the temperature is too high, all the states will have a similar level of probability. On the other hand, when $T \rightarrow 0$, the average state becomes very close to the global minimum. This idea, though very attractive at the first glance, can not be implemented directly in practice. In fact, with a low temperature, it will take a very long time to reach equilibrium and, more seriously, the state is more easily trapped by local minima. Therefore, it is necessary to start at a high temperature and then decrease it gradually. Correspondingly, the probable state then gradually concentrate around the global minimum (Figure 5.2).

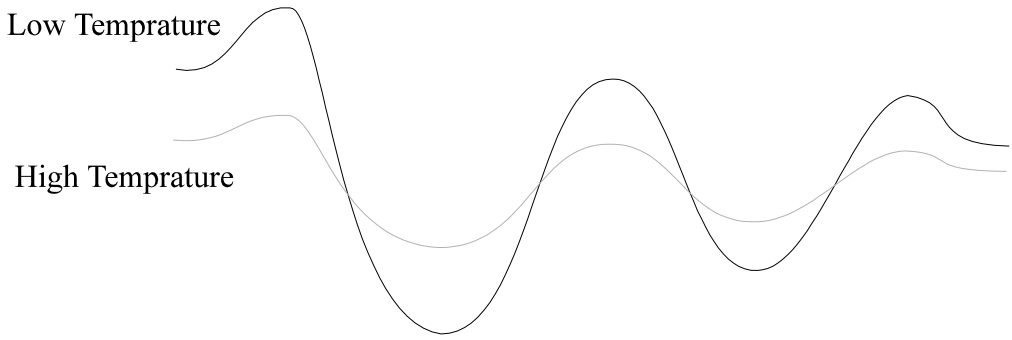

Figure 5.2 The energy levels adjusted for high and low temperature

This has an analogy with metallurgical annealing, in which a body of metal is heated near to its melting point and is then slowly cooled back down to room temperature. This

process eliminates dislocations and other crystal lattice disruptions by thermal agitation at high temperature. Furthermore, it prevents the formation of new dislocations by cooling the metal very slowly. This provides necessary time to repair any dislocations that occur as the temperature drops. The essence of this process is that global energy function of the metal will eventually reach an absolute minimum value.

If the material is cooled rapidly, its atoms are often captured in unfavorable locations in the lattice. Once the temperature has dropped far below the melting point, these defects survive forever, since any local rearrangements of atoms costs more energy than whatever available in thermal fluctuations. The atomic lattice thus remains captured in a local energy minimum. In order to escape from local minima and to have the lattice in the global energy minimum, the thermal fluctuations can be enhanced by reheating the material until energy-consuming local rearrangements occur at a reasonable rate. The lattice imperfections then start to move and annihilate, until the atomic lattice is free of defects-except for those caused by thermal fluctuations. These can be gradually reduced if the temperature is decreased so slowly that thermal equilibrium is maintained at all times during the cooling process. How much time must be spent for the cooling process depends on the specific situation. A great deal of experience is required to perform the annealing in an optimal way. If the temperature is decreased quickly, some thermal fluctuations are frozen in. On the other hand, if one proceeds too slowly, the process never ends.

The amazing thing about annealing is that the statistical process of thermal agitation leads to approximately the same final energy state. This result is independent of the initial condition of the metal and any of the details of the statistical annealing process. The mathematical concept of simulated annealing derives from an analogy with this physical behavior.

The ***simulated annealing*** algorithm, is a variant of the Metropolis algorithm in which the temperature is time dependent. In analogy with metallurgical annealing, it starts with a high temperature and gradually decreases it. At each temperature, it applies several times the update rule given by Eq. (5.1.8). An annealing schedule specifies a finite sequence of temperature values and a finite number of transitions attempted at each value of the temperature. The annealing schedule developed by [Kirkpatrick et al 1983] is as follows.

The initial value T_0 of the temperature is chosen high enough to ensure that virtually all proposed transitions be accepted by the simulated annealing algorithm. Then the cooling is performed. At each temperature, enough transitions are attempted so that there is a predetermined number of transitions per experiment on the average. At the end, the system is frozen and annealing stops if the desired number of acceptances is not achieved at predetermined number of successive temperatures. In the following, we provide the annealing procedure in more detail:

SIMULATED ANNEALING

Step 1. Set Initial values: assign a high value to temperature as $T(0)= T_0$, decide on constants κ_T, κ_A and κ_S, Typical values for which are $0.8<\kappa_T<0.99$, $\kappa_A=10$, and $\kappa_S=3$.

Step 2. Decrement the temperature: $T(k) = \kappa_T T(k-1)$ where κ_T is a constant smaller but close to unity.

Step 3. Attempt enough number of transitions at each temperature, so that there are κ_A accepted transitions per experiment on the average.

Step 4. Stop if the desired number of acceptances is not achieved at κ_S successive temperatures else repeat steps 2 and 3.

A very important property of simulated annealing is its asymptotic convergence. It has been proved in [Geman and Geman 84] that if $T(k)$ at iteration k is chosen such that it satisfies

$$T(k) \geq \frac{T_0}{\log(1+k)}, \qquad (5.1.9)$$

provided the initial temperature T_0 is high enough, then the system will converge to the minimum energy configuration.

The main drawback of simulated annealing is the large amount of computational time necessary for stochastic relaxation. Many elementary transformations are performed at each temperature step in order to reach a near equilibrium state.

5.2 Boltzmann Machine

Boltzmann machine [Hinton et al 83] is a connectionist model having stochastic nature. The structure of the Boltzmann machine is similar to Hopfield network, but it adds some probabilistic component to the output function. It uses simulated annealing concepts, in spite of the deterministic nature in state transition of the Hopfield network [Hinton et al 83, Aarts et al 1986, Allwright and Carpenter 1989, Laarhoven and Aarts 1987].

A Boltzmann machine can be viewed as a recurrent neural network consisting of N two-state units. Depending on the purpose, the states can be chosen from binary space, that is $\mathbf{x} \in \{0,1\}^N$ or from bipolar space $\mathbf{x} \in \{-1,1\}^N$. The energy function of the Boltzmann machine is:

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N} w_{ij} x_i x_j - \sum_{i}^{N} \theta_i x_i \qquad (5.2.1)$$

The connections are symmetrical by definition, that is $w_{ij}=w_{ji}$. Furthermore in the bipolar case, the convergence of the machine requires $w_{ii}=0$ (or equivalentlty $\theta_i=0$). However in the binary case self-loops are allowed.

The objective of a Boltzmann machine is to reach the global minimum of its energy function, which is the state having minimum energy. Similar to simulated annealing algorithm, the state transition mechanism of Boltzmann Machine uses a stochastic acceptance criterion, thus allowing it to escape from its local minima. In a sequential Boltzmann machine, units change their states one by one, while they change state all together in a parallel Boltzmann machine .

Let X denote the state space of the machine, that is the set of all possible states. Among these, the state vectors differing only one bit are called ***neighboring states***. The ***neighborhood*** $N_{\mathbf{x}} \subset X$ is defined as the set of all neighboring states of $\mathbf{x}$. Let a $\mathbf{x}^j$ to denote the neighboring state that is obtained from $\mathbf{x}$ by changing the state of *neuron* j. Hence, in binary case we have

$$x_i^j = \begin{cases} x_i & \text{if } i \neq j \\ 1-x_i & \text{if } i = j \end{cases} \qquad \mathbf{x} \in (0,1)^n, \mathbf{x}^j \in N_{\mathbf{x}} \tag{5.2.2}$$

In bipolar case, this becomes:

$$x_i^j = \begin{cases} x_i & \text{if } i \neq j \\ -x_i & \text{if } i = j \end{cases} \qquad \mathbf{x} \in (-1,1)^n, \mathbf{x}^j \in N_{\mathbf{x}} \tag{5.2.3}$$

The difference in energy when the global state of the machine is changed from $\mathbf{x}$ to $\mathbf{x}^j$ is:

$$\Delta E(\mathbf{x}^j | \mathbf{x}) = E(\mathbf{x}^j) - E(\mathbf{x}) \tag{5.2.4}$$

Note that the contribution of the connections w_{km}, $k \neq j$, $m \neq j$, to $E(\mathbf{x})$ and $E(\mathbf{x}^j)$ is identical, furthermore $w_{ij}=w_{ij}$. For the binary case, by using equations (5.2.1) and (5.2.2), we obtain

$$\Delta E(\mathbf{x}^j | \mathbf{x}) = (2x_j - 1)\left(\sum_i w_{ij} x_i + \theta_j\right) \qquad \mathbf{x} \in \{0,1\}^N \tag{5.2.5}$$

For the bipolar case it is

$$\Delta E(\mathbf{x}^j | \mathbf{x}) = (2x_j)\left(\sum_i w_{ij} x_i + \theta_j\right) \qquad \mathbf{x} \in \{-1,1\}^N, w_{ii} = 0 \tag{5.2.6}$$

Therefore, the change in the energy can be computed by considering only local information.

In a sequential Boltzmann machine, a trial for a state transition is a two-step process. Given a state $\mathbf{x}$, first a unit j is selected as a candidate to change state. The selection probability usually has uniform distribution over the units. Then a probabilistic function determines whether a state transition will occur or not. The state $\mathbf{x}^j$ is accepted with probability

$$P(\mathbf{x}^j | \mathbf{x}) = \frac{1}{1 + e^{\Delta E(\mathbf{x}^j | \mathbf{x})/T}} \tag{5.2.7}$$

where T is a control parameter having analogy in temperature. Initially the temperature is set large enough to accept almost all state transitions with probability close to 0.5, and then T is decreased in time to zero (Figure 5.3). With a proper cooling schedule, the sequential Boltzmann machine converges asymptotically to a state having minimum energy.

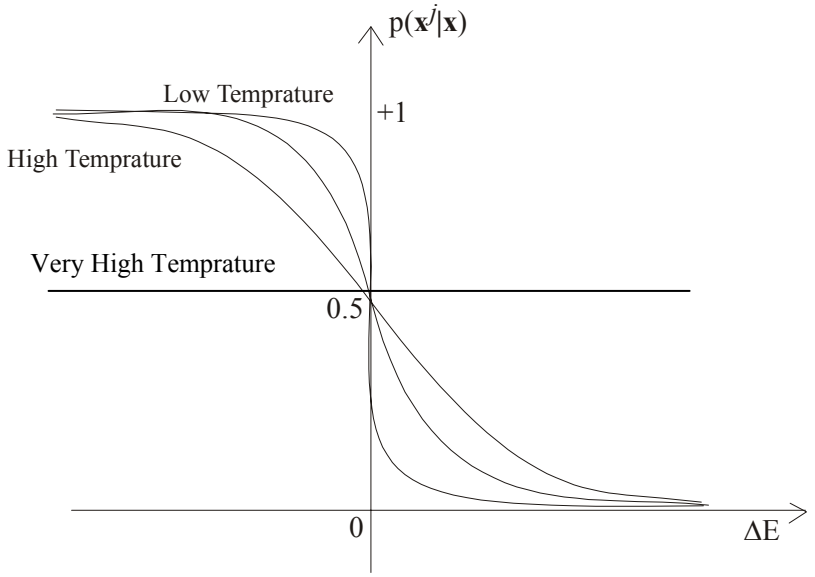

Figure 5.3. Acceptance probability in Boltzmann machine for different temperatures

A Boltzmann machine starts execution with a random initial configuration. Initially, the value of T is very large. A cooling schedule determines how and when to decrement the control parameter. As $T \to 0$, less and less state transitions occur. If no state transitions occur for a specified number of trials, it is decided that the Boltzmann machine has reached the final state.

A state $\mathbf{x}^* \in X$ *is called a* ***locally minimal state,*** if

$$\Delta E(\mathbf{x}^{*j} \,|\mathbf{x}^*) \geq 0 \quad j = 1..N \quad (5.2.8)$$

Note that, a local minimum is a state whose energy can not be increased by a single state transition. Let the set of all local minima be denoted by X^*. While the Hopfield network is trapped mostly in one of these local minima, the Boltzmann machine can escape from the local minima because of its probabilistic nature. Although the machine asymptotically

converges to a global minimum, the finite-time approximation of the Boltzmann Machine prevents guaranteeing convergence to a state with minimum energy. However, still the final state of the machine will be a nearly minimum one among X^*.

Use of Boltzmann machine as a neural optimizer involves two phases as it is explained for the Hopfield network in Chapter 4. In the first phase, the connection weights are determined. For this purpose, an energy function for the given application is decided. In the non-constrained optimization applications, the energy function can be directly obtained by using the cost function. However, in the case of constrained optimization, the energy function must be derived using both the original cost function and the constraints. The next step is to determine the connection weights $\{w_{ij}\}$ by considering this energy function. Then in the second phase, the machine searches the global minimum through the annealing procedure.

5.3 Mean Field Theory

Although the use of simulated annealing provides for escaping from the local minima, it results in an excessive computation time requirement that has hindered experimentation with the Boltzmann machine. In order to overcome this major limitation of the Boltzmann machine, a mean field approximation may be used. In mean field network, the binary state stochastic neurons of the Boltzmann machine are replaced by deterministic analogue neurons [Peterson and Anderson 87].

Mean-field approximation is a well-known concept in statistical physics [Glauber 63]. It can not be denied that in the context of a stochastic machine it would be desirable to know the states of the neurons at all time. However, we must nevertheless recognize that, in the case of a network with a large number of neurons, the neural states contain vastly more information than what is required in practice. In fact, to answer the most of the physical questions about the stochastic behavior of the network, we need only to know the average values of neural states. There is a close analogy between Hopfield Networks and some simple models of Magnetic Materials in statistical physics. The analogy becomes particularly useful when we generalize the networks to use stochastic units, as we considered in Boltzmann machine.

In real neural networks, neurons fire with a variable strength, and there are delays in synapses, random fluctuations from the release of transmitters in discrete vesicles, and so on. These are effects that we can loosely think of as noise, and crudely are represented by thermal fluctuations in the ising model.

The thermal fluctuations can be introduced into the Hopfield model by replacing the deterministic units of the Hopfield network with the stochastic units in an analogy to spins in Ising model of Magnetic materials [Hertz et al 91] .

Consider the discrete Hopfield network that we considered previously, where states were restricted to be $\mathbf{x} \in \{-1,1\}^n$. Now lets replace the deterministic units in this network by stochastic units [Hinton and Sejnowski 83, Peretto 84] that behave in the following manner:

$$\chi_i = \begin{cases} +1 & \text{with probability} \quad f_T(a_i) \\ -1 & \text{withprobability} \quad 1 - f_T(a_i) \end{cases} \qquad \mathbf{x} \in (-1,1)^N \qquad (5.3.1)$$

where $f_T(\alpha)$ is the sigmoid function:

$$f_T(a) = \frac{1}{1 + e^{-2a/T}} \quad (5.3.2)$$

Moreover, T is the pseudo temperature. Such a stochastic unit may be interpreted as ordinary deterministic threshold unit with a random threshold θ drawn from a probability density $f_T'(\theta)$.

Notice that the sigmoid function defined by equation (5.3.1) has the property

$$1 - f_T(a) = f_T(-a) \quad (5.3.5)$$

In the case of the network having a single element, the average value for $\chi = \chi_1$ can be calculated as:

$$\begin{aligned} <\chi> &= P(\chi=+1).(+1) + P(\chi=-1).(-1) \\ &= \frac{1}{1+e^{-2a/T}} - \frac{1}{1+e^{+2a/T}} = \frac{e^{a/T}-e^{-a/T}}{e^{a/T}+e^{-a/T}} \\ &= \tanh(a/T) \end{aligned} \qquad (5.3.6)$$

$f_h(a/T)$=tanh(a/T) function has the same kind of shape as $f_T(\alpha)$ except that it goes from -1 to 1 instead of 0 to 1. In fact

$$f_h(a/T) = \tanh(a/T) = 2f_T(a) - 1 \quad (5.3.7)$$

It should be not forgotten that, at a given time, χ itself is still either +1 or -1. It flips back and forth randomly between these two values, taking on one of them more frequently according to $f_T(a)$.

In the case of many interacting neurons, the problem is not easily solved. The evolution of a spin, represented by χ_i depends on the activation

$$\alpha_i = \sum_{j=1}^{n} w_{ji}\chi_i + \theta_i \quad (5.3.8)$$

which involves random variables χ_j that themselves fluctuate back and forth. In general, there is no way to solve the many-spin problem exactly. However, there is an approximation, known as the mean-field approximation, which often yields adequately good results. The basic idea of mean-field approximation is to replace the actual fluctuating activation potential α_j for each neuron j in the network by its average $<\alpha j>$ as:

$$\begin{aligned} <\alpha_i> &= <\sum_{j}^{n} w_{ij}\chi_j + \theta_i> \\ &\cong \sum_{j}^{n} w_{ij} <\chi_j> + \theta_i \end{aligned} \qquad (5.3.9)$$

Then, we can compute the average $<\chi_i>$ as in the case of single-unit problem:

$$< \chi_i >= \tanh(< \alpha_i > / T) \cong \tanh(\tfrac{1}{T} \sum_j^n w_{ij} < \chi_i > + \tfrac{1}{T} \theta_i) \qquad (5.3.10)$$

These are still N non-linear equations in N unknowns, but at least they no longer involve stochastic variables

The mean field approximation often becomes exact in the limit of *infinite range interaction*s, where each spin interacts with all the others. Crudely speaking this is because α_i is then the sum of too many terms, and then the central limit theorem can be applied. Even for short range interactions, for which $w_{ij} \approx 0$ if spin i and j are more than a few lattice sites apart, mean field theory can often give a good qualitative description of the phenomena.

Mean Field Network was originally proposed by [Amit et al 85] for being used as associative memory, and examined in a series of papers [Amit et al 1985b, 1987a, 1987b]. A detailed theoretical analysis on Mean Field Networks is included [Hertz et al 1991]. We will explain its use as neural optimizer in the next section.

5.4 Mean Field Annealing

As in the case of Hopfield or Boltzmann optimizers, the first step of using Mean Field network as a neural optimizer is to determine the connection weights by considering the cost function to be optimized. Therefore, the solution of the optimization problem is determined by the solution of the Eq. (5.3.10). In fact when the system has converged, the spins for which $<\chi_i> \geq 0$ have probability $P(\chi_i=1) \geq P(\chi_i= -1)$ and vice versa. Thus, a final decision process sets the spin values to -1 and 1 according to the mean values $<\chi_i>$. Contrary to simulated annealing, this method is intrinsically parallel by nature. The convergence process of the mean field algorithm is purely deterministic and is controlled by a dynamical system. This reduces the computational effort considerably.

The main drawback of this method is the difficulty in the choice of the parameter T. It has no major influence on the quality of the results when it is chosen in a certain range. In addition, the range of possible temperatures increases with the size of the optimization problem. In order to overcome the difficulty in choice of the ambient temperature, mean field annealing is proposed.

An approach in mean field annealing consists of annealing during the convergence of the mean field approximation. That is, the temperature is decreased slowly, while the coupled mean field equations for the averages $< \chi_j >$ are solved iteratively. [Soukoulis et al 83, Peterson and Anderson 87, Bilbro et al 89, G. L. Bilbro and W .E. Snyder 88, Van den Bout and Miller 89, Herault and Niez 89]. This can be done simply by using a continuous valued Hopfield network, in which the shape of the sigmoid output function changes as a function of temperature. Mean field annealing performs better than simulated annealing in several optimization problems [Van den Bout and Miller, 1988, 1989, Cortes and Hertz, 1989, Bilbro and Snyder 1989].

The temperature can be slightly decreased from a high value to a smaller one as soon as every neuron has been updated once [Herault and Niez 1989]. The system does not reach a near equilibrium state at any temperature during the convergence process but when the temperature is small enough, the system is frozen in a good stable state.

Another approach [Van den Bout and Miller 88] uses the critical temperature. During cooling process, there is a critical temperature T_c at which the mean field variables $< \chi_i >$ begin to move significantly towards +1 and -1. The principle is then to estimate theoretically this critical temperature and to let the system evolve at this temperature until equilibrium is reached. Then one decreases the temperature to near zero and iterates until the system has reached a near equilibrium state.

The results obtained by mean field annealing process are comparable to those obtained by the stochastic relaxation of simulated annealing. Besides, the convergence time of mean field annealing is faster than it.

5.5. Gaussian Machine

An extension of the mean field annealing is the Gaussian machine, whose structure is the same as the continuous state asynchronous Hopfield network except the following properties:

- A Gaussian distributed random noise is added to each input.
- The gain of the amplifiers is time-variant.

The total input s_i to neuron i is given by formula

$$s_i = \sum_j w_{ji} x_i + \theta_i + \varepsilon \qquad (5.5.1)$$

where w_{ji} is the synaptic weight between neurons j and i, x_j is the output of neuron i, θ_i is the input bias and ε is a Gaussian distributed random noise having zero mean and σ^2 variance. This noise term breaks the determinism of each neuron in the Hopfield network and allows escaping from local minima. The deviation σ is defined as

$$\sigma = kT \qquad (5.5.2)$$

where k is a constant equal to $\sqrt{8/\pi}$ and T is the pseudo-temperature whose value decreases in time to zero.

Gaussian distribution, which is known also as normal distribution, has the following formula:

$$f(u) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(u-\mu)^2/2\sigma^2} \qquad (5.5.3)$$

where μ is the mean and has value 0 in our case.

The activation value a_i of neuron i is changed according to the difference equation

$$\frac{\Delta a_i}{\Delta t} = \frac{-a_i}{\tau} + s_i \qquad (5.5.4)$$

where τ is the time constant that can be set to 1 with no loss of generality. Δt has the range $0 < \Delta t \leq 1$.

For the simulation of the Gaussian machine, [Akiyama et al 91] uses asynchronous neuron updates. Changes of the outputs are propagated to the other neurons immediately. The unit of a time step is defined as the period for updating N neurons.

The output value x_i is determined by the sigmoid function

$$x_i = \frac{1}{1 + e^{-\frac{a_i}{A}}} \tag{5.5.5}$$

as defined previously, where $1/A$ is the gain of the curve and the value of A is decreased in time so that the sigmoid function becomes unit step function.

The Gaussian machine minimizes the same function as Hopfield model but can escape from local minima when the noise is sufficiently large. The parameters A and T are controlled by some "sharpening" and "annealing" schedules respectively. The activation level A is started from some large value and decreased in time towards zero. For large A, the output takes middle-range values between 0 and 1. This allows rough searching for energy minima allowing vague decisions. Decreasing A to zero towards the end of the simulation provides a means of assigning binary values to the outputs. A is controlled by the following hyperbolic scheduling:

$$A = \frac{A_0}{1 + t / \tau_A} \tag{5.5.6}$$

where A_0 is the initial value of A and τ_A is the time constant of the sharpening schedule.

The temperature T is also decreased in time, in turn it decreases the deviation of the noise ε. T is controlled by the following hyperbolic annealing schedule:

$$T = \frac{T_0}{1 + t / \tau_T} \tag{5.5.7}$$

where T_0 is the initial temperature and τ_T is the time constant of the annealing schedule, which may differ from τ_A.

The choice of the parameters A_0, T_0, τ_A and τ_T of the Gaussian machine are critical for a good performance in terms of convergence in a shorter time and convergence to a better minimum.