Evolution of Hopfield Type Networks for Object Extraction Using Simulated Annealing

Susmita De and Sankar K. Pal[1]

Department of Computer Science and Engineering,
Jadavpur University,
Calcutta — 700 032

[1]*Machine Intelligence Unit,*
Indian Statistical Institute,
203 B.T. Road,
Calcutta — 700 035

Abstract

An application of Simulated Annealing (SA) algorithm to evolve Hopfield type optimum neural network architectures for object background classification is demonstrated. Different optimization criteria involving minimization of energy value of the network, maximization of percentage of correct classification of pixels (pcc), and a combination of pcc and number of connections of the network (noc) have been considered. It is found that the elitist model of SA is better than its conventional counterpart.

1 Introduction

Simulated Annealing (SA) [1]- [3] is an evolutionary computing paradigm, modeled on the process found in nature. This algorithm can be considered as an optimization technique. It executes iteratively on assumed solutions and uses probabilistic rules for state transition.

The present article describes a method where the searching capability of SA is exploited to evolve Hopfield type optimum neural network architectures for object extraction [4]. A concept of elitist strategy is also introduced into SA and it is seen that the elitist model of SA improves the performance significantly.

2 Simulated annealing: an introduction

As the name signifies, *Simulated Annealing* (SA) simulates the concept of "annealing of physical systems" for solving optimization problems. In condensed matter physics, *annealing* is a process of slowly cooling a physical system from a sufficiently high temperature in order to obtain *thermal equilibrium* state with globally minimum energy. As is known from statistical mechanics, if the system is in thermal equilibrium at a given temperature T, then the probability $P_T(s)$ that the system is in a given state (configuration) s depends upon the energy value $E(s)$ of the state and follows the *Boltzmann distribution*:

$$P_T(s) = \frac{e^{\frac{-E(s)}{kT}}}{\sum_{s \in S} e^{\frac{-E(s)}{kT}}}, \qquad (1)$$

where k is the Boltzmann constant and S is the set of all possible configurations (stat ..). As the temperature decreases, this distribution generates lower energy states and when the temperature approaches zero, only the minimum energy state is feasible (have non-zero probability of occurrence).

In order to simulate this physical phenomenon in solving optimization problems, an analogy can be made in the following manner: the configuration/ state of a system corresponds to one of the assumed solutions of the problem. Energy value corresponds to the value of the objective function and a low energy state of the system represents a local minima. Temperature T is a control parameter of the problem. The algorithm starts with a high value of T (say, T_{max}) and an initial configuration. From this initial state a sequence of configurations (assumed solutions) is generated as follows. From any configuration i (having energy E_i) the neighborhood operator yields the configuration j (having energy E_j). Based on some probabilistic acceptance rule, the process continues either with the new state or with the old one. The neighborhood operation and the acceptance rule are repeated for a sufficient but fixed number of times (say, nt). Starting from the current

state i, the value of T is then lowered in steps and the system is allowed to execute the above described procedure nt times at each temperature. The algorithm is terminated with a small value of T (say, T_{min}) where actually no new configurations are accepted any more. The final configuration is considered to be the optimum solution of the problem.

A well known probabilistic acceptance rule, called as Metropolis simulation [2] is described below.

begin
$T = T_{max}$
select a random state s_c *and compute its fitness* $fit(s_c)$
while $(T \geq T_{min})$
  begin
  for $i = 1$ *to* nt
    begin
    pick an adjacent state s_n *from* s_c *at random and compute its fitness* $fit(s_n)$
    if $(fit(s_c) \geq fit(s_n))$
      set $s_c = s_n$
    else
      set $s_c = s_n$ *with probability* $(e^{\frac{fit(s_n) - fit(s_c)}{T}})$
    end
  decay T
  end
end

The above pseudo coded algorithm considers the problem of minimization and can be changed accordingly for maximization problem.

3 Evolution of Hopfield type neural network architectures for object extraction

In this section we first of all describe the principle of object extraction using Hopfield type neural networks. This is followed by evolutionary scheme based on SA for designing such networks.

3.1 Principle of object extraction using Hopfield type neural network

To use a Hopfield type neural network for object extraction [4], a neuron is assigned corresponding to every pixel. Each neuron can be connected to its neighbors (over a window) only. The connection can be full (a neuron is connected with all of its neighbors) or can be partial (a neuron may not be connected with all of its neighbors). The network topology for a fully connected third order neighborhood is depicted in Fig. 1. Here the maximum number of connections of a neuron with its neighbors is 8. In practice, all these connections may not exist. Again, different neurons may have different connectivity configuration within its neighbors. The

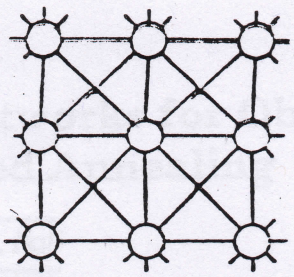

Figure 1: Topology of the neural network with third order connectivity (in the proposed system all connections may not exist)

energy function of this model has two parts. The first part is due to the local field or local feedback and the second part corresponds to the input bias of the neurons. In terms of images, the first part can be viewed as the impact of the gray levels of the neighboring pixels, whereas the second part can be attributed to the gray value of the pixel under consideration. The total energy contributed by all pixel pairs will be $-\sum_i \sum_j W_{ij} V_i V_j$, where V_i, V_j are the status of ith and jth neurons, respectively and W_{ij} is the connection strength between these two neurons.

For every neuron i, the initial input bias I_i and the initial state V_i are taken to be proportional to the actual gray level for the corresponding pixel. If a gray level of a pixel is high (low), the corresponding intensity value of the scene is expected to be high (low). The input bias value is taken in the range [-1 1]. Under this framework an ON (1) neuron corresponds to an object pixel and the OFF (-1) one to a background pixel. So the threshold between object and background can be taken as 0. Thus the amount of energy contributed by the input bias values is $-\sum_i I_i V_i$. So the expression of energy for this problem takes the form

$$Energy = -\sum_i \sum_j W_{ij} V_i V_j - \sum_i I_i V_i. \qquad (2)$$

Stable states of the network (the local minima of the energy function) are assumed to correspond to the partitioning of a scene into compact regions. So from a given initial state, the status of a neuron is modified iteratively to attain a stable state.

3.2 Architecture evolution using SA

Each state of an SA represents a possible network