Implementing semantic networks in stochastic neural nets

Luyuan Fang
Department of Computer Science
The Flinders University
Bedford Park SA 5042

Tao Li
Department of Computer Science
The University of Adelaide
Adelaide SA 5000

W. H. Wilson
Department of Computer Science
The Flinders University of South Australia
Bedford Park SA 5042

Abstract. A stochastic model of semantic memory is proposed. This model overcomes the difficulty of multiple pattern retrieval in a model proposed by Hinton. Our new model is also self-organizing with respect to the referential frequency of stored items. The dynamic nature of neural memory is captured in the stochastic model. Various kinds of retrievals can be realized on the basis of stochastic transition. The result is a dynamic and flexible model of parallel semantic memory.

1 Introduction

The recent revival of neural network research has brought together many disciplines in cognitive science such as cognitive psychology, artificial intelligence, computer science, neurophysiology, and cybernetics. This is not without any special reason. Neural network research is the ground for all these disciplines and ties together many issues about cognition. Many models (Hopfield 1982, Hopfield 1984, Hinton and Anderson 1981, Rumelhart and McClelland 1986, Cognitive Science Vol.9 1985) of neural networks have been proposed and some of these are being implemented using VLSI and optical circuits (Hopfield and Tank 1986, Sivilotti 1985, Psaltis and Farhat 1985). Neural networks are expected to bring a significant impact to cognitive science and will modify our view on computation. However, there are many problems remaining to be solved in this area, among which are the scaling problem and the complexity of interconnection networks.

Most of research in neural networks has been focused on either the learning ability of such networks or the storage of patterns for associative retrieval. One of such research was devoted to implementing semantic networks in synchronous neural nets. Hinton (1981) outlined an approach to store semantic networks in synchronous parallel hardware of the form of threshold neurons. A distributed implementation in which a node of a semantic net is represented by a pattern
of activity rather than a particular neuron was used in his work. Perceptron learning procedure (Minsky 1969) was applied to insert patterns into the parallel hardware.

According to our recent investigation, this approach has the advantages of high storage capacity (comparing to the asynchronous neural nets (Hopfield 1982 and 1984)) and high speed of associative search. In addition, it solves the problem of property inheritance by appropriate encoding of microfeatures. However, Hinton's model of synchronous neural nets was deterministic in nature. This made certain query patterns of associative search difficult or impossible. We extend Hinton's model to include stochastic selection of next states. Some of the original problems can be solved using our model.

2 A Model of Parallel Semantic Memory

We begin with the original model of Hinton. The parallel hardware consists of a network of perceptrons, each of which is a threshold logic unit with adjustable weights on the input ports. We will use the word neuron for perceptron from now on. There are three major subnets, ROLE1, REL, and ROLE2, in the associative memory for semantic network. ROLE1 and ROLE2 store two sets of nodes in a semantic network and REL stores the relations between the nodes in ROLE1 and ROLE2 (the edges that link the nodes). Hence, a relation is represented by a triple (R1,Rel,R2) where R1 is stored in ROLE1, R2 is stored in ROLE2 and Rel is stored in REL. There is an auxiliary subnet PROP. Since networks composed of single layer neurons have certain limitations (as pointed out by Minsky(1969)), hidden units must be introduced to overcome such limitation. The neurons in the PROP subnet are used as hidden units. The purpose of PROP subnet is to cause resonance states of the whole system in order to realize partial match retrieval. Details about the use of this subnet can be found in Hinton's paper (1981). Within each subnet (except PROP subnet), the neurons are completely connected, that is, there is a link from every neuron to every other neuron. The links in the PROP subnet and between PROP and the other three nets are randomly distributed. The weights of these links are also randomly initialized. This organization is shown in Fig 1.

Within each subnet, the stored patterns are inserted using the perceptron convergence procedure so they are relatively stable states within the subnet. However, these relatively stable states may change subject to the influence from other subnets and from the PROP subnet. For example, the three triples (one,greater,zero), (two,greater,one), and (zero,less,two) are stored in the network. If an initial state (one,greater,?), where "?" denotes a query, is inserted in the network, oneεROLE1 and greaterεREL will cause the PROP subnet to change into a state which actually encodes (one,greater,zero). This new PROP state will in turn affect the third subenet (for ROLE2) so that the state of ROLE2 will change from "?" to zero. Thus, partial retrieval is realized. Also, since one and greater are relatively stable, they will remain the unchanged dur-
Figure 1: Organization of the Neural Semantic Network

ing the partial retrieval process. The unstable pattern \( ? \) will be affected by the changes in PROP and will become zero. This illustrates the working principle of the system.

Property inheritance and other important operations can be accomplished by appropriate encoding of micro features. For example, if the code for person is 01100000, the code for a specific person smith will be 01100001 to reflect the fact that smith is an instance of person. Property inheritance is realized through the common micro code 0110. Relational features can also be implicitly encoded in the triples. This enables partial retrieval queries with set operations (discussed later).

3 A Stochastic Parallel Semantic Memory

Hinton’s deterministic parallel semantic memory also has several disadvantages. The major difficulty of that model is that it is impossible to retrieve multiple patterns, all of which satisfy the given query. This problem cannot be overcome in a deterministic model of distributed neural nets. Even if proper encoding of common micro features is chosen, the result will still be difficult to interpret.

On the other hand, a deterministic model is so mechanistic that it hardly captures the dynamic nature of neural nets. The process of thinking as exhibited in biological neural nets is very dynamic and the pattern of activity changes constantly. A deterministic machine will not be able to simulate this process at all. But a stochastic machine may approximate this process to a more realistic extent.

3.1 Defining the Stochastic Model

Hinton’s model can be viewed as a deterministic finite state machine. In his model, all neurons are activated by a single clock pulse. The system changes state when the clock is on. An \( n \times n \) matrix \( T \) (representing the connections
of a subnet such as ROLE1, ROLE2 and REL), where n is the number of neurons in the system and $T_{ij}$ in T denotes the weight on the link from neuron i to neuron j, determines the state transition. The state change is done by multiplying the matrix T with the current state vector $S = \{s_1, s_2, ..., s_n\}$ and the result $R = S \times T = \{r_1, r_2, ..., r_n\}$ is compared with the threshold vector $G = \{g_1, g_2, ..., g_n\}$. If $r_i > g_i$, neuron i will be turned on; otherwise, it will be turned off. This changes the system into a new state. This model has relatively high storage capacity (Li 1987, Fang 1987).

In order to convert Hinton's model to a stochastic model, a parameter must be added to describe the transition probability of the system. In addition, the transition probability must be modifiable to allow learning and self-organization. The use of a complete transition matrix with $2^n \times 2^n$ entries is apparently not realistic, especially for large n. For a richly connected neural net, the transition matrix will double the storage requirement. For a not richly connected net, the storage increase will be even higher. In Hinton's original model, the network was not richly connected.

The links in the three subnets ROLE1, REL, and ROLE2 are used to store stable patterns. Actually these patterns are relatively stable because they can be caused by the PROP subnet. The PROP subset produces global resonance to accomplish partial match retrieval. Hence we would like to associate another parameter $p_{ij}$ for each link $l_{ij}$ to and within the PROP subnet. This parameter is the probability for this link to be effective (the weight of which does not become zero). When the link is effective, the normal weight associated with it is used in computing the next states. When the link is not effective, the weight becomes zero (indicating that the link is not connected). Hence the parameter is called the effective probability. This new parameter can also be changed to allow stochastic learning. Since the new parameter is only associated to each link in the PROP subnet, the total storage increase is not significant for practical networks.

Although simple, this extension to neural semantic memory can be quite effective. Not only does it overcome the problem of multiple pattern retrieval but also it captures the dynamic behavior of biological neural networks. We are presently simulating this stochastic semantic network model for natural language processing applications.

3.2 Parameters and Adjustments for Learning

The most important parameter for our stochastic semantic memory model is the effective probability for links of the PROP subnet. This parameter controls whether a link is connected, which in turn controls the state transition. The higher the effective probability is for a link, the more times it will be connected and the more it will affect the state transition.

In order to realize self-organization, the effective probability is allowed to change. We choose to use a non-linear function for changing the effective probability. This curve is shown in Fig 2.
The referential rate $\theta_{ij}$ is defined by

$$\theta_{ij} = \delta + \frac{2(1-2\delta)R(i,j)}{t + R(i,j)}$$  \hspace{1cm} (1)

where $R(i,j)$ is the number of times the link $l_{ij}$ has responded to some particular patterns in (ROLE1, REL, ROLE2) and $t$ is the time elapsed. This definition of referential rate guarantees that $\theta_{ij}$ is always within the range $[\delta, 1-\delta]$, where $\delta$ is a small constant. Therefore, every link has a chance to become effective and no links can always be effective. Since each node $n_i$ in the PROP subnet responds to certain patterns (ROLE1, REL, ROLE2), the effective probability of the links of $n_i$ should be updated according to the curve in Fig 2 when those patterns appear. The curve is really a shifted sigmoid monotonic function. Hence the effective probability is defined as $P_{ij} = sigmoid(\theta_{ij})$.

The rule of updating the effective probability of the link $a_{ij}$ is given as follows.

- if $a_{ij} < 0$ and $n_i = 1 \land n_j = 0$, then $R(i,j) = R(i,j) + 1$.
- if $a_{ij} > 0$ and $n_i = 1 \land n_j = 1$, then $R(i,j) = R(i,j) + 1$.
- otherwise, $R(i,j) = R(i,j)$.

If proper micro features are chosen to encode relations in a semantic network, property inheritance can be easily realized. If the properties of a class are inherited by $n$ objects of the class, the micro features common to the class of objects will be referenced more frequently. Hence the nodes and links of the PROP subnet will respond to these micro features more actively. The effective probability for the links in the common micro features will become much higher. Under appropriate control, the links for common micro features tend to have
effective probability above the point $f_2$ and the links for individual objects tend to have effective probability below $f_1$. The overall effect is that common micro features are more stable than individual features. Therefore, multiple objects which satisfy the same query condition can be sequentially retrieved. This is because the individual features are less stable than common features and the patterns of individual objects can only stay for a short period.

Another important parameter is the length of time in which an individual pattern can stay there. This is the *individual stay time*. Whether an object can be effectively retrieved will depend on this parameter. If the pattern of an object stays for a very brief instant, it will be difficult to retrieve it. If the pattern of an object stays too long, it will have an affect on the common micro features (the common micro features will change). When some object takes an excess amount of time, other objects which satisfy the same conditions will not have enough time to appear before the patterns of common micro features change.

### 3.3 A Complete Semantic Memory Organization

In order to make the stochastic semantic memory applicable to natural language processing, we need a more complete system of semantic memory. We combine the stochastic semantic memory with a short term memory (consisting of a set of registers) to form a complete semantic memory system. This complete system is shown in Fig 3.

The long term memory is the stochastic semantic memory discussed in the
previous sections. The short-term memory consists of several registers which are also constructed using perceptrons. The queries for partial match retrieval are initially stored in the short-term memory. The query patterns are injected into the long-term memory from the registers. When a sequence of queries are injected, the long-term memory is allowed to stabilize for a short time. The short-term memory will then begin to pick up relatively stable patterns from the long-term memory. The transfer of patterns from long-term memory to short-term memory is also a self-adjusting process. A register must learn to adjust its link weights in order to store a relatively stable pattern. This learning process requires a short time and also has a filtering effect for random noise patterns.

4 Operations on Stochastic Semantic Network

We now discuss various operations which can be performed on the stochastic semantic network and the implementation of these operations. The main operations under consideration include the partial match retrievals of various types and the storage of new states in the network.

4.1 Partial Match Retrieval Queries

There are various types of queries which can be used for partial match retrieval in our parallel stochastic semantic network. Partial retrieval queries for neural networks cannot be as complicated and extensive as for relational data bases. It is extremely difficult to realize complicated queries in neural nets. Also, the purpose of semantic and associative memory is to retrieval data using pattern matching techniques rather than satisfying sophisticated constraints. Next we discuss several types of queries together with their realization in stochastic semantic networks.

The one pattern query represents the simple type. In one pattern query, a single (and partially specified) pattern is given. Each pattern consists of a triple of the form (R1,R,R2) where R1εROLE1, R2εROLE2, and RεREL. Any one or more item in the triple can be a “?” which represents an empty item. The empty item is to be retrieved from the semantic memory.

The result of a single pattern query can be either a complete pattern or a sequence of complete patterns (which are retrieved stochastically). For example, if the triples (JOHN, FATHER, TOD), (JOHN, FATHER, JANE) and (JOHN, FATHER, BOB) are stored, and a one pattern query (JOHN, FATHER,?) is placed to the stochastic semantic network, the three patterns will be stochastically retrieved one by one in a sequence. The effective probability plays the most important role in the retrieval of multiple patterns. Since the retrieval is stochastic, there is no absolute guarantee that all the relevant patterns can be retrieved. But there is a high probability that the relevant patterns can be completely retrieved.

Sometimes we would like to find triples which satisfy more constraints than one pattern query. For example, we would like to find which son of JOHN is
married to TERRY. We use two triples to represent the query (JOHN, FA-
THER, ?) and (TERRY, MARRIED, ?). The triple to be found must satisfy
both conditions. This is a conjunctive query. For a conjunctive query, each
query triple is placed to the stochastic semantic memory in turn (a short time
in between two contiguous patterns). The common micro features of both triples
will be combined to form a relatively stable pattern. The combined pattern will
cause resonance in the semantic network and the desired object sequence will
be retrieved.

If the triples to be retrieved are required to satisfy any one or more of many
given conditions, we have a disjunctive query. To realize this type of retrievals,
we first insert one of the conditions and wait until some relatively stable patterns
appear. These patterns are then collected in the short term memory. Another
condition is inserted and more triples are retrieved. If a newly retrieved triple
is not already in the short term memory, then it is stored there. Keep inserting
new conditions until no more left.

Both conjunctive and disjunctive queries require more than one condition.
These are the multiple input queries. Since the input and the output of par-
tial match retrievals may both contain multiple patterns (triples), they can be
classified by the numbers of input and output patterns as SISO (single input
pattern/single output pattern) retrieval, SIMO (single input pattern/multiple
output pattern) retrieval, MISO (multiple input pattern/single output pattern)
retrieval, and MIMO (multiple input pattern/multiple output pattern) retrieval.
Among multiple input retrievals there are also conjunctive and disjunctive re-
trievals. We do not discuss more complicated retrievals since they are not
practical to semantic networks constructed by stochastic neural models.

4.2 State Insertion and Storage Capacity

To store a semantic network into our stochastic memory, we must insert states
(i.e., triples) into its long term memory. For the time being, state insertion
is done with the perceptron convergence procedure (Minsky 1969). To insert
a state into the memory, the neurons in the three subnets ROLE1, REL, and
ROLE2 are clamped into the corresponding activity pattern. Then the weights
on the links of the three subnets are adjusted so that this activity pattern
becomes stabilized. This new state is inserted.

When the number of states inserted approaches the upper limit of the stor-
age capacity, inserting new states will result in the loss of some states stored
previously. Also, some spurious states may appear as more states are inserted.
After intensive experiments and simulation on the system, we found that this
model, coupled with the perceptron convergence procedure, has the advantages
of high storage capacity, fast convergence speed, and low spurious states.

For a network of N neurons, at least N states can be stored. Sometimes the
storage capacity may reach as high as 1.4N with a tolerable number of noise
states. But the asynchronous model of Hopfield (1982, 1984) has a storage ca-
pacity of 0.15N-0.25N (the 0.25N capacity can be reached through "unlearning"