Chapter 3

The Special Features of DNN

3.1 Introduction

The DNN has some interesting features besides being efficient for associative recall. The DNN architecture exhibits a novel characteristic, termed here as *relative pruning*, wherein the nodes are optionally and dynamically used (and reused) during the process of training. The dynamically organizing features has led us to term the network as DNN. The DNN has a composite structure wherein all nodes are similar and each node of the network is a Hopfield network by itself. During the training all the nodes may not participate constantly. Hence optimal utility of the nodes will definitely improve the efficiency of the network. This enabled us to introduce a new feature in DNN, i.e. *relative pruning* which provides the parallel reusability of the nodes in the DNN.

The learning rule of DNN exhibits another interesting feature of being sensitive to the order of presentation of training patterns. Artificial Neural Networks are biologically inspired and are introduced with the hope to reproduce some of the flexibility and power of human brain by artificial means. Researchers usually think of the functionality of the brain while considering new network configurations and their learning algorithms. Human brain is biased in memorizing and retrieving certain memories. One can find this characteristic in the learning rule of DNN, which is order-sensitive. The natural instinct of human learning is to have longer impression of patterns that are learnt earlier, which
in general represents order-sensitivity. In this chapter, these characteristics of the DNN namely, the *order-sensitivity* and *relative pruning* are discussed.

3.2 Order-Sensitive Learning

The human brain continuously learns from environment and memorizes the information, most probably in chronological order. When new memories are stored it may lose the old memories. But this process is not always like a pipeline, there are many exceptions to it. A priority is given to or bias is shown for certain memories, that is certain memories have longer impression and biased while retrieving. Emphasizing the same, the natural instinct of human learning is to have longer impression of patterns that are learnt earlier, "first impression is the best impression".

This biological phenomena is supported by many day to day observations. One can observe while a child is trying to learn words and recognize the words he is biased towards the words he has learnt first. In other words when he has to recognize a misspell word he may be inclined to match it with words he knew in the order he learnt them. For example a child is usually taught the word 'apple' for the letter 'a' first. He might learnt, many more words later starting with the letter 'a' like 'ape', 'ant' and 'ass' etc.. But when he is given a misspell word such as 'aple', he is more inclined to match it with 'apple' rather than with some other word like 'ape'. It should be noted that both 'apple' and 'ape' are equally likely corrections.

Order sensitivity can also be observed when solving Crossword puzzles or Scrambled words. When forming words with the given clues in the puzzle we often try the missing letter in an alphabetical order.

Colour recognition stands another example for order-sensitiveness. A child who is taught
colours in particular order say for example blue, green would correctly classify a true blue object and a true green object. But when an object with an intermediary colour is given he is biased to classify it as blue rather than green.

This clearly shows that human brain may be biased with order of learning things when recognizing ambiguous objects. This prompted us to the development of order-sensitive learning rule. It is emphasized here that order-sensitivity is one of the important features in human learning process and has not yet been explored in the context of artificial neural networks. Moreover, the natural instinct of human learning is to have longer impression of patterns that are learnt earlier.

Putting things more formally, let $X$ and $Y$ be two distinct training patterns. A learning rule is said to be order-sensitive if the network behaves differently for different order of presentation of the training patterns. In the present context of Hopfield network, it is identified that the learning is order-sensitive, if the learning rule generates different synaptic matrices when $X$ precedes $Y$ in the training compared to the matrix when $X$ follows $Y$.

The learning rule, proposed is truly an order-sensitive one in the sense that

i) the learning rule generates different synaptic matrices for the same set of training patterns presented in different order.

ii) the basins of attractions are biased towards earlier-learnt-pattern.

Let us consider the following example for illustration.

Let $P = 11101100001$, $Q = 11010011100$

Let $W_{PQ}$ be the synaptic matrix computed as per derivation given in Section 2.6.3 considering $P — X$, $Q — Y$ and $W_{QP}$ is synaptic matrix obtained by $P — Y$, $Q — X$. 
In other words, $W_{PQ}$ corresponds to training of the network when $P$ is presented first, followed by $Q$ and $W_{QP}$ corresponds to the training when the patterns are presented in the reverse order. The synaptic matrix $W_{PQ}$ and $W_{QP}$ are given below.

$$W_{PQ} = \begin{bmatrix}
1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -n^3 & 0.5 \\
0.5 & 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -n^3 & 0.5 \\
0.5 & 0.5 & 1.0 & -n^3 & 0.5 & 0.5 & -n^3 & -n^3 & -n^3 & 0.5 \\
0.5 & 0.5 & -n^3 & 2.0 & -n^3 & -n^3 & 1.5 & 1.5 & 1.5 & -n^3 & -n^3 \\
0.5 & 0.5 & -n^3 & 1.0 & 0.5 & -n^3 & -n^3 & -n^3 & -n^3 & 0.5 \\
\end{bmatrix}$$

$$W_{QP} = \begin{bmatrix}
0.5 & 0.5 & 0.5 & -n^3 & 0.5 & 1.0 & -n^3 & -n^3 & -n^3 & -n^3 & 0.5 \\
0.5 & 0.5 & -n^3 & 1.5 & -n^3 & -n^3 & 2.0 & 1.5 & 1.5 & -n^3 & -n^3 \\
0.5 & 0.5 & -n^3 & 1.5 & -n^3 & -n^3 & 1.5 & 2.0 & 1.5 & -n^3 & -n^3 \\
0.5 & 0.5 & -n^3 & 1.5 & -n^3 & -n^3 & 1.5 & 1.5 & 1.5 & -n^3 & -n^3 \\
-n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 \\
0.5 & 0.5 & 0.5 & -n^3 & 0.5 & 0.5 & -n^3 & -n^3 & -n^3 & -n^3 & 1.0 \\
\end{bmatrix}$$
Thus, the learning rule generates different weight matrices and hence yields different behavior of the network. Let us take the input pattern 11000000000 as initial state, the Hopfield network with $W_{PQ}$ as the synaptic weights, converges to $P$ as the stable state. Whereas if the same is presented to the network with $W_{QP}$ as synaptic matrix the stable state to which it converges is $Q$. Thus we observe that when 11000000000 is the initial state, the network is biased towards the pattern which it learnt earlier. And also this input state has equi-Hamming-distances from $P$ and $Q$ i.e. 6. As explained above in case of Hopfield network with synaptic matrix $W_{PQ}$ the output state is $P$ because it is the earlier-learnt-pattern. Similarly in case of $W_{QP}$ the output state is $Q$.

Let $P, Q$ be two user-specified stable states and $W_{PQ}$ be the synaptic matrix corresponding to the training of the network when $P$ is presented first, followed by $Q$ and $W_{QP}$ represents the synaptic matrix when the training patterns are presented in the reverse order. Let $Z$ be the input state which is equidistant from $P$ and $Q$. The learning rule is said to be order-sensitive when $Z$ converges to $P$ in case of the network $W_{PQ}$ and to $Q$ in case of the network $W_{QP}$.

Let $P = (p_1, p_2, \cdots, p_n)$ and $G(P) = \{i : p_i = 1\}$. Define

\[
S_1 = G(P) \cap G(Q), \quad |S_1| = s_1 \\
S_2 = G(P) \setminus G(Q), \quad |S_2| = s_2 \\
S_3 = G(Q) \setminus G(P), \quad |S_3| = s_3
\]

It may be noted that $S_1$ is also $G(P) \cap G(Q) \cap G(Z)$.

\[
K_1 = \{i : i \notin G(P) \land i \notin G(Q) \land i \notin G(Z)\}, \quad |K_1| = k_1 \\
K_2 = \{i : i \notin G(P) \land i \notin G(Q) \land i \notin G(Z)\}, \quad |K_2| = k_2 \\
K_3 = \{i : i \notin G(P) \land i \notin G(Q) \land i \notin G(Z)\}, \quad |K_3| = k_3 \\
K_4 = \{i : i \notin G(P) \land i \notin G(Q) \land i \notin G(Z)\}, \quad |K_4| = k_4 \\
K_5 = \text{otherwise}
\]
Theorem: The learning rule given in Section 2.6 is order-sensitive in the following sense.

1. For \( W_{PQ} \), \( Z \) converges to \( P \) if it is nearer to \( P \) or to \( Q \). For \( W_{QP} \), \( Z \) converges to \( Q \) if it is nearer to \( Q \).

2. If \( Z \) is equidistant from \( P \) and \( Q \), for \( W_{PQ} \), \( Z \) converges to \( P \) when \( k_2 > k_3 \) or \( \frac{s_1 + 1}{2} \geq k_3 \).

Proof: The case 1 is already proved in Section 2.7.

The proof for case 2 is given as following.

\( Z \) is equi-Hamming distant from \( P \) and \( Q \) only when

\[ k_1 + k_3 = k_2 + k_4. \]

Using the update rule (Eq. 2.2.3), \( Z^k \) is the state obtained from \( Z \) in one transition. If \( Z^k \) is nearer to \( P \) than to \( Q \), then by the earlier analysis (Section 2.7.1) \( Z^k \) converges to \( P \), else \( Z^k \) converges to \( Q \). So it is sufficient to check only one change of transition from \( Z \). This can be studied by identifying the value of \( k \) such that \( k^{th} \) bit is updated according to the update rule. In case of the network \( W_{PQ} \), the values \( \Delta E^i = E_{Z^i} - E_Z \) for \( i \in K_1, K_2, K_3 \) and \( K_4 \) are as follows (Section 2.7.1):

\begin{align*}
\Delta E^1 & = -\frac{1}{2} \left[ 1 + 2 \left( \frac{s_1}{2} + \frac{k_2}{2} + k_3(-n^3) \right) \right] - 0.5, \quad i \in K_1, \\
\Delta E^2 & = \frac{1}{2} \left[ 1 + 2 \left( \frac{s_1}{2} + \frac{k_2}{2} + k_3(-n^3) \right) \right] + 0.5, \quad i \in K_2, \\
\Delta E^3 & = \frac{1}{2} \left[ 1 - s_3 + \frac{s_2}{s_3} + 2 \left( \frac{s_1 s_2}{s_3^2} + k_2(-n^3) + (k_3 - 1) \left[ 1 + \frac{s_2(s_2 - 1)}{s_3(s_3 - 1)} \right] \right) \right] + 0.5, \quad i \in K_3, \\
\Delta E^4 & = -\frac{1}{2} \left[ 1 - s_3 + \frac{s_2}{s_3} + 2 \left( \frac{s_1 s_2}{s_3^2} + k_2(-n^3) + (k_3)[1 + \frac{s_2(s_2 - 1)}{s_3(s_3 - 1)}] \right) \right] - 0.5, \quad i \in K_4.
\end{align*}

So the bit \( k \) that is the candidate of update is

\[ \Delta E^k = \min \Delta E^i \quad (3.2.1) \]

Clearly, for \( i \in K_1 \) and \( i \in K_4 \), \( \Delta E^i > 0 \) and for \( i \in K_3 \) and \( i \in K_2 \), \( \Delta E^i < 0 \).
Since the dominating factors in the above equations for \( i \in K_3 \) and \( i \in K_2 \) are \(-k_3 n^3\) and \(-k_2 n^3\) respectively, so \( \Delta E'_i(i \in K_3) < \Delta E'_i(i \in K_2) \) only when \( fc_2 > k_3 \). So the bit \( k \) that is subjected to change in \( Z \) is in \( K_3 \), which results in moving \( Z^k \) nearer to \( P \) than to \( Q \), and hence the \( Z^k \) converges to \( P \). So it is proved that in case of the network \( WPQ \) a equi-Hamming distance pattern \( Z \) is converged to \( P \) when \( fc_2 > k_3 \).

Similarly, when \( k_2 = k_3 \), then \( s_2 = 53 \) and \( k_1 = fc_4 \), because \( k_1 + k_3 = k_2 + k_4 \). Then it is clear from the above equations that \( \Delta E'_i(i \in K_3) < \Delta E'_i(i \in K_2) \) only when \( \frac{s_2+1}{2} > k_3 \). So the bit \( k \) that is subjected to change in \( Z \) is in \( K_3 \), which results in moving \( Z^k \) nearer to \( P \) than to \( Q \), and hence the \( Z^k \) converges to \( P \). So it is proved that in case of the network \( WPQ \) a equi-Hamming distance pattern \( Z \) is converged to \( P \).

Similarly in case of \( WQP \), it can be shown that a equi-Hamming distance pattern \( Z \) is going nearer to \( Q \) in the next transition, when \( k \in K_2 \). In order to show this the synaptic matrix \( WQP \) is used and is given as follows:

\[
\begin{align*}
W_{ii} &= \begin{cases} 
1 & \text{if } i \in G(P), \\
1 - s_2 + s_3/s_2 & \text{if } i \in S_2, \\
-n^3 & \text{Otherwise}
\end{cases}
\end{align*}
\]

and the off-diagonal elements \( W_{ij}, \ i \neq j \) are given by

\[
\begin{align*}
W_{ij} &= \begin{cases} 
1/2 & \text{if } i, j \in G(P), \\
\frac{s_3}{2s_2} & \text{if } i \in S_1, j \in S_2; \text{ or } i \in S_2, j \in S_1, \\
1 + \frac{s_3(s_3-1)}{2s_2(s_2-1)} & \text{if } i, j \in S_2, \\
-n^3 & \text{otherwise}
\end{cases}
\end{align*}
\]
3.3 Experimental Results

While discussing the realistic examples of order-sensitive learning, color recognition is considered as an example. That is, a child who is taught colors in particular order say for example blue, green would correctly classify a true blue object and a true green object. But when an object with an intermediary color is given, the child is biased to classify it as blue (first learnt) rather than green.

Basing on the above phenomenon an experiment is conducted as a practical application to the proposed order-sensitive learning rule. The aim of the experiment is the following: If the two user-specified stable states (P and Q) are represented by two different colors and the input state (Z) which is at equi-Hamming distance from the two stable states is represented by a third color then there will be an ambiguity over to which of the first two colors the third color is closer. Then the order-sensitive learning will resolve this ambiguity by matching the third color to the earlier of the first two.

The steps involved in the experiment are: 1. Selecting two patterns randomly that represent P and Q. 2. Mapping the P and Q (binary numbers) to colors. 3. Generating all the patterns that are at the same hamming distance from the selected two patterns and mapping them to colors. 4. Selecting a color from the colors generated in the step-3 which satisfies our aim (ambiguity over to which of the first two colors the third color is closer).

Mapping of binary pattern to color is achieved by converting binary patterns into RGB values. That is dividing the pattern in to three parts, first part will denote the RED value, second part will denote the BLUE value and the last part will show the GREEN value. The colors are generated by using the "PAINT BRUSH" in MS-Windows. The results are depicted in Figure 3.1.
Figure 3.1 Colour Experiment
Hence the order-sensitive property of the learning rule is quite useful in automating the resolving of ambiguity in matching a color which is equally closer to two colors. For this, the experiment is to be conducted in a reverse fashion, that is color is to be mapped to binary patterns and order-sensitive learning rule has to be applied.

3.4 Relative Pruning

A major problem in the applications of neural networks is the choice of a topology. A considerable amount of architectures and methods for the construction of optimal neural networks have been proposed in the literature. Some of these approaches try to improve well-known architectures by training a network which is expected to be big enough to solve the given problem and subsequently remove its units (prune units). The topology of DNN is well organized such that the nodes that are not currently participating in the training can be pruned and reused for parallel processing of the next series of data. This pruning and parallel reusing of nodes will improve the efficiency of the network.

Basing on another biological phenomena a new notion called *relative pruning* is introduced in the network. Consider the biological situation wherein, a person being trained to drive a vehicle. In the initial stages of this training, all the units or the neurons in a particular layer are engaged in the processing of information (controlling this phenomenon). After a certain stage, when the trainee has achieved certain efficiency, some of the neurons can be relaxed for this activity. Processing in a similar fashion, we arrive at a situation that some of the neurons or units may be assigned relatively a different job. Thus as far as this information processing is concerned one may think, that these neurons have undergone a sort of *relative pruning*.

This biological phenomena is a motivation for our architecture. In our architecture too, a
concept some what similar to the above, gives a scope for relative pruning. In processing a particular task some of the neurons may be assigned relatively a different job and as and when required these neurons may be brought back to this particular task. One may even observe a sort of periodic phenomena as far as the sharing of information among neurons is concerned. Thus by looking at this aspect a new word namely, ‘relative pruning’ is introduced.

Relative pruning: Relative engagement of neurons for a particular task in a network is called Relative pruning.

Relative pruning is different from the standard notation of pruning of the network [Karnin 90; Reed 93; Siettma 91; Thimm 95] which aims at removing the neurons or weights that are not participating in training and without loss of generality of the training algorithm. On the other hand, in our proposed network, the network gets relatively pruned to the extent that half the number of basic nodes are relieved in each iteration and this finally results in leaving one node in the network. The advantage of this relative pruning is that the network structure employs the hardware most optimally when it is implemented and the relatively pruned nodes can be reused for parallel processing of the next series of data.

To further emphasize the differences between pruning and relative pruning, the concept of pruning is explained first. Pruning methods optimize both size and the generalization capabilities of neural networks. Most of this work is related to feed forward networks. Pruning is done as network trimming within the assumed initial architecture. The network is trimmed by removal of unimportant weights. This can be accomplished by estimating the sensitivity of the total error to the exclusion of each weight in the network [Karnin 90]. The weights which are insensitive to error changes can be discarded after
each step of incremental training. Unimportant neurons can also be removed [Siettma 91]. In either case it is advisable to retain the network with the modified architecture. The trimmed network is of smaller size and is likely to be trained faster than before its trimming.

The concepts of pruning in general and relative pruning in particular context are explained above. However for sake of completeness the process of relative pruning is presented as the following.

DNN has a composite structure consisting of several basic nodes which are similar to each other. It is assumed in the present work that all basic nodes memorize same number of patterns (say p). As explained in Chapter 2 we reemphasize that when each basic node memorizes p patterns then p basic nodes are grouped together. Within a group of basic nodes designate one of them as the leader of the group. For simplicity consider the first node as the leader. After the nodes in a group reach the respective stable states these nodes transmit their stable states to the leader in that group. At this stage DNN adopts a relative pruning strategy and it retains only the leader of each group and ignores all other basic nodes within a group. In the next pass the DNN consists of lesser number of nodes, but the structure is retained. These leader nodes are treated as basic nodes and each of them is trained to memorize p patterns corresponding to p stable states of the member nodes of the group. These leader nodes are again grouped together taking p nodes at a time. This process is repeated till a single node remains.

Thus in one cycle, the nodes carry out state-transitions in parallel, keeping the weights unchanged and in the next cycle, the nodes communicate among themselves to change the weights. At this stage half of the network is pruned and the remaining half is available for the next iteration of two cycles. In this process, the network eventually converges to
the closest pattern for any given pattern. The Hopfield nodes are connected to each other in a fashion similar to interconnection network and taking advantage of such structure, the active communication between subsets of nodes is determined and unnecessary nodes are pruned. If one can make use of the pruned nodes in the subsequent process, similar to the above mentioned biological phenomenon, the present pruning process becomes a relative pruning. Thus reiterating, one can use the pruned nodes to relatively a different assignment in case of this architecture.

3.5 Conclusions

In this chapter the major characteristics of DNN are introduced. The concept of order-sensitivity is explained with some live examples from children psychology and how this important concept achieved in DNN is explained. The notion of relative pruning and the difference between the concepts of pruning and relative pruning are explained. The need of relative pruning is strengthened with examples from biology. Thus this chapter adds strength to the need of introducing a new architecture.