# Probabilistic Memory Capacity of Recurrent Neural Networks

Seiji MIYOSHI* and Kenji NAKAYAMA†
*Graduate School of Natural Science and Technology, Kanazawa Univ.
†Department of Electrical and Computer Eng. Faculty of Eng., Kanazawa Univ.
*†2-40-20, Kodatsuno, Kanazawa 920, Japan
*E-mail miyoshi@ddecl.kobe-kosen.ac.jp

*ABSTRACT*

In this paper, probabilistic memory capacity of recurrent neural networks(RNNs) is investigated. This probabilistic capacity is determined uniquely if the network architecture and the number of patterns to be memorized are fixed. It is independent from a learning method and the network dynamics. It provides the upper bound of the memory capacity by any learning algorithms in memorizing random patterns. It is assumed that the network consists of N units, which take two states. Thus, the total number of patterns is the Nth power of 2. The probabilities are obtained by discriminations whether the connection weights, which can store random M patterns at equilibrium states, exist or not. A theoretical way for this purpose is derived, and actual calculation is executed by the Monte Carlo method. The probabilistic memory capacity is very important in applying the RNNs to real fields, and in evaluating goodness of learning algorithms. As an example of a learning algorithm, the improved error correction learning is investigated, and its convergence probabilities are compared with the upper bound. A linear programming method can be effectively applied to this numerical analysis.

## 1. Introduction

Many studies have been done about memory capacity of recurrent neural networks(RNNs)[1],[2]. Especially, in the case of Hopfield network, memory capacity is estimated about $0.15N$ by computer simulations[1], where $N$ is the number of units. The connection weights are decided by the outer product rule in the Hopfield network. Various theoretical investigations on the capacity have been done. However, the maximum memory capacity, which doesn't depend on learning methods, has not been well discussed.

On the other hand, the ability of linear threshold elements to classify patterns was studied by Cover[3]. However, this study dealt with the patterns whose components take continuous values, and a set of patterns were considered to be in $\phi$-general position, where every $N$ element subset of the vectors is linearly independent in $N$-dimensional space. When the units take two states, the patterns locate at the vertices of an $N$-dimensional hypercube. So, the patterns cannot be considered to be in $\phi$-general position.

In our approach, the memory capacity of RNNs is evaluated by the probability that the given number of the random patterns can be memorized at equilibrium states.

In Sec.2, the network equations of an RNN are shown. In Sec.3, a concept of probabilistic memory capacity is described. Furthermore, in order to obtain the probability, a method to discriminate whether the connection weights exist or not is proposed. In Sec.4, numerical analysis results of the probabilities with respect to the number of units and the number of patterns are shown. In Sec.5, as an example of a learning method, the improved error correction learning[4] is investigated, and its convergence probabilities are compared with the above probabilities. In Sec.6, the proposed method is modified so as to apply a linear programming method to a numerical calculation process.

## 2. Recurrent Neural Network

Let $u_i(n)$ and $v_i(n)$ be the output and the internal state of the $i$th unit, respectively. Furthermore, the connection weight from the $i$th unit to the $j$th unit is denoted $w_{ji}$. The network equations are given by

$$v_j(n) = \sum_{i=1}^{N} w_{ji} u_i(n) \ , \ w_{ii} = 0 \tag{1}$$

$$u_j(n+1) = \begin{cases} +1, & v_j(n) \geq 0 \\ -1, & v_j(n) < 0 \end{cases} \tag{2}$$

$N$ is the number of the units. Patterns memorized by the RNN are $N$-dimensional vectors. Memorized patterns are selected at random from all possible $2^N$ patterns. Self-feedbacks are not used, that is $w_{ii} = 0$. If this condition is not imposed, it would be able to store all the $2^N$ patterns by making up all self-feedbacks to positive and all the other weights to zero. Therefore, $w_{ii} = 0$ is quite natural assumption for the purpose of this paper.

## 3. Probabilistic Memory Capacity

### 3.1. Definition

The memory capacity usually depends on a set of patterns. For example, a set of patterns $\mathbf{p}_1 = (+1, +1, +1, \cdots, +1, +1, +1)$ and $\mathbf{p}_2 = (+1, +1, +1, \cdots, +1, +1, -1)$ cannot be memorized by an RNN. The reason is the following: In both patterns, the input potential of the $N$th unit is given by

$$\begin{aligned} \sum_{i=1}^{N} w_{Ni} u_i &= \sum_{i=1}^{N-1} w_{Ni} u_i + w_{NN} u_N \\ &= \sum_{i=1}^{N-1} w_{Ni} u_i \\ &= \sum_{i=1}^{N-1} w_{Ni} \end{aligned} \tag{3}$$

This value must be positive for $\mathbf{p}_1$ because the output of the $N$th unit is $+1$, and must be negative for $\mathbf{p}_2$ because the output of the $N$th unit is -1. These two requirements contradict to each other. Therefore, this set cannot be memorized.

As described above, the connection weights don't always exist. The possibility is highly dependent on the pattern combinations.

There are ${}_{2^N}C_M$ combinations in the case of selecting $M$ patterns from all possible $2^N$ patterns. When $a\%$ of ${}_{2^N}C_M$ combinations can be memorized, the probabilistic memory capacity of this RNN for $M$ patterns is defined to be $a\%$. This capacity is determined uniquely if the network architecture and the number of patterns to be memorized are fixed. It is independent from a learning method and the network dynamics. It provides the upper bound of the memory capacity by any learning algorithms in memorizing random patterns.

### 3.2. Existence of Connection Weights

$M$ patterns to be memorized are selected from $2^N$ patterns, and are expressed by

$$\mathbf{p}_k = (p_{k1}, p_{k2}, \cdots, p_{kN}) \tag{4}$$

$$p_{kj} = +1 \text{ or } -1$$

$$1 \leq k \leq M$$

In order to obtain the probabilistic memory capacity, it is necessary to discriminate whether a set of $M$ patterns can be memorized or not. In other words, this discrimination can be said the discrimination whether $w_{ji}$, which satisfy the alignment condition Eqs.(5)-(7) for $M$ patterns, can be obtained or not. Thus, the discrimination is equivalent to the check of "the existence of connection weights".

$$\sum_{i=1}^{N} w_{ji} p_{ki} \geq 0 \quad for \quad p_{kj} = +1 \tag{5}$$

$$\sum_{i=1}^{N} w_{ji} p_{ki} < 0 \quad for \quad p_{kj} = -1 \tag{6}$$

$$1 \leq k \leq M \tag{7}$$

### 3.3. Conditions on Connection Weights

The condition, under which the connection weights exist, can be represented geometrically as follows: Now, whether connection weights exist or not concerning the $N$th unit is considered. The $M$ vectors composed of the remaining ($N$-1) units represented by the following expression correspond to $M$ vertices of the ($N$-1)-dimensional hypercube.

$$\mathbf{p}'_k = (p_{k1}, p_{k2}, \cdots, p_{k,N-1}) \ , \ 1 \leq k \leq M \tag{8}$$

The whole patterns are given by

$$\mathbf{p}_k = (\mathbf{p}'_k, p_{kN}) \ , \ p_{kN} = \pm 1 \tag{9}$$

In order to memorize $\mathbf{p}_k$, the following conditions are required.

$$w\mathbf{p}_k'^{\,t} \geq 0 \quad for \quad p_{kN} = +1 \tag{10}$$

$$w\mathbf{p}_k'^{\,t} < 0 \quad for \quad p_{kN} = -1 \tag{11}$$

These conditions are expressed in other words like the following. If the vertices, where the value of the $N$th unit is +1, and other vertices, where the value of the $N$th unit is -1, can be divided by a hyperplane through the origin point, then the connection weights exist. The normal vector of the hyperplane can be used as the connection weights. The reason, why the hyperplane is through the origin point, is that the bias term is assumed to be zero in Eq.(1). If such hyperplanes exist for all units, the connection weights of the RNN memorizing the $M$ patterns exist.

Figure 1 shows an example of the relation between the vertices and the hyperplane. In this example, $N$=4 , $M$=5 and the connection weights for the 4th unit exist. Memorized patterns are $A = (1,-1,1,-1)$ , $B = (-1,-1,-1,-1)$ , $C = (-1,1,1,-1)$ , $D = (1,1,1,1)$ , $E = (1,1,-1,1)$. The values of the 4th units of A, B and C are -1. So, they are represented by black vertices. On the other hand, the values of the 4th units of D and E are +1. So, they are represented by white vertices. In this case, three black vertices and two white vertices can be divided by the hyperplane through the origin point as shown in Fig.1. This means that the connection weights for the 4th unit exist. Then the components of the normal vector of the hyperplane can be used as the connection weights.

So, it is necessary and sufficient to discriminate whether the hyperplanes described above for all units and all given patterns are exist or not.

## 3.4. Discrimination Whether Connection Weights Exist or Not

The normal vector of the hyperplane, corresponding to the connection weights, is represented by $\mathbf{z}$ in this paper. $\mathbf{z}$ is an ($N$-1)-dimensional vector.

$$\mathbf{z} = (z_1, z_2, \cdots, z_{N-1}) \tag{12}$$

If the connection weights exist, the vertices can be divided by a hyperplane through the origin point. So, $\mathbf{z}$ corresponding to the connection weights satisfies the following conditions[5].

$$\mathbf{p}_k'\mathbf{z}^t = \sum_{j=1}^{N-1} p_{kj} z_j \geq 0 \quad if \quad p_{kN} = +1 \tag{13}$$

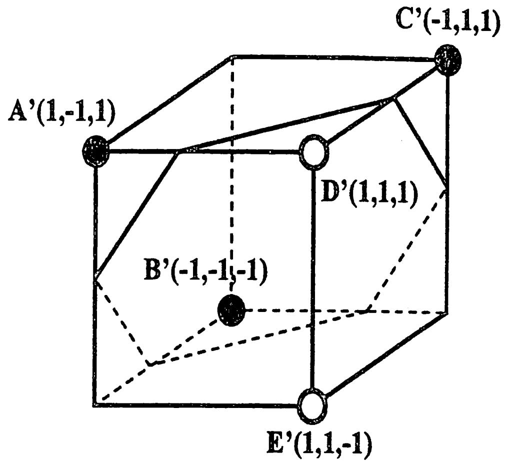

Fig. 1: Hyperplane which can divide vertices corresponding to patterns (N=4 , M=5).

$$\mathbf{p}_k'\mathbf{z}^t = \sum_{j=1}^{N-1} p_{kj} z_j < 0 \quad if \quad p_{kN} = -1 \tag{14}$$

An example of normal vector is shown in Fig.2, where $N$=4 , $M$=7 and the connection weights for the 4th unit exist. The black vectors correspond to the patterns in which the values of the 4th units are -1. On the other hand, the white vectors correspond to the patterns in which the values of the 4th units are +1. In this case, three black vectors and four white vectors can be divided by the plane through the origin point. The normal vector of the hyperplane is $\mathbf{z}$.

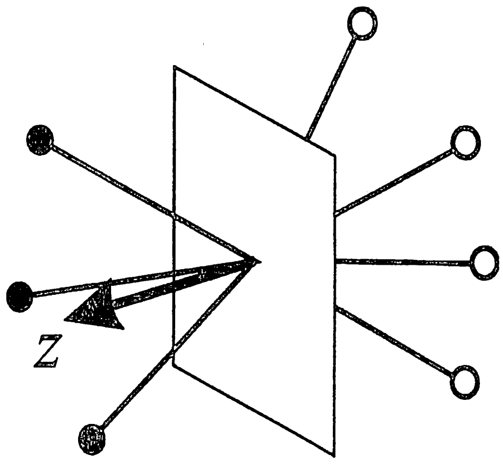

Fig. 2: Normal vector of hyperplane which divides vertices

As the components of patterns are restricted to be +1 or −1, it isn't necessary to consider the infinite number of the vectors for candidates of $\mathbf{z}$. From some geometric consideration, it is assumed that:

·When $N$ is less than 4, it is sufficient to consider about $3^{N-1}$ vectors in which each component is one of {-1,0,+1}.

·When $N$ is not less than 4, it is sufficient to consider about $(2N-5)^{N-1}$ vectors in which each component is one of

$$\{\underbrace{-N+3,-N+4,\cdots,-1,0,+1,\cdots,N-4,N-3}_{2N-5}\} \tag{15}$$

So, it is able to know whether connection weights exist or not concerning the unit by using a computer.

The same way is effective for the other units. If the connection weights exist for all units, the given set of patterns is able to be memorized.

## 4. Numerical Analysis

If $N$ isn't quite small, it is difficult to examine all combinations of $M$ patterns which are selected from $2^N$ patterns. So, the Monte Carlo method is adopted. That is, the probabilities can be estimated by using a sufficient number of combinations of $M$ patterns selected at random from $2^N$ patterns. The method described in the previous section is used to discriminate whether connection weights exist or not, which satisfy Eqs.(5)-(7) for the given $M$ patterns. Table 1 shows the probabilities which are obtained about various combinations of $N$ and $M$. Each probability is calculated by using 1000 sets of $M$ patterns selected at random.

Table: 1: Probabilities(%) of the connection weights.

| $M$ / $N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 3 | 100.0 | 35.8 | 1.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 100.0 | 61.0 | 23.7 | 5.3 | 0.8 | 0.2 | 0.0 | 0.0 |
| 5 | 100.0 | 75.2 | 41.1 | 17.7 | 3.5 | 0.3 | 0.1 | 0.0 |
| 6 | 100.0 | 84.8 | 62.9 | 38.3 | 17.7 | 4.9 | 1.4 | 0.7 |
| 7 | 100.0 | 92.5 | 76.7 | 57.9 | 38.4 | 20.6 | 8.6 | 2.0 |

From this table, the memory capacity is estimated about $0.5N$ with 60% probability. These probabilities are interpreted as follows : When we develop an associative memory using the RNN, if we want to memorize $0.5N$ random patterns, the probability of success is about 60%. Of course, this table represents the probabilities about no more than extremely limited combinations of $N$ and $M$. Some features may appear clearly on the large table which is calculated about much more combinations of $N$ and $M$. This is a near future work concerning Sec.6.

## 5. Estimation of Goodness of Learning Methods

### 5.1. Improved Error Correction Learning

As an example of learning algorithms for memorizing patterns by a recurrent neural network, the improved error correction learning is taken into account[4].

In this learning algorithm, hysteresis threshold $\pm T$ and hysteresis margin $dT$ were introduced in order to improve recall performance from noisy patterns and stabilize a learning process, respectively[4]. This learning algorithm is shown in the following.

I. Initialize connection weights.

$$w_{ji}(0)=0\ ,\ 1\leq i,j\leq N \tag{16}$$

II. Calculate network state.
The weighted sum input of all units are calculated after setting the network state to a certain pattern to be memorized. Letting $p_i(n)$ be the state, that is the output, of the $i$th unit for the given pattern, the weighted sum input $v_j(n)$is given by

$$u_i(n) = p_i(n) \tag{17}$$

$$v_j(n) = \sum_{i=1}^{N} w_{ji}u_i(n) \tag{18}$$

III. Calculate correction of weights.

When $p_j(n) = +1$,
if $v_j(n) \geq T$, then

$$\Delta w_{ji}(n) = 0 \tag{19}$$

if $v_j(n) < T$, then

$$\Delta w_{ji}(n) = \frac{T + dT - v_j(n)}{N-1}p_i(n) \tag{20}$$

When $p_j(n) = -1$,
if $v_j(n) \leq -T$, then

$$\Delta w_{ji}(n) = 0 \tag{21}$$

if $v_j(n) > -T$, then

$$\Delta w_{ji}(n) = \frac{-T - dT - v_j(n)}{N-1}p_i(n) \tag{22}$$

$dT$ is a margin for the threshold level, which can avoid unstable behavior in an error correction learning process and can accelerate the learning speed [4].

IV. Update connection weights.
Using $\Delta w_{ji}(n)$ calculated in III, all connection weights are adjusted at the same time by

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{23}$$

V. II-IV are repeated until $\Delta w_{ji}(n) = 0$ for all patterns.

After convergence, the network always satisfies the followings for all the input patterns which are memorized.

$$if\ \ p_j = +1\ \ then\ \ v_j \geq +T \tag{24}$$

$$if\ \ p_j = -1\ \ then\ \ v_j \leq -T \tag{25}$$

## 5.2. Estimation of Goodness of Error Correction Learning

Table 2 shows the simulation results of convergence probabilities of the improved error correction learning algorithm for memorizing random patterns. Each probability is calculated by using the same 1000 sets as the calculations of Table 1.

Table: 2: Convergence probability(%) of the improved error correction learning. $dT/T$=0.1

| $M$ / $N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 3 | 100.0 | 13.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 100.0 | 47.8 | 8.5 | 1.0 | 0.2 | 0.0 | 0.0 | 0.0 |
| 5 | 100.0 | 67.1 | 30.3 | 8.1 | 1.6 | 0.1 | 0.1 | 0.0 |
| 6 | 100.0 | 79.4 | 54.0 | 27.9 | 10.1 | 2.9 | 0.7 | 0.5 |
| 7 | 100.0 | 90.8 | 69.5 | 49.0 | 27.9 | 12.5 | 4.2 | 0.9 |

Comparing Table 1 and Table 2, as a matter of course, the convergence probabilities by the improved error correction learning are lower than or equal to the upper bound. The lowness is caused by the fact that the improved error correction learning cannot memorize the patterns using the condition of '=' in '$v_j(n) \geq 0$' about Eq.(2) as far as using positive $T$. It is necessary to use the condition of '=' when more than ($N$-2) vertices are in a hyperplane through the origin point. In fact, removing the condition of '=' in '$v_j(n) \geq 0$' about Eq.(2), all the probabilities of the connection weights have agreed with Table 2 perfectly. In other words, the improved error correction learning using positive $T$ can find the connection weights as far as more than ($N$-2) vertices aren't in a hyperplane through the origin point. Of course, if zero $T$ is used in Eq.(20), it may be possible to find the connection weights about such vertices. However, considering actual calculation, infinitesimal numerical error may make it difficult to converge. So, in this paper, zero $T$ isn't considered and it is considered that the improved error correction learning cannot find the connection weights about such vertices.

The ratios of the both Tables are shown in Table 3. In the low probability cases, the statistical condition may not be satisfied and the errors are assumed to be large. So, these cases are removed and expressed by "–" in Table 3. The larger the number of memorized patterns is, the lower the ratio is.

Table: 3: The ratios of the convergence probabilities of the improved error correction learning and the upper bound.

| $M$ / $N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 3 | 100.0 | 38.2 | - | - | - | - | - | - |
| 4 | 100.0 | 78.4 | 35.9 | 18.9 | - | - | - | - |
| 5 | 100.0 | 89.2 | 73.7 | 45.8 | 45.7 | - | - | - |
| 6 | 100.0 | 93.6 | 85.9 | 72.8 | 57.1 | 59.2 | - | - |
| 7 | 100.0 | 98.2 | 90.6 | 84.6 | 72.7 | 60.7 | 48.8 | - |

## 6. Linear Programming Method

In Sec.3.4, we have proposed a method to discriminate whether the connection weights exist or not. However, the calculation amount of this method is very large. It has the order of $N^N$. There is another method to discriminate more efficiently.

Considering about the $N$th unit, Eqs.(5)-(7) can be rewritten as follows:

$$\sum_{i=1}^{N-1} w_{Ni} p_{ki} \geq 0 \quad for \quad p_{kN} = +1 \tag{26}$$

$$\sum_{i=1}^{N-1} w_{Ni} p_{ki} < 0 \quad for \quad p_{kN} = -1 \tag{27}$$

$$1 \leq k \leq M \tag{28}$$

We want to know whether the solution of these simultaneous inequalities exists or not. Eqs.(26),(27) can be rewritten as follows:

$$\sum_{i=1}^{N-1} w_{Ni}p_{ki} \geq 0 \quad for \quad p_{kN} = +1 \tag{29}$$

$$-\sum_{i=1}^{N-1} w_{Ni}p_{ki} > 0 \quad for \quad p_{kN} = -1 \tag{30}$$

Eqs.(30) can be rewritten as follows without losing generality.

$$-\sum_{i=1}^{N-1} w_{Ni}p_{ki} \geq d > 0 \quad for \quad p_{kN} = -1 \tag{31}$$

$w_{Ni}$ can be rewritten as follows by introducing non-negative variables $t_i$ and $s_i$.

$$w_{Ni} = t_i - s_i \tag{32}$$

Then Eqs.(29),(31) can be rewritten as follows.

$$\sum_{i=1}^{N-1} t_i p_{ki} - \sum_{i=1}^{N-1} s_i p_{ki} \geq 0 \quad for \quad p_{kN} = +1 \tag{33}$$

$$-\sum_{i=1}^{N-1} t_i p_{ki} + \sum_{i=1}^{N-1} s_i p_{ki} \geq d \quad for \quad p_{kN} = -1 \tag{34}$$

$$t_i \geq 0 \ , \ s_i \geq 0 \ , \ d > 0 \tag{35}$$

Now considering the variables $r_i$ and $q_{ki}$ like follows:

$$r_i = \begin{cases} t_i & (1 \leq i \leq N-1) \\ s_{i-N+1} & (N \leq i \leq 2N-2) \end{cases} \tag{36}$$

if $p_{kN} = +1$, then,

$$q_{ki} = \begin{cases} p_{ki} & (1 \leq i \leq N-1) \\ -p_{k,i-N+1} & (N \leq i \leq 2N-2) \end{cases} \tag{37}$$

if $p_{kN} = -1$, then,

$$q_{ki} = \begin{cases} -p_{ki} & (1 \leq i \leq N-1) \\ p_{k,i-N+1} & (N \leq i \leq 2N-2) \end{cases} \tag{38}$$

By introducing the above variables, Eqs.(33)-(35) can be rewritten as follows:

$$\sum_{i=1}^{2N-2} r_i q_{ki} \geq d_k \tag{39}$$

$$r_i \geq 0 \ , \ d_k \geq 0 \tag{40}$$

This is the standard form of the linear programming about $r_i$. So, it can be solved by the first procedure of the simplex method, that is introducing the slack variables, the artificial variables and the auxiliary objective function.

Applying this method, the calculation amount can be reduced remarkably. So, it is much easier to enlarge Table 1 than the case of the method described in Sec.3.4.

## 7. Conclusion

In this paper, probabilistic memory capacity of recurrent neural networks(RNNs) has been investigated. This probabilistic capacity is determined uniquely if the network architecture and the number of patterns to be memorized are fixed. It is independent from a learning method and the network dynamics. It provides the upper bound of the memory capacity by any learning algorithms in memorizing random patterns. The probabilities have been obtained by discriminations whether the connection weights, which can store random M patterns at equilibrium states, exist or not. A theoretical way for this purpose has been derived, and actual calculation has been executed by the Monte Carlo method. The probabilistic memory capacity is very important in applying the RNNs to real fields, and in evaluating goodness of learning algorithms. As an example of a learning algorithm, the improved error correction learning has been investigated, and its convergence probabilities has been compared with the upper bound. At last, it has been proposed to apply linear programming method in order to discriminate whether the connection weights exist or not more efficiently. The actual calculation by this new method is a near future work.

## References

[1] J.J.Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. of the National Academy of Sciences of the U.S.A.*, Vol.79, pp. 2554-2558, 1982.

[2] D.J.Amit, "Modeling Brain Function: The World of Attractor Neural Network," *Cambridge University Press*, 1989.

[3] T.M.Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. on Electronic Computers*, Vol.EC-14, pp. 326-334, 1965.

[4] K. Nakayama and K. Nishimura, "A delta rule algorithm using double hysteresis thresholds for recurrent associative memory," *Proc. ICNN'94, Florida*, pp. 1163-1168, June 1994.

[5] Gilbert Strang, "Linear Algebra and Its Applications," *Academic Press*, 1976.