A Implementation of Simple Convolution Decoder Using a Temporal Neural Networks

Hee-Tae Chung, Kyung-Hun Kim, *Member, KIMICS*

Abstract—Conventional multilayer feedforward artificial neural networks are very effective in dealing with spatial problems. To deal with problems with time dependency, some kinds of memory have to be built in the processing algorithm. In this paper we show how the newly proposed Serial Input Neuron (SIN) convolutional decoders can be derived. As an example, we derive the SIN decoder for rate code with constraint length 3. The SIN is tested in Gaussian channel and the results are compared to the results of the optimal Viterbi decoder. A SIN approach to decode convolutional codes is presented. No supervision is required. The decoder lends itself to pleasing implementations in hardware and processing codes with high speed in a time. However, the speed of the current circuits may set limits to the codes used. With increasing speeds of the circuits in the future, the proposed technique may become a tempting choice for decoding convolutional coding with long constraint lengths.

Index Terms—Communication, Information

I. INTRODUCTION

Data transmission over a noisy and possibly distorted channel leads usually to bit errors. To overcome this problem, error correction codes (ECC) is used. A group of such codes are the convolutional codes [1]. The convolutional encoding process is very simple and can be implemented using simple digital components. It is seen that using such an implementation a very fast decoder can be built. It is also noted that the proposed decoder fits very well for VLSI circuits.

The design of the error correcting codes is predicated upon the use of an algebraic code (the "generator polynomial" or "generator matrix") and some *a priori* assumptions on the channel characteristics. In most cases, these assumptions have been greatly simplified. Specifically, most codes and associated channel simulations assume the presence of additive white gaussian noise (AWGN) and/or binary symmetric channel (BSC) characteristics.

Manuscript received December 4, 2003.

H. T. Chung is with the Division of Digital Information Engineering, Pusan University of Foreign Studies, Uam-Dong, Nam-Gu, Pusan, 608-738, Korea(Tel : +82-51-640-3425, E-mail : htchung@pufs.ac.kr)

K. H. Kim is with the Graduate School of Electronic & Computer Engineering, Pusan University of Foreign Studies, Uam-Dong, Nam-Gu, Pusan, 608-738, Korea(Tel : +82-51-640-3425, E-mail : hunkk@orgio.net)

Conventional multilayer feedforward artificial neural networks are very effective in dealing with spatial problems. They are, however, not quite suitable in solving temporal problems with time dependency, thus some kinds of memory have to be built in the processing algorithm. A popular approach is to introduce memory with time delays so that the temporal problem can be solved spatially. A temporal artificial neural computing architecture is proposed based on the SIN by Huang and Chiang[3],[4]. They designed two the SIN-based model reference adaptive controllers for a class of non-autonomous systems. Good output tracking performance of the on-line learning controller implies that the SIN must have some temporal signal processing ability. We would like to investigate this ability of the SIN.

To deal with problems with time dependency using artificial neural computing architecture, some kinds of dynamic properties should be provided. According to Elman [5], the artificial neural network is dynamic if it has some memory. A common approach to introduce memory is to use time delays so that the temporal sequence can be buffered to form a spatial array and a static mapping is then applicable to perform appropriate processing.

Another way to introduce dynamic behavior is to feedback appropriate information so that the representation of time dependency is implicit. The network proposed by Elman uses a buffer so that the output of the hidden layer at time step interacts with the input signal at time step. The feedback connections via the buffer enable Elman networks to learn to process temporal as well as spatial patterns.

In Huang and Chiang[3], [4], the SIN was proposed for constructing a universal function approximator defined in the real line. In this paper, we would like to investigate this ability of the SIN. Since the SIN contains a feedback loop, some memory effect can be provided; therefore, temporal signals can be processed. For most conventional neural networks, network parameters (i.e. weights and biases) are updated in the training phase, and are fixed during operation. But, for the proposed network, its parameters may have different values for different time instants during the operation phase. Therefore, we have to determine time trajectories of these parameters before its actual operation. A direct computation method will be introduced so that a set of feasible parameters in the network parameter space can be obtained. In here, it has been demonstrated by simulations that the SIN decoder works well compared with Viterbi algorithm. Because the SIN itself is very simple compared with Viterbi algorithm, it seems possible to implement it on a fast chip.

II. BASIC OPERATION OF THE SIN

In the Elman network, a buffer is designed to store the current hidden layer output so that the output can interact with next input. In this way, temporal information of the input sequence can be processed. Instead of adding a buffer, we allow the weight of the neuron to be replaced by the previous summer output so that this kind of feedback enables the neuron to process temporal sequences. The replacement of the weight makes the neuron behavior become dynamic. In order to have proper mapping performance, a sequence of biases has to be determined.

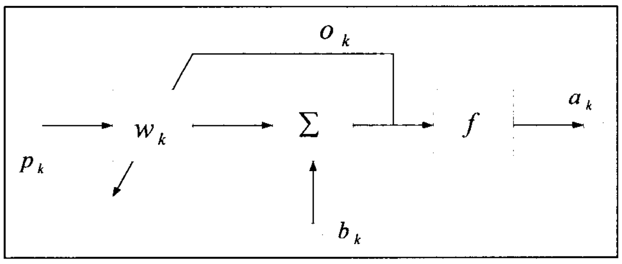


Fig. 1 Structure of the SIN.

Figure. 1 illustrates the structure of the SIN, where $p_k \in \mathrm{R}$, $\omega_k \in \mathrm{R}$ and $b_k \in \mathrm{R}$ are input, weight, and bias of the SIN at time step $k, k = 1, \cdots n$, respectively. The summation block denoted by $\sum$ performs a summation operation of its two inputs to generate signal o_k The operation can be expressed by

$$o_k = p_k \omega_k + b_k \tag{2.1}$$

The activation function f can be selected as conventional forms such as the hard limit function or a pure linear function.

The output of the SIN is denoted by $a_k \in \mathrm{R}$ which is calculated by $a_k = f(o_k)$. For most conventional artificial neurons, the weight and bias are updated based on some learning rules during its training phase, but fixed during operation phase. In order to introduce temporal processing ability into the new structure, we replace $\omega_{k+1} = 1, \cdots, n-1$ so that the weight at each time step will be fully affected by the last state. The initial weight is a value to be determined depending on the problem given. The sequence of biases $\{b_k\}, k = 1, \cdots, n$ is also to be found to enable the neuron to give proper output. Determination of the initial weight and the bias sequence can be done by direct computation based on the neuron operation or via learning.

Let us assume that the initial weight and bias sequence are determined, and the operation can be started with $k = 1$ to have

$$o_1 = \omega_1 p_1 + b_1 \tag{2.2}$$

Since we are only concerned with the output after the entire input sequence is presented, the values $\alpha_1, i = 1, \cdots, n-1$ are not computed. Before we proceed to the calculation of o_2 at $k = 2$, the weight ω_2 is replaced by o_1, and we have

$$o_2 = \omega_2 p_2 + b_2 = o_2 p_2 + b_2 \tag{2.3}$$

In the same manner, we may have the relation for $k = n$ as

$$o_n = \omega_n p_n + b_n = o_{n-1} p_n + b_n \tag{2.4}$$

And we may calculate the output of the neuron as

$$a_n = f(o_n) = f(o_{n-1} p_n + b_n) \tag{2.5}$$

For convenience of presentation, a simplified block in Fig. 2 is used to represent a serial input neuron.

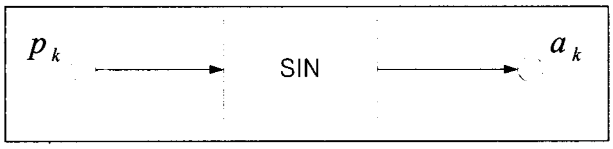


Fig. 2 Simplified the SIN representation.

III. SIN LEARNING

From the previous section, we know that before using the SIN on a given problem its initial weight, and a sequence of bias terms $b_j, j = 1, \cdots, n$ have to be found. Let us define $b_0 \equiv \omega_1$ so that the network parameters to be determined become $b_j, j = 0, \cdots, n-1$. In this section, a modified operation of the SIN is suggested to approximate real-valued functions defined on the real line.

From (2.2), the output of the SIN at the first time step is

$$o_1 = b_0 p_1 + b_1 \tag{3.1}$$

Likewise, the output of the second time step is

$$o_2 = o_1 p_2 + b_1 = b_0 b_1 p_2 = b_1 p_2 + b_2 \tag{3.2}$$

The expression can be further written in a more compact form

$$o_n = \sum_{r=0}^{n} \left(b_r \prod_{s=o}^{n-r-1} p_{n-s} \right) \tag{3.3}$$

The actual output becomes $\alpha_n = f(o_n)$ where f is the activation function. In the application of the SIN on function approximation problems, the pure linear function can be selected as the activation function. This gives the SIN output

$$o_n = f\left(\sum_{r=0}^{n} \left(b_r \prod_{s=o}^{n-r-1} p_{n-s} \right) \right) \tag{3.4}$$

Now, assume that input $p_k, k = 1, \cdots, n$ are identical then define $x = p_k$ for all k. Let $c_k = b_{n-k}, k = 0, \cdots, n$. and (3.4) can be further written as

$$a_n = f\left(\sum_{r=0}^{n} c_r x_r\right) \tag{3.5}$$

Equation (3.5) indicates that the SIN output a_n is an th order polynomial with coefficients $c_r, r = 0, \cdots, n$. Since by the Stone-Weierstrass a continuous function on a cube in R^n can be uniformly approximated by a polynomial, the SIN structure can be used in function approximation applications.

Note that we have assumed $x = p_k$ for all $k = 1, \cdots, n$ which implies that with the presentation of a specific input, $n+1$ iterations in the SIN structure are performed to produce the output a_n according to (3.1) Iterations times is $o(n)$ where n is the order of the approximating polynomial which is specified by the user. Since computations in the SIN involve only additions and multiplications, it is very suitable to be implemented in current microcomputers or DSPs.

Convergence of the polynomial in (3.5) is determined by the convergence of the sequence constructed by its coefficients [6]. The famous Cauchy-Hadamard theorem indicates that the radius of convergence of the polynomial in (3.5) is the reciprocal of the limit value of the coefficient sequence if it is convergent; otherwise, it is the reciprocal of the greatest limit points of the coefficient sequence if it is bounded. If the polynomial is not convergence nor bounded, the radius of convergence is zero, and the polynomial converges only at a point, Since the coefficients of (3.5) is updated from the SIN learning, the convergence of the polynomial is guaranteed as long as learning converges.

The learning rule is derived by a transformation of (3.4) into some space where the perceptron rule is applicable. Define a new set of variables $z_n, k = 0, \cdots, n$ as

$$\begin{aligned} z_n &= 1, \\ z_r &= \prod_{s=0}^{n-r-1} p_{n-s}, \\ r &= 0,1,\cdots,n,-1. \end{aligned} \tag{3.6}$$

Then (3.4) can be further written in the form

$$a_n = f\left(\sum_{r=0}^{n} b_r z_r\right) = f(B^T Z) \tag{3.7}$$

It is obvious that (3.7) is identical to the mapping equation of a perceptron with multiple inputs and one output. The vector **B** is the weight vector and **Z** is the input vector. This implies that we may apply any learning rule available to the perceptron to the learning of the SIN after a transformation of its inputs using (3.6). In here we apply the principles of performance learning to a single-layer linear neural network. The Widrow-Hoff rule [7], [8] can be used as

$$\begin{aligned} B(k+1) &= B(k) + 2\gamma \\ &[a_{target}(k) - a_n(k)]z(k) \end{aligned} \tag{3.8}$$

where $\gamma \rangle 0$ is learning rate and $a_{target}(k)$ is target value of a_n at th learning iteration.

IV. ENCODER

Assume a general convolutional encoder for rate code with constraint length 3. The design of the ECC is predicated upon the use of an algebraic code (the "generator polynomial" or "generator matrix") and some a priori assumptions on the channel characteristics. In most cases, these assumptions have been greatly simplified. Specifically, most codes and associated channel simulations assume AWGN and/or binary symmetric channel (BSC) characteristics.

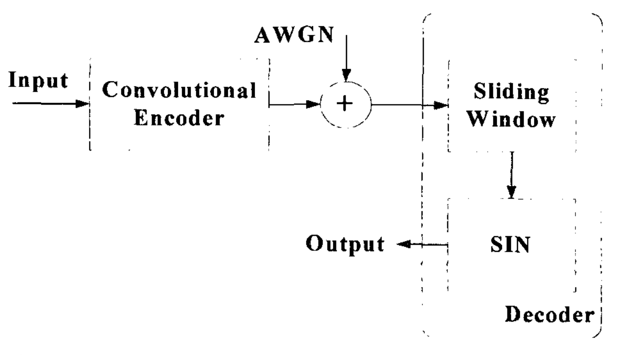


Fig. 3 Digital communication system with a the SIN ecoder.

Assume a general convolutional encoder for rate code with constraint length 3. The design of the ECC is predicated upon the use of an algebraic code (the "generator polynomial" or "generator matrix") and some a priori assumptions on the channel characteristics. In most cases, these assumptions have been greatly simplified. Specifically, most codes and associated channel simulations assume AWGN and/or binary symmetric channel (BSC) characteristics.

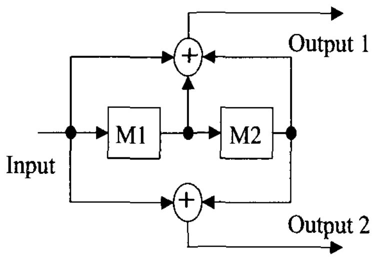


Fig. 4 Convolutional encoder 1/2 rate.

Figure 4 illustrates convolutional encoder with two memories for 1/2 rate. For example, from Fig. 4 where M1 and M2 are memories it is easily seen that the output is [1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 0 0] when the input is [1 1 0 1 0 1]. The other example from Fig. 5, the output is [1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 1 0] when the input is [1 1 0 1 0 1 0 1 1 1 0 1 1 0 1].

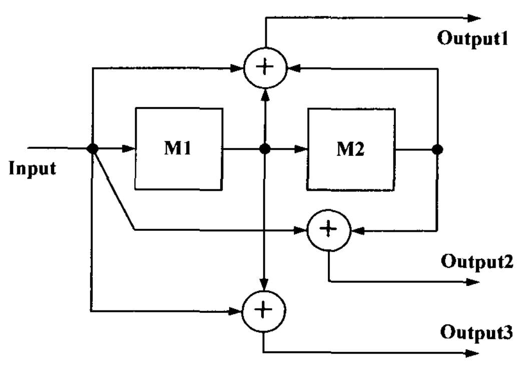


Fig. 5 Convolutional encoder 1/3 rate.

V. PROPOSED THE SIN DECODER

The transmitted signal with AWGN is reassigned to be an input of the SIN decoder with the sliding window in Fig. 3. The neural network processes the received code word sequentially as shown in Fig. 6, obtaining the input bit associated with the code word. Then, the sliding window is shifted by 2 bits position, including next two bits and discarding the previous two bits and then the next code word is processed. If input data passed through Fig. 5, the sliding window would be shifted by 3 bits position, including next three bits and discarding the previous three bits.

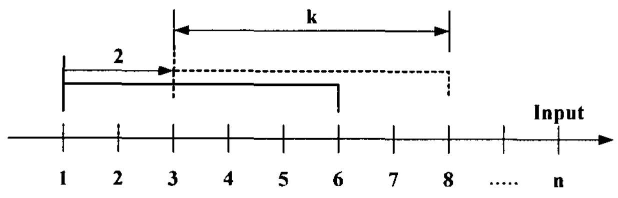


Fig. 6 Sliding windows for 1/2 rate.

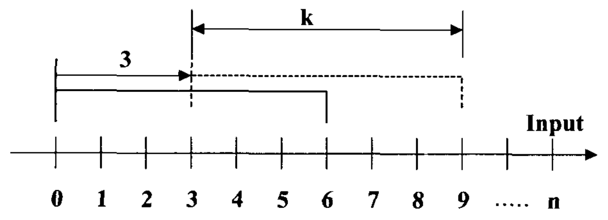


Fig. 7 Sliding windows for 1/3 rate.

In this manner, we obtain a new input form, and then use it as that of the SIN. With the previous example, a codeword will be composed as follows

$$\text{Input} = \begin{matrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{matrix}, \quad \text{Target} = \begin{matrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{matrix}$$

Figure 8 shows how to divide inputs with lines which are weights and biases. After transformed by (3.6), the new input will be,

$$\text{Input} = \begin{matrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{matrix}, \quad \text{Target} = \begin{matrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{matrix}$$

In Fig. 9, transformed inputs for 1/2 rate are divided by a single line, whereas in Fig. 8 three lines are required to divide original inputs. In the same manner, the other transformed inputs for 1/3 rate are also divided by a single line like Fig. 11, whereas in Fig. 10 five lines are required to do so. It means fewer iterations needed to get desired convergence.

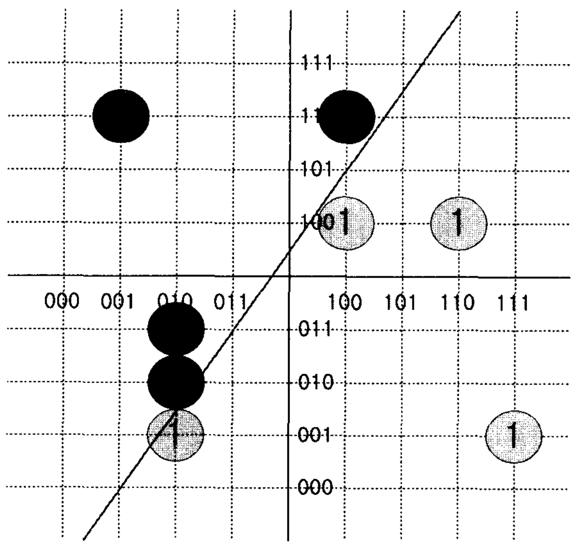


Fig. 8 Decision boundaries with multilayer perceptron for 1/2 rate.

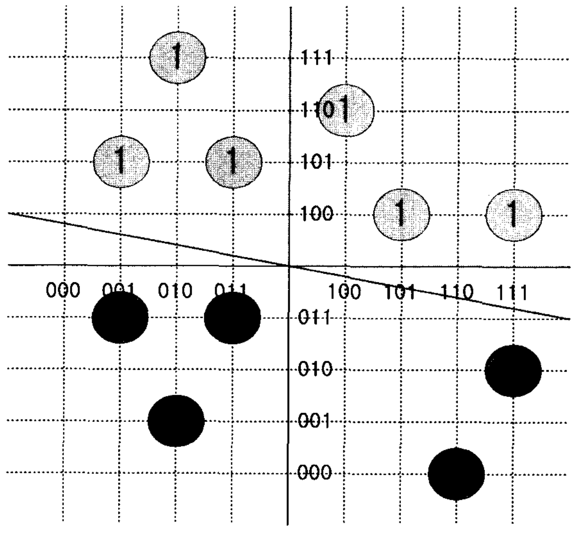


Fig. 9 Decision boundary with the SIN for 1/2 rate.

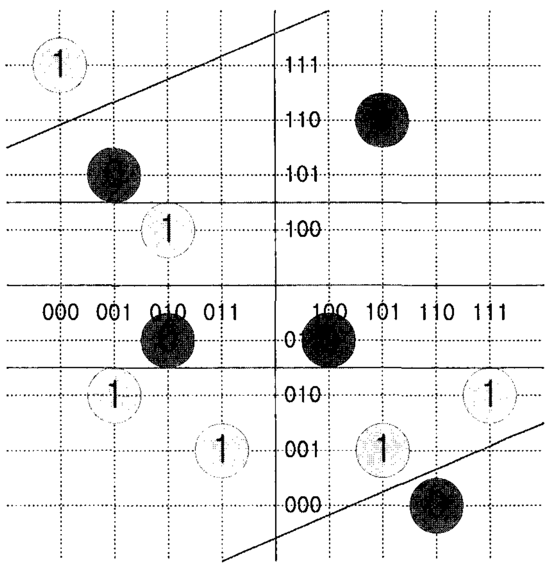


Fig. 10 Decision boundaries with multilayer perceptron for 1/3 rate.

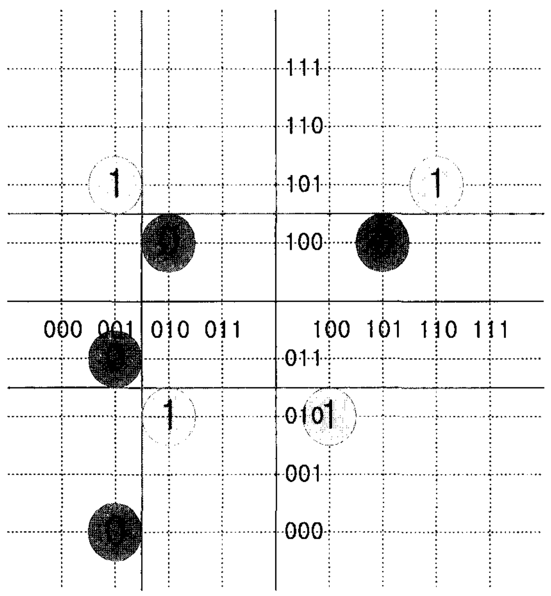


Fig. 11 Decision boundary with the SIN for 1/3 rate.

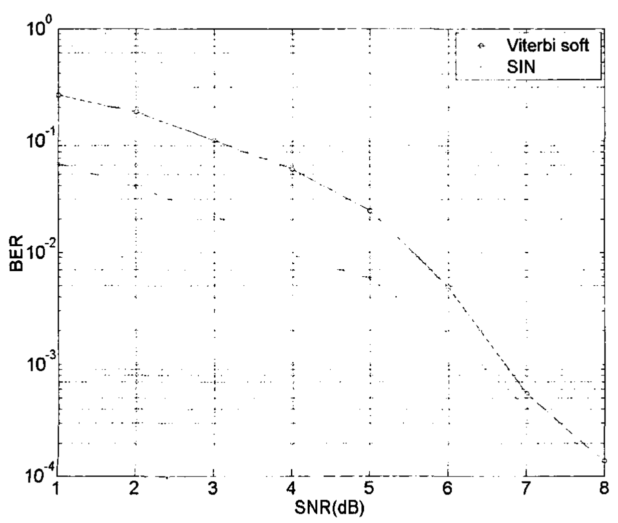


Fig. 12 The result of the SIN & Viterbi decoder for 1/2 rate.

The SIN decoder for 1/2 rate has better bit error rate and speed than Viterbi soft algorithm as shown in Fig. 10, but for the decoder for 1/3 rate the latter is better than the former for signal over SNR 4 in Fig. 13.

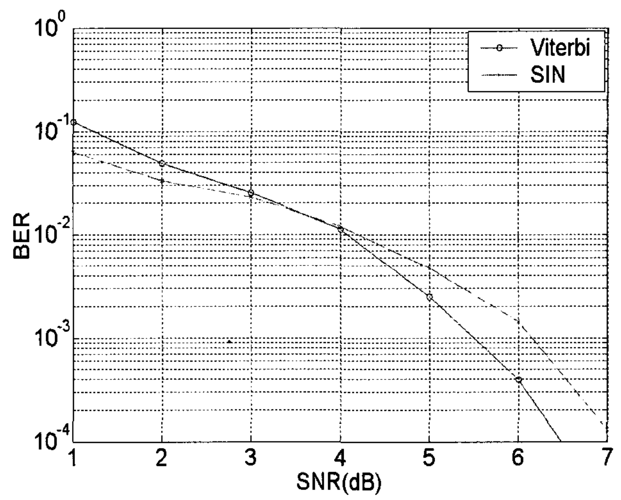


Fig. 13 The result of the SIN & Viterbi decoder for 1/3 rate.

VI. CONCLUSION

This work has proposed the SIN decoder for the receiver of digital communication systems whose signaling scheme can be characterized as a finite set of signal elements. It has been noticed that for low signal-to-noise ratio the results for the SIN decoder are the best among ones for the associated Viterbi processor and similar for higher signal-to-noise ratio. However, it must be emphasized that the decoding delay inherent to Viterbi processors do not present in the SIN decoder. We have shown how the SIN based convolutional decoders can be derived. The example was rate L=3 code. It has been demonstrated by simulations that the SIN decoder works well. Because the SIN itself is very simple, it seems possible to implement it on a fast chip. When circuit technology evolves, the faster chips can be built and this may make the proposed the SIN decoder an interesting choice.

REFERENCES

[1] J.G. Proakis, *Digital Communications*(McGraw- Hill Book, 2nd Edition, 1989).
[2] X. Wang and B. Wicker, "An artificial neural net viterbi decoder", *IEEE trans. communications*, vol. 44, no. 2, 1996.
[3] A. C. Huang and Y. F. Chiang, "Function approximation using serial input neuron", *Neurocomputing*, vol.47, pp. 85-101, 2002.
[4] A. C. Huang, "SIN-based model reference adaptive control of a class of nonlinear systems", *Proc. 6th Int, Conf. on Automation Technology*, vol. 2, pp. 657-662, May 2000.
[5] J. L. Elman, "Finding structure in time", *Cognitive Science*, vol. 14, pp. 179-211, 1990.

[6] W. Rudin, *Principes of Mathematical Analysis*(3rd Edition, McGraw-Hill Book, New York, 1976).

[7] M.T. Hagan, H.B. Demuth and M. Beale, *Neural Network Design*(PWS publishing, Boston, MA, 1996).

[8] D.W. Patterson, *Artificial Neural Networks : Theory and Applications*(Prentice Hall, Englewood Cliffs, NJ, 1996).

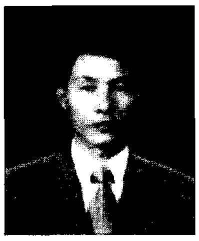
[9] G. D. Forney, Jr., "The Viterbi algorithm", *Proc. IEEE*, vol. 61(3), pp. 268-278, Mar. 1973.

[10] G. Marcone and E. Zincolini, "An efficient neural decoder for convolutional codes", *European Trans. Tele communication.*, vol. 6(4), pp. 439-445, July-Aug. 1995.

[11] A. Hämäläinen and J. Henriksson, "Convolutional decoding using recurrent neural network", *Proc. Int. Joint Conf. on Neural Networks*, July 1999.

[12] A. Hämäläinen and J. Henriksson, "A recurrent neural decoder for convolutional codes", *Proceedings of 1999 IEEE Int. Conf. on Communications*, pp. 1305-1309, June 1999.

[13] A. Hämäläinen and J. Henriksson, "Novel use of channel information in a neural convolutional decoder", *Proc. Int. Joint Conf. on Neural networks*, pp. 337-342 vol.5, 2000.



Hee-Tae Chung

Received the B.S., M.S., and Ph. D. degrees electronic engineering from the Kyungpook National University, Taegu, Korea, in 1986, 1988, and 1996, respectively. Between 1996 and 1997, he worked as a Patent Examiner at the Korean Industrial Property Office. Currently, he is an Associate Professor at the Divension of Digital Information Engineering, Pusan University of Foreign Studies, Pusan, Korea since 1997.

His current research interests include the application of intelligent control to robot systems, remote control based on Web, adaptive control and neural networks.

Kyung-Hun Kim

Obtained his Master degree in Electrical Engineering from the Kyoungpook National University, Daegu, Korea in 2003. He is currently studying for doctor course at Pusan University of Foreign Studies.