Novel Use of Channel Information in a Neural Convolutional Decoder

Ari Hämäläinen and Jukka Henriksson
Nokia Research Center, P.O.Box 407, 00045 Nokia Group, Finland

Abstract

A neural convolutional decoder which exploits the channel information is introduced. The method uses a recurrent neural network, tailored to the used convolutional code and the channel model. No supervision – besides possible channel estimation – is required. Also, no distinct equalizer is needed. As an example, we show the structure of the neural decoder for 1/2 rate code with constraint length 3 in a two-path channel environment. For testing, the 1/2 rate code with constraint length 5 is used in two-path fading channels. The simulation results show that the proposed decoder works well compared to the traditional way of using some equalizer and the Viterbi decoder. The hardware implementation of the neural decoder seems feasible and its complexity increases only polynomially while in Viterbi algorithm the complexity increases exponentially as a function of the constraint length.

1 Introduction

In digital transmission we have to guarantee that the bits are correctly received. To attain this, error correction coding is often used. The convolutional codes are one possible choice for this purpose [7]. The convolutional encoding process is very simple and can be implemented using simple digital components. As usual, the decoding process is much more demanding, especially if large constraint length is used.

The optimal method for maximum likelihood decoding is the Viterbi algorithm [2]. The complexity of this algorithm increases exponentially with increasing constraint length and, thus suboptimal algorithms have been proposed. Some of them are based on artificial neural networks [1, 5, 6, 8, 9], but the problem is that the performance of these solutions is not good enough from the practical point of view.

In [3] it was shown, how a recurrent neural network (RNN) convolutional decoder can be derived. The simulation results showed that, using this neural decoder, it is possible to get very close to performance of the optimal Viterbi decoder. In [4] the RNN decoder was tested with the codes proposed to the third generation WCDMA cellular systems.

In this paper we propose and test a new structure of the RNN decoder where the channel information is included. Traditionally the effect of the multipath channel is removed using some equalizer and then the bits are decoded. Here the channel information is included into the decoder and, thus, no additional equalizer is needed.

In Section 2 it is shown, how the RNN decoder with two-path channel information for 1/2 rate code with constraint length $L = 3$ can be derived. Similar RNN decoders for other codes and channel models can be built. In Section 3, the simulation results for 1/2 rate code with $L = 5$ in a two-path Rayleigh fading channel are given. Finally in Section 4 some conclusions are drawn.

2 Encoder and RNN Decoder with Channel Information

The general transmission system is depicted in Fig. 1. In the transmitter the data bits are first encoded. These encoded bits are modulated and then transmitted over the noisy channel. The channel here may be a radio channel, optic fiber, or some other media. In the receiver the signal is demodulated and then the bits are decoded using the RNN decoder. The estimated channel coefficients are included into the RNN decoder.

0-7695-0619-4/00 $10.00 © 2000 IEEE

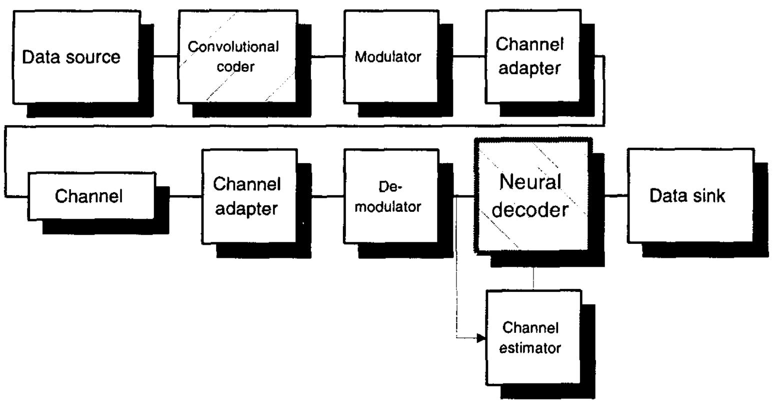

Figure 1: System model.

To show how the decoder can be derived, we fix the modulation to 4 QAM and the code to 1/2 rate code with constraint length $L = 3$. So in this case the transmitted symbol is a codeword. The assumed channel model is a two-path channel.

The decoding problem is to find the bit sequence B that minimizes the function $f(B)$:

$$\min_B f(B) := \min_B \sum_{s=0}^{T} \| r(t+sdt) - h\gamma_c(B(t+sdt)) \|^2, \tag{1}$$

where $r(t+sdt)$ is the received noisy complex valued symbol, $h = [h_1, h_2]$ the vector of complex valued channel coefficients, γ is the encoder, and $\gamma_c(B(t+sdt)) = [\gamma(B(t+sdt)), \gamma(B(t+(s-1)dt))]$ the vector, where the components are the two sequential codewords. The time index is s and the sampling interval is dt. In the following discussion $s := t + sdt$.

The generator matrix for the 1/2 rate code with $L = 3$ is ([7], Ch. 5.3.5.)

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

The encoder for $\frac{1}{2}$ rate code with constraint length L can be written as the complex function

$$\gamma(B(s)) = \gamma_1(B(s)) + i\gamma_2(B(s)),$$

where real and imaginary parts are given by

$$\gamma_j(B(s)) = \prod_{i=1}^{L} b(s+1-i)^{g_{j,i}}(-1)(-1)^{g_{j,i}},$$

and $g_{j,i}$s are the elements of the matrix G. So, in this case ($L = 3$) the encoder is

$$\gamma(B(s)) = -b(s)b(s-2) + ib(s)b(s-1)b(s-2)$$

and the received noisy codeword is $r(s) = r_1(s) + ir_2(s)$.

As in [3], one can derive the structure of the neuron by using the gradient descent. Because of the limited space, we show only the result here. The derived structure for the kth neuron is

$$S_k = f_a(-\mathrm{re}(r(k)h_1^c)S_{k-2} + \mathrm{im}(r(k)h_1^c)S_{k-1}S_{k-2}$$
$$+\mathrm{im}(r(k+1)h_1^c)S_{k+1}S_{k-1} - \mathrm{re}(r(k+1)h_2^c)S_{k-2}$$

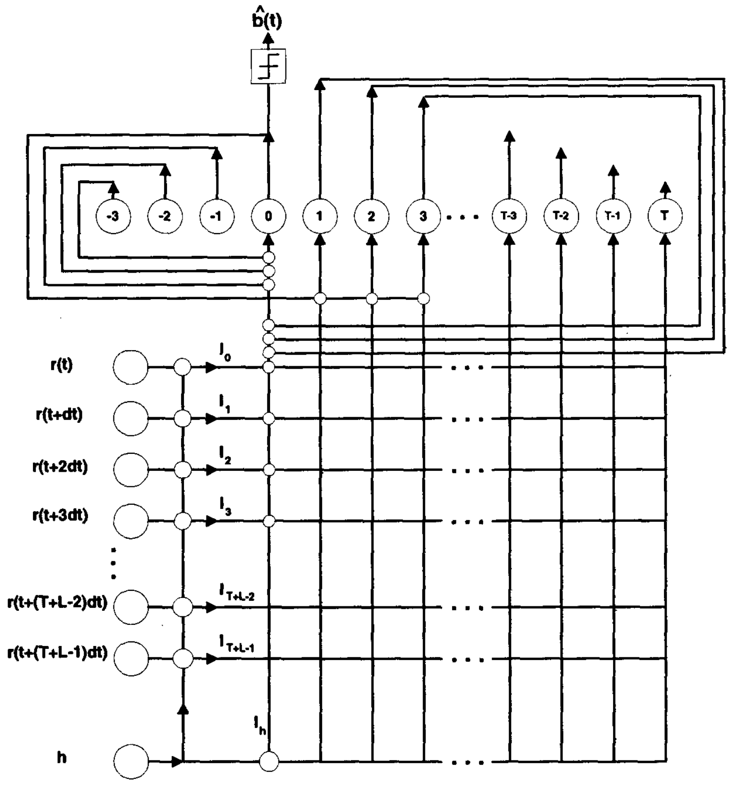

Figure 2: The RNN decoder. The connections of the neuron 0 are shown. The neuron has inputs from neurons -3,-2,-1,1,2,3 and from external inputs I. The neuron feeds it output to neurons 1,2 and 3. Similar connections are also for neurons $1, ..., T-3$. The states of the neurons -3,-2 and -1 are set to previous bit decisions and the neurons $T-2, ..., T$ at the end have only connections from the existing neurons.

$$\begin{aligned}
&+\mathrm{im}(r(k+1)h_2^c)S_{k-1}S_{k-2} - \mathrm{re}(r(k+2)h_1^c)S_{k+2} \\
&+\mathrm{im}(r(k+2)h_1^c)S_{k+2}S_{k+1} + \mathrm{im}(r(k+2)h_2^c)S_{k+1}S_{k-1} \\
&-\mathrm{re}(r(k+3)h_2^c)S_{k+2} + \mathrm{im}(r(k+3)h_2^c)S_{k+2}S_{k+1} \\
&-\mathrm{re}(h_1^c h_2)S_{k-3} \\
&-\mathrm{re}(h_1^c h_2)S_{k-2}S_{k-1}S_{k-3} + \mathrm{im}(h_1^c h_2)S_{k-2}S_{k-3} \\
&-\mathrm{im}(h_1^c h_2)S_{k-1}S_{k-3} - \mathrm{re}(h_1^c h_2)S_{k+1}S_{k-1}S_{k-2} \\
&-\mathrm{im}(h_1^c h_2)S_{k+1}S_{k-2} - \mathrm{re}(h_1^c h_2)S_{k+2}S_{k+1}S_{k-1} \\
&+\mathrm{im}(h_1^c h_2)S_{k+2}S_{k-1} - \mathrm{re}(h_1^c h_2)S_{k+3}S_{k+2}S_{k+1} \\
&+\mathrm{im}(h_1^c h_2)S_{k+3}S_{k+1} - \mathrm{im}(h_1^c h_2)S_{k+3}S_{k+2} \\
&-\mathrm{re}(h_1^c h_2)S_{k+3} + \mathrm{AWGN}\ )
\end{aligned} \tag{2}$$

where f_a is an activation function, typically a sigmoid function or a hard-limiter. In this paper we use only the signum function $f_a(x) = \mathrm{sign}(x)$. The index $k = 0$ corresponds to the current time $s = 0$ and c means the complex conjugate. The network has a fixed number of neurons and, thus, at the boundary some of the connections are missing. These connections can be modeled by setting states of the neurons $T+1, \ldots$ to 0 [3, 4]. The structure of the network is depicted in Fig. 2 and the neuron in Fig. 3.

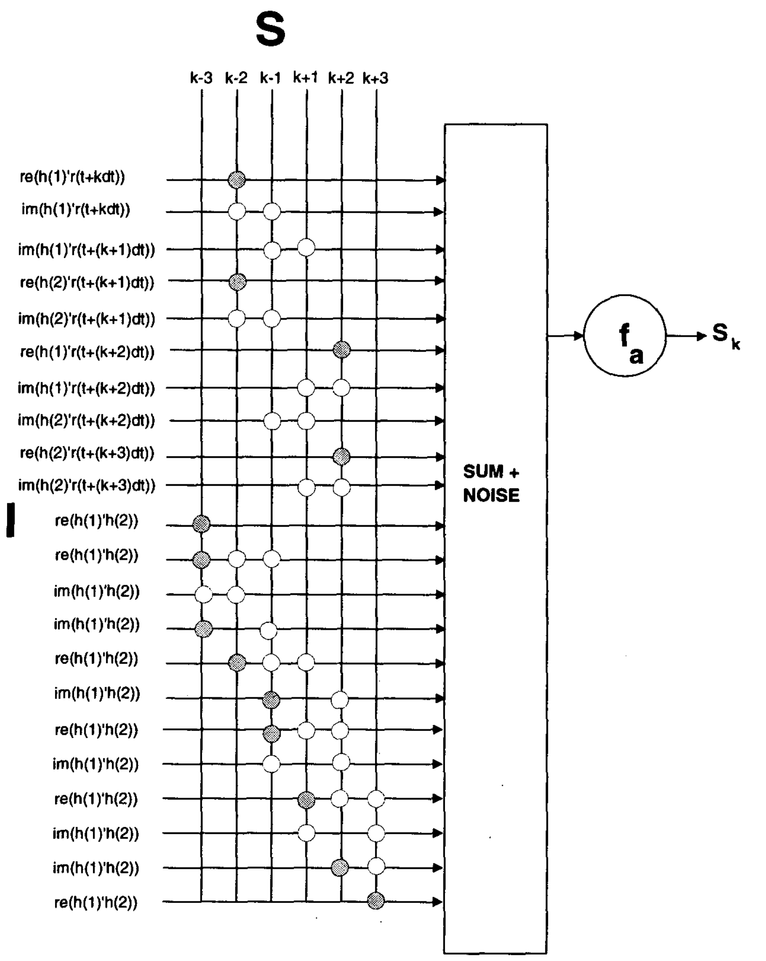

Figure 3: The structure of a single neuron. Inputs I are multiplied by those states S were the connections are marked with dots. A gray dot means an additional multiplication by -1.

The inputs to the network at time t are the received noisy codewords $r(t), ..., r(t + Tdt)$ and the channel coefficients h, or rather the products of the components of r and h.

Because the RNN is an optimization network, it may get stuck in local minima. To reduce the possibility of a local minimum we can add some white Gaussian noise in each neuron (AWGN). Further, to improve the performance, we can use several parallel networks and choose the best result by using re-encoding [3, 4].

The functioning of the RNN decoder can be written as follows.

RNN decoder: For each received codeword $r(0)$ at time t do

1. Fix $S_{-(L-1)}$,...,S_{-1} to previous decisions, $b(-L+1)$,...,$b(-1)$. Set the products of received codewords $r(s)$ and estimates h_1 and h_2 to inputs I_s.
2. Initialize $S(i)$, $i = 0, ..., T$, e.g., using random bit values.
3. For each $k = 0, ..., T$, update neuron k using equation (2).
4. Decrease the variance of the additive noise and go to 3 or stop if some fixed number of iterations has been done or some other stopping criterion is fulfilled. The decoded bit $\hat{b}$ is the state S_0 of the neuron 0.

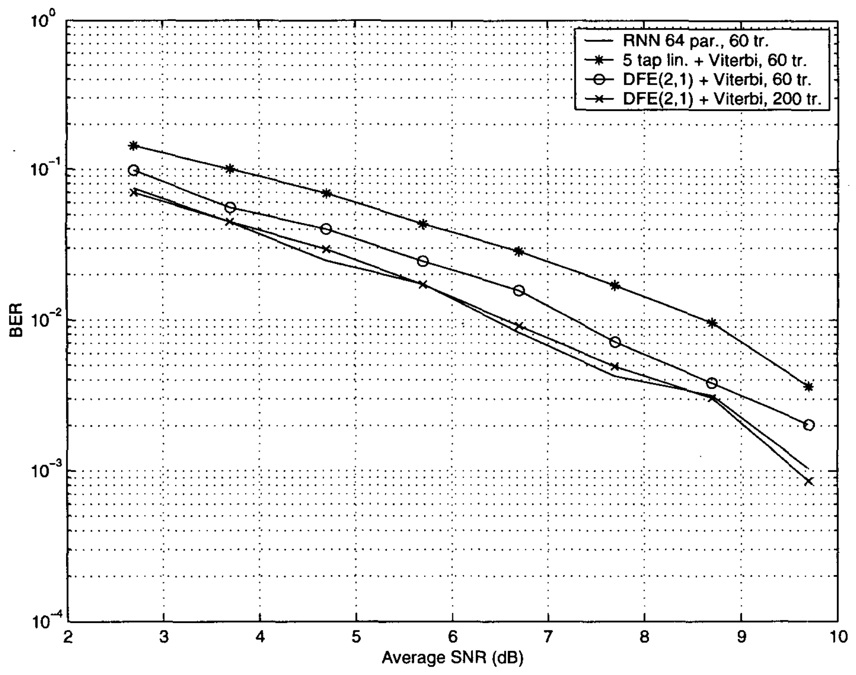

Figure 4: Simulation test for 1/2 rate code with $L = 5$. Bit error rate (BER) vs. average signal to noise ratio per bit (SNR). The RNN decoder is compared to Viterbi decoder with linear (lin.) or decision feedback (DFE) equalizer.

The code used in this section was very simple. In practise, stronger codes are used. Using this same approach, we can derive the RNN decoder for other codes and also for more complicated channel models.

3 Simulations

We tested the RNN decoder with a two-path Rayleigh fading channel. The fading corresponds to 120 km/h mobile speed. The bits were transmitted in bursts. The number of transmitted bursts was 200 and the channel was assumed to be constant during one burst. The burst consists of the training bits and data bits. The channel coefficients and equalizer parameters were estimated from the training bits and the bit error rate was estimated from the data bits.

The code used in the test is 1/2 rate code with $L = 5$ and its generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

The channel coefficients h_1 and h_2 are estimated from the 60 bits training sequence. The RNN stabilizes in 1000 iterations and 64 parallel networks are used. The variance of the additive noise is $\sigma^2 = 9.0$ at the beginning and decreases during the stabilization period.

In Fig. 4 the RNN decoder is compared to a 5-tap linear equalizer + Viterbi (lin.+Vitrbi) and decision feedback equalizer + Viterbi (DFE+Viterbi), where the DFE has two feedforward and one feedback taps. The parameters of the equalizers are estimated from the training sequence of 60 or 200 bits. As can be seen, the RNN decoder works very well. The DFE+Viterbi needs a longer training sequence to achieve the same performance as the RNN decoder.

4 Discussions and Conclusions

In this paper we proposed the RNN decoder, in which the channel information is included and, thus, a separate equalizer is not needed. The simulation showed that we can get the same performance as by using the equalizer and Viterbi decoder.

The advantage of the RNN decoder is that its complexity is of the order of $O([LH]^3)$, where H is the number of the channel taps. In Viterbi algorithm the complexity increases exponentially $O(2^L)$ as a function of constraint length L. If channel information is included into the Viterbi decoder, its complexity increases even faster, the order of $O(2^{LH})$. This shows that for high values of L and H the RNN decoder might be a tempting choice. However, this comparison does not tell all, because the complexity is different. In Viterbi algorithm the complexity comes from the memory needed, while in RNN decoder it is connections and multiplications. The ultimate comparison can be done based on the size on silicon or on the implementation costs.

Several parallel networks were used mostly to speed up the simulations. If the RNN decoder can be built as a fast circuit, then one or only a few parallel networks might be enough to get good results.

Interleaving process is not included into our model. This is the topic of the further work. Also, this idea should be tested with more complex channel models.

When circuit technology evolves, faster chips can be built and then this approach to integrate the channel estimate and decoding process (also interleaving) might be interesting choice, especially for high constraint lengths L. A patent application on this invention has been submitted.

Acknowledgment This study has been supported by Nokia Mobile Phones and the Technology Development Centre Finland (TEKES) which are gratefully acknowledged.

References

[1] M.D. Alston and P.M. Chau. A neural network architecture for the decoding of long constraint length convolutional codes. In *Proceedings of the 1990 International Joint Conference on Neural Networks, San Diego, California. June 1990*, pages 121–126.

[2] G.D. Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[3] A. Hämäläinen and J. Henriksson. Convolutional decoding using recurrent neural networks. In *Proceedings of International Joint Conference on Neural Networks, Renaissance Hotel, Washington, DC, USA, July, 10-16*, 1999.

[4] A. Hämäläinen and J. Henriksson. A recurrent neural decoder for convolutional codes. In *Proceedings of 1999 IEEE International Conference on Communications, Vancouver, Canada, June 6-10*, pages 1305–1309, 1999.

[5] S. Haykin. *Neural Networks: a Comprehensive Foundation.* Macmillan, 1994.

[6] G. Marcone and E. Zincolini. An efficient neural decoder for convolutional codes. *European Trans. Telecomm.*, 6(4):439–445, July-August 1995.

[7] J.G. Proakis. *Digital Communications.* McGraw-Hill, second edition, 1989.

[8] V. Sagar, G.M. Jacyna, and H. Szu. Block-parallel decodeing of convolutional codes using neural network decoders. *Neurocomputing*, 6(4):455–471, Aug. 1994.

[9] X. Wang and S.B. Wicker. An artificial neural net Viterbi decoder. *IEEE Trans. Communications*, 44(2):165–171, February 1996.