A COMPARISON BETWEEN RECURRENT NEURAL NETWORK ARCHITECTURES FOR DIGITAL EQUALIZATION

Jorge D. Ortiz-Fuentes[1] *Mikel L. Forcada*[1]

[1] *Universitat d'Alacant,*
Dept. Llenguatges i Sistemes Informàtics,
E-03071 Alacant (Spain).

ABSTRACT

This paper shows a comparison between three different first-order recurrent neural network (RNN) architectures (fully recurrent, partially recurrent, and Elman), trained using the real-time recurrent learning (RTRL) algorithm and the GSM training sequence ratio (26/114) for digital equalization of 2-ary PAM signals. The results show no substantial effect of the particular architecture or the number of units on the overall performance. This is due to the assumption of a suboptimal equalization scheme by the RNNs, because of the learning algorithm. The results are compared to those obtained using a classical (decision-feedback equalizer) approach.

1. INTRODUCTION

When digital signals are transmited through a communication channel, one of the problems that arises is intersymbol interference (ISI). ISI is mainly due to multipath distortion (the fact that signals arrive with different delays due to the presence of different propagation paths) or to the restricted bandwidth of the channel. The signal must then be reconstructed at the receiver by using an equalizer that approximates the inverse of the filter modelling the channel. When channel conditions are not stationary, the need for an adaptive equalizer arises. We will focus on *trained* adaptive equalizers, that is, those that transmit a training sequence of bits known both to the transmitter and to the receiver, which the equalizer uses to *learn* how to reverse the effect of the channel on the original signal.

Many kinds of trained adaptive equalizers have been described: some of them are based on statistical methods, others are based on linear system theory, such as the linear transversal equalizer (LTE) and the decision feedback equalizer (DFE)[1]. Neural networks (NN) —a natural choice for an adaptive, trainable system— have recently been applied to this field achieving better performance than classical methods in some aspects[5, 6]. In particular, recurrent neural networks (RNN) are, in some respects, very similar to DFEs in that outputs are fed back to the classifier to assist in subsequent decisions[2, 3]; however, unlike DFEs, RNNs may store additional information about the past signals in the form of an internal state, that is, their behavior may not be explained solely in terms of a finite window of inputs and outputs.

This summary describes a comparison among three classical RNN architectures for a digital equalization task. Section 2 describes the network architectures used. Section 3 describes the parameters of the simulations and presents the results. Finally, the conclusions are given in Section 4.

2. NETWORK ARCHITECTURES

We have chosen three classical first-order recurrent neural architectures: a simple, fully recurrent NN used by Kechriotis *et al.*[6], a partially recurrent neural network used by Robinson and Fallside[7] for speech recognition purposes, and an augmented recurrent architecture having a layer mapping states to outputs, used by Elman[4] to study temporal sequences. All the architectures are single-input, single-output (SISO) and have N hidden state units. The input, the state of the i-th hidden unit, the network output, and the desired output at time t are denoted by $u[t]$, $x_i[t]$, $y[t]$, and $d[t]$ respectively. The state vector will be denoted by $\mathbf{x}[t]$.

The equations describing these architectures are:

Fully recurrent (FR) NN:

$$\mathbf{x}[t] = F_{N,N+1}(\mathbf{x}[t-1], u[t]);$$

$$y[t] = x_1[t].$$

Partially recurrent (PR) NN:

$$\mathbf{x}[t] = F_{N,N+1}(\mathbf{x}[t-1], u[t]);$$

$$y[t] = F_{1,N+1}(\mathbf{x}[t-1], u[t]).$$

Elman RNN:

$$\mathbf{x}[t] = F_{N,N+1}(\mathbf{x}[t-1], u[t]);$$

$$y[t] = F_{1,N}(\mathbf{x}[t]).$$

where $F_{i,j}$ stands for a single-layer perceptron having i outputs and j inputs, and therefore ij weights and i biases. The total number of parameters (weights and biases) for each network is: FRNN, $N^2 + 2N$; PRNN, $N^2 + 3N + 2$; Elman NN, $N^2 + 3N + 1$. The activation function of all units is the hyperbolic tangent.

These architectures are trained by using Williams and Zipser's [8] real-time recurrent learning (RTRL) algorithm which updates weights every time a target or desired output is supplied.

3. SIMULATION CONDITIONS AND RESULTS

3.1. Effects of the architecture

Two kinds of simulations were done to test the effects of *architecture* and *number of hidden units*. The training sequence ratio for all of them was 26/114, the same as in the Global System for Mobile communications (GSM) standard (training sequences alternate with data) [9]. Every single simulation was run over 1000 GSM blocks of random data contaminated with additive white Gaussian noise and filtered through two different channel models: a minimum-phase channel ($H(z) = 1 + 0.7z^{-1}$) and a non-minimum-phase (NMP) channel ($H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$). The decision delay was set to the delay of the signal with the highest energy. The learning rate chosen was $\alpha = 0.1$ after preliminary experimentation, and a weight decay factor of $\gamma = 0.001$ [10] was applied. Digital values -1 and $+1$ are represented by neural outputs $-y_p$ and $+y_p$, with $y_p = 0.9$. Results for each case are averaged over 10 simulations each, and each simulation starts with small random weights and biases around 0.0.

For the architecture comparison, a network with $N = 3$ hidden units was selected for each network design (fully connected, partially recurrent, and Elman nets). The number of adjustable parameters (weights and biases) is comparable (15, 20, and 19 respectively). Figure 1 shows the bit-error-rate (BER) performance vs. signal-noise ratio for each architecture. The perfomance is almost undistinguishable, a surprising result in view of the apparent differences both in representational capability.

We then set out to assess the effect of the number of hidden units. Figure 2 shows the results for the fully recurrent and the Elman architectures, using 1 to 4 hidden units and the non-minimum-phase channel (the results with the partially recurrent network were very similar). The data clearly indicate that the number of units does not appreciably affect the performance.

3.2. Internal representation

After obtaining the results shown on figure 2, we studied the weight structure of trained nets, to infer the equalization strategy learned, and checked whether the learning algorithm was setting most of the weights to 0 so that the RNN used a single hidden unit. Although some of them showed the structure mentioned before, the rest of the tests did not point so clearly in that direction. This may be probably due to the assumption of suboptimal distributed strategies which are roughly equivalent to a local strategy using a single hidden unit. This deserves a more detailed study that will be reported elsewhere.

3.3. Suboptimal strategy

The last step to study the suitability of using RNN for digital equalization as compared with other approaches was to test whether the assumption of a suboptimal strategy [2, 3] was due to the training algorithm or to an inherent representational shortcoming of the architecture.

Since the worst results were obtained for the NMP channel, we decided to train a 2-hidden-neuron FR recurrent neural network with its weights set to the values that allow it to emulate the behavior of an analytically obtained DFE for that channel. When noise is low, a simple DFE such as

$$y[t] = \mathrm{sgn}[A(u[t] - 0.8704y[t-1] - 0.3482y[t-2])]$$

will equalize it. A possible FRNN realization of this DFE would be:

$$x_1[t] = \tanh(4.698u[t] - 4.089x_1[t] - 1.636x_2[t-2]),$$

$$x_2[t] = \tanh(1.636x_1[t-1]),$$

with $y[t] = x_1[t]$ for no decision delay and $y[t] = x_2[t]$ for a decision delay of 1 unit. The rest of the weights would be zero. Figure 3 shows the performance of this synthetic RNN equalizer, which is clearly superior to any of the trained equalizers in this paper.

It is interesting to report here that, when training starts from this synthetic network instead of from a network with small random weights and biases, the BER performance of the equalizer is somewhat degraded, especially when noise is low; this is shown in figure 3. Additional experiments show that degradation is worsened by the presence of the weight decay term γ. Also, higher values of the learning rate α lead to more degradation; this is due to the fact that the weight update rule of the RTRL algorithm is an approximation to true gradient descent in the limit $\alpha \to 0$. The results shown correspond to $\gamma = 0$ and $\alpha = 0.1, 0.01, 0.001$.

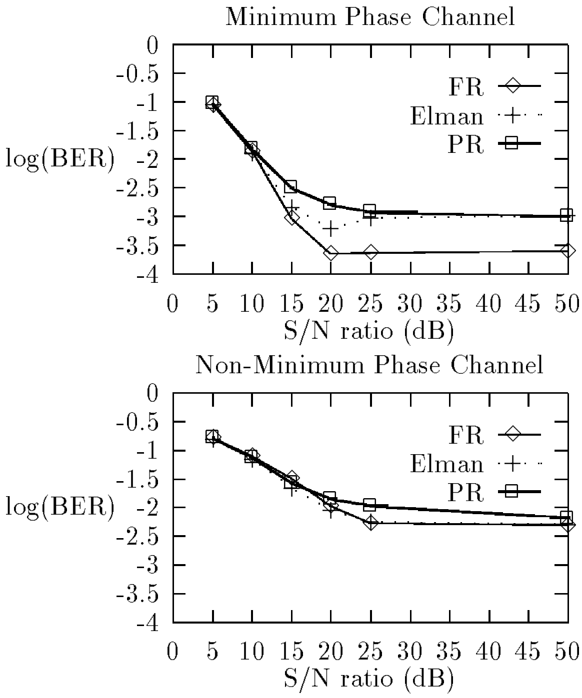

Figure 1. Bit error rate versus signal-noise ratio for the three different architectures (FR: fully recurrent; PR: partially recurrent).

4. CONCLUSIONS

Our work shows that:

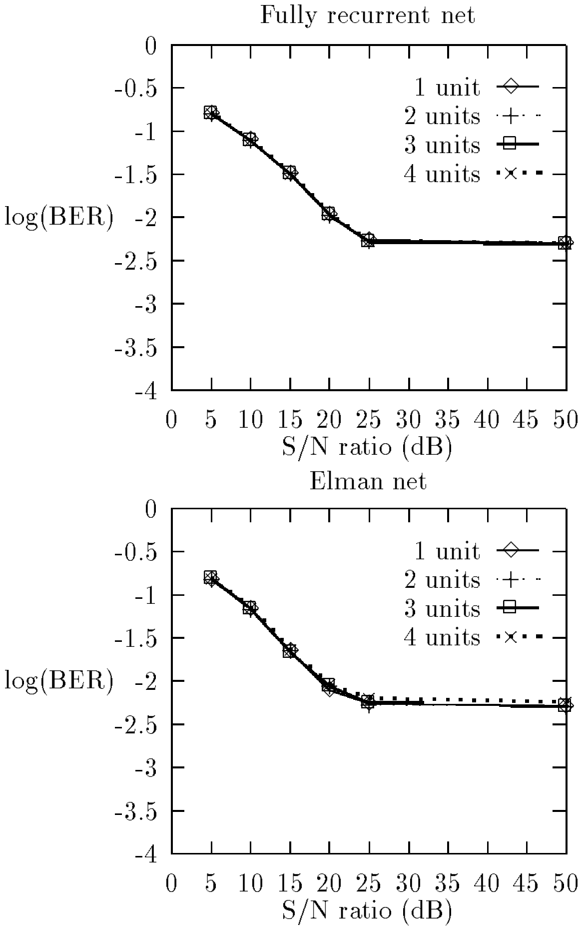

Figure 2. Bit error rate versus signal-noise ratio for the fully recurrent and Elman nets as a function of the number of hidden units

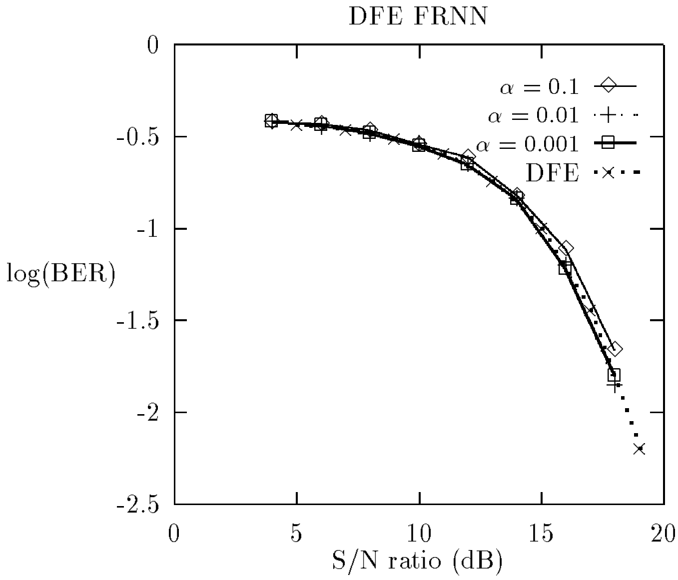

Figure 3. Bit error rate versus signal-noise ratio for the FRNN emulating the optimum DFE for the channel, before and after training.

- The BER performance of RNN equalizers trained using RTRL on 2-ary PAM signals using the GSM training ratio (26 train/114 data) is both independent of the particular architecture used and independent of the number of hidden units in the RNNs.

- This is due to the assumption of a suboptimal equalization strategy which is equivalent to using a single hidden neuron; indeed, we show that the architectures are capable of a much better BER performance when weights are set to suitable values. The suboptimality of results indicates that RTRL may not be the best choice of learning algorithm for adaptive equalization purposes.

- The RTRL learning algorithm tends to degrade the performance of a synthetic RNN designed to mimic the optimum DFE, when it is run on real data. This is due in part to the fact that the error gradient is approximated but not exactly computed by the RTRL learning rule.

This suggests that one of the main lines of our future research should address learning algorithms: RTRL seems to be inefficient for this purpose and is computationally very expensive. We also plan to study the suboptimal solutions reached by the network in more detail. It is also important to note that the work reported here deals with stationary channels, a very uncommon situation in mobile digital communications.

Acknowledgments: The authors wish to acknowledge the support of the Spanish Comisión Interministerial de Ciencia y Tecnología through grant TIC95-0984-C02-01.

REFERENCES

[1] Proakis, J.; *Digital Communications*, New York: McGraw-Hill (1995).

[2] Bradley, M.J.; Mars, P.; "A critical assessment of recurrent neural networks as adaptive in digital communications", *Proc. IEE Colloquium on Applications of Neural Networks to Signal Processing*, London, (1995), p. 11/1–4.

[3] Bradley, M.J.; Mars, P.; "Application of recurrent neural networks to communication channel equalization", *Proc. ICASSP '95*, Detroit, (1995), **5**: 3399–3402.

[4] Elman, J.L.; "Finding structure in time", *Cognitive Science* **14** (1990) 179–211.

[5] Chen, S.; "Adaptive equalisation using neural networks", in Murray, A.F.(ed.), *Applications of Neural Networks*, Kluwer, (1995), 241–265.

[6] Kechriotis, G.; Zervas, E.; Manolakos, E.S.; "Using Recurrent Neural Networks for Adaptive Communication Chanel Equalization", *IEEE Trans. on Neural Networks*, (1994), **5**:2, 267–278.

[7] Robinson, A.J.; Fallside, F.; "A recurrent error propagation speech recognition system", *Computer Speech and Language* **5**, 259–274.

[8] Williams, R.J.; Zipser, R.A; "A learning algorithm for continually training recurrent neural networks", *Neural Computation* **1** (1989) 270–280.

[9] Scourias, J. "Overview of the global system for mobile communications" `http://ccnga.waterloo.ca/~jscouria/GSM/gsmreport.html` (1995).

[10] Hertz, J., Krogh, A., Palmer, R.G.; *Introduction to the Theory of Neural Computation*, Reading, Mass.: Addison-Wesley (1991), p. 157.

A COMPARISON BETWEEN RECURRENT NEURAL NETWORK ARCHITECTURES FOR DIGITAL EQUALIZATION

Jorge D. Ortiz-Fuentes[1] *and Mikel L. Forcada*[1]

[1] *Universitat d'Alacant,*
Dept. Llenguatges i Sistemes Informàtics,
E-03071 Alacant (Spain).

This paper shows a comparison between three different first-order recurrent neural network (RNN) architectures (fully recurrent, partially recurrent, and Elman), trained using the real-time recurrent learning (RTRL) algorithm and the GSM training sequence ratio (26/114) for digital equalization of 2-ary PAM signals. The results show no substantial effect of the particular architecture or the number of units on the overall performance. This is due to the assumption of a suboptimal equalization scheme by the RNNs, because of the learning algorithm. The results are compared to those obtained using a classical (decision-feedback equalizer) approach.