Solving Nonlinear Equations Using Recurrent Neural Networks

Karl Mathia and Richard Saeks, Ph.D.
Accurate Automation Corporation
7001 Shallowford Road
Chattanooga, Tennessee 37421

Abstract

A class of recurrent neural networks is developed to solve nonlinear equations, which are approximated by a multilayer perceptron (MLP). The recurrent network includes a linear Hopfield network (LHN) and the MLP as building blocks. This network inverts the original MLP using constrained linear optimization and Newton's method for nonlinear systems. The solution of a nonlinear equation with computer simulation illustrates the algorithm.

1 Problem Context

A class of recurrent neural networks is used to solve nonlinear equations, where the motivation for using artificial neural networks is their learning capability. For this work the properties of multilayer perceptron (MLP) as universal function approximators [3] are of particular advantage: data with unknown structure, taken from a nonlinear system, are approximated by an MLP. The MLP is then inverted, i.e. the approximated equation is solved, using recurrent neural networks. This class of recurrent networks is comprised of MLPs and LHN [4] as building blocks.

Finding the solution of *n*-dimensional nonlinear equations can be a challenging problem, even when a unique solution exists. The recurrent networks presented here employ Newton's method for nonlinear systems [1] for the solution of such systems. The LHN perform linear optimization, and if necessary, a generalized linear Hopfield network is used to perform constrained linear optimization [5]. The solution of a nonlinear equation is presented as an application example.

2 Multilayer Perceptron and Linear Hopfield Network

The *multilayer perceptron* is a widely used feedforward neural network, in particular since the backpropagation learning algorithm by Werbos [7] was popularized in 1986 [6]. The class of MLPs considered here is shown in Figure 1a. The neurons in the single hidden layer have sigmoid activation functions, while the output neurons are linear [3]. The MLP is viewed as a parameterized nonlinear mapping $\boldsymbol{f}:\Re^m \to \Re^n$,

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}_2 \cdot \boldsymbol{g}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{w}_b) = \boldsymbol{W}_2 \cdot \boldsymbol{g}(\boldsymbol{a}). \tag{1}$$

The free parameters $\boldsymbol{W}_1 \in \Re^{q \times m}$, $\boldsymbol{W}_2 \in \Re^{n \times q}$ and $\boldsymbol{w}_b \in \Re^{q \times 1}$, the weight matrices of hidden layer, output layer, and the weight vector which connects bias element and hidden layer, are specified during training. The nonlinear operator $\boldsymbol{g}:\Re^q \to \Re^q$ represents the hidden layer with q sigmoidal neurons, with input vector $\boldsymbol{a}$ (activation) and output vector $\boldsymbol{z}$,

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{a}) = [g(a_1), g(a_2), \ldots, g(a_q)]^T . \tag{2}$$

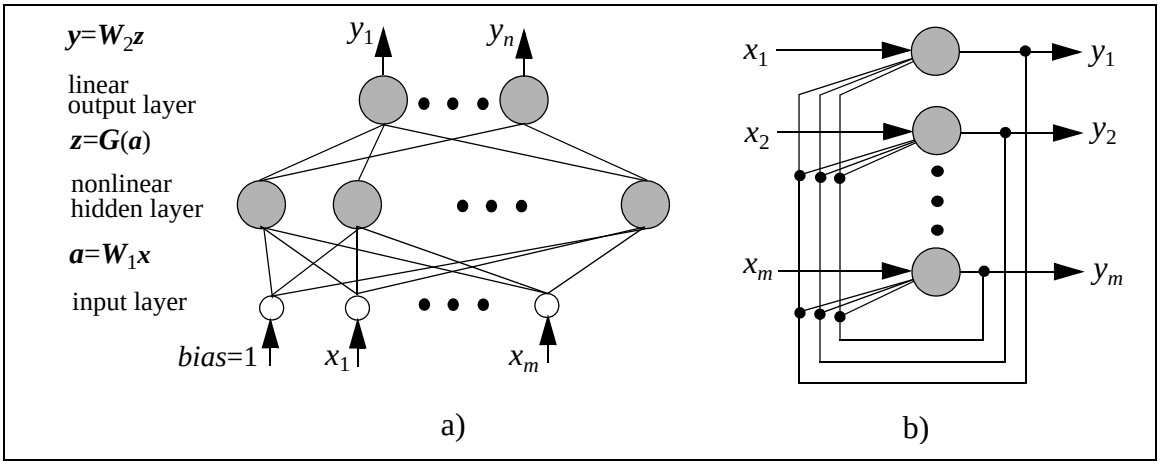

Figure 1 Architectures of a) the multilayer perceptron, and b) the linear Hopfield network.

The architecture of *Hopfield networks* [2] is shown in Figure 1b, with constant input $\boldsymbol{x}$ and state $\boldsymbol{y}$. For this work the nonlinear activation functions (step or sigmoid functions) are replaced with linear activation functions. The discrete-time dynamics of the linear Hopfield network (LHN) are then defined by the difference equation

$$\boldsymbol{x}_{k+1} = \boldsymbol{W}\boldsymbol{x}_k + \boldsymbol{y}, \tag{3}$$

where k is discrete time and $\boldsymbol{W} \in \Re^{m \times m}$ is the weight matrix. The LHN is used to solve linear equations of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. In [4] and [5] it is shown that the LHN in Equation (3) converges to a unique solution if the spectral radius ρ satisfies $\rho(\boldsymbol{W}) < 1$. Necessary and sufficient conditions for the design of stable $\boldsymbol{W}$ are

$$\boldsymbol{W} = \boldsymbol{I} - \alpha \boldsymbol{A}^T \boldsymbol{A}, \quad \boldsymbol{y} = \alpha \boldsymbol{b}, \quad 0 < \alpha < \frac{2}{\rho(\boldsymbol{A}^T \boldsymbol{A})}, \tag{4}$$

for both invertible and noninvertible $\boldsymbol{A}$. If $\boldsymbol{A}$ is singular the Moore-Penrose pseudo-inverse $\boldsymbol{A}^+$ is computed, for nonsingular $\boldsymbol{A}$ the pseudo-inverse is identical with the 'normal' inverse.

3 Solving Nonlinear Equations using Recurrent Networks

Newton's method for nonlinear systems is an iterative algorithm to solve systems of nonlinear equations $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, the algorithm is defined by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}_k^{-1}(\boldsymbol{f}(\boldsymbol{x}_k) - \boldsymbol{y}), \tag{5}$$

where $\boldsymbol{J}_k = \boldsymbol{J}(\boldsymbol{x}_k)$ is the Jacobian matrix of $\boldsymbol{f}$ at iteration step k. Newton's method gives quadratic convergence, provided a sufficiently accurate initial condition $\boldsymbol{x}_0$ and a nonsingular Jacobian matrix $\boldsymbol{J}$ at all iteration steps [1]. Here the algorithm provides a pattern for designing recurrent neural networks. When applying Newton's method to solve Equation (1) we make two assumptions: first, the inverse of $\boldsymbol{f}$ exists,

that is $m = n$, and second, the MLP has more hidden neurons then it has input and output nodes, that is $dim(\boldsymbol{f}) < dim(\boldsymbol{G})$, or $m < q$. The Jacobian of the MLP is obtained via the chain rule,

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{a}} \cdot \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{W}_2 \cdot [\boldsymbol{G}(\boldsymbol{a})(\boldsymbol{I} - \boldsymbol{G}(\boldsymbol{a}))]_{\boldsymbol{a} = \boldsymbol{a}_0} \cdot \boldsymbol{W}_1 \, . \tag{6}$$

The simplicity of the partial derivative of $\boldsymbol{G}$ with respect to $\boldsymbol{a}$ is due to the properties of the sigmoid function [6]. With the MLP in Equation (1) and the Jacobian in Equation (6) the recurrent network is defined by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha(\boldsymbol{W}_2(\boldsymbol{G}_k(\boldsymbol{I} - \boldsymbol{G}_k))\boldsymbol{W}_1)^{-1}(\boldsymbol{W}_2\boldsymbol{g}(\boldsymbol{W}_1\boldsymbol{x}_k + \boldsymbol{w}_b) - \boldsymbol{y}) \, , \tag{7}$$

where $\boldsymbol{G}_k = \boldsymbol{G}(\boldsymbol{a}_k)$ is a constant diagonal matrix. Assuming a sufficiently accurate starting point $\boldsymbol{x}_0$, the question still remains if the inverse Jacobian exists (m=n). This can be assumed, since the matrix $\boldsymbol{G}_k(\boldsymbol{I} - \boldsymbol{G}_k)$ is positive definite, and if both weight matrices have full (row or column) rank. Since singularities cannot be excluded, at least in theory, or if $m \neq n$, a constrained algorithm like the generalized linear Hopfield network [5] can be applied to compute a suitable generalized inverse. Then the result is not altered when using $\boldsymbol{J}^+$, as mentioned above. The learning rate α in Equation (7) does not appear in Equation (5) and is used here to obtain smooth trajectories. The block diagram of Newton's method and the corresponding recurrent neural network are shown in Figure 2.

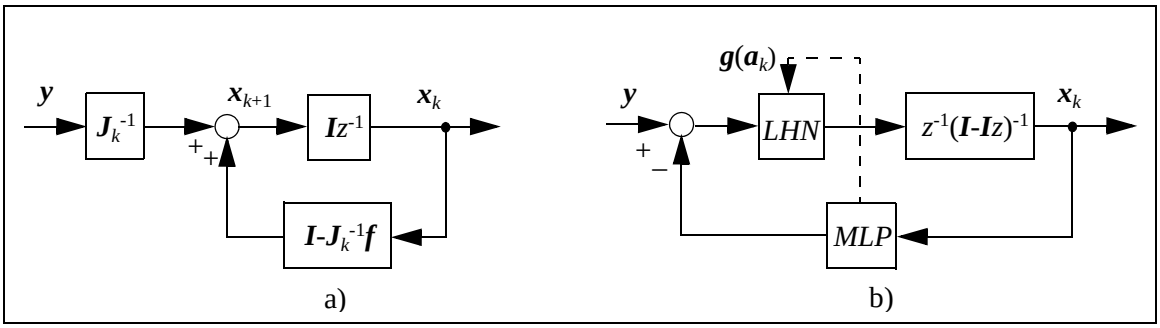

Figure 2 Block diagram of a) Newton's method, and b) the corresponding recurrent network with multilayer perceptron (MLP) and linear Hopfield network (LHN) as building blocks.

4 Example

The performance of the recurrent network in Equation (7) is illustrated with the solution of the following quadratic Equation (8) (m = n = 2). A multilayer perceptron (Figure 1a) with q = 16 hidden neurons learned this mapping over a 'data grid' of the input range $-1.5 \leq (x_1, x_2) \leq 1.5$, where the distance between data points was 0.1. The error backpropagation learning algorithm [6] was used.

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} -0.3 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 + (x_1 - 0.5)^2 + (x_2 + 0.25)^2 \\ 0.5 + (x_1)^2 + (x_2 - 0.5)^2 \end{bmatrix} \, . \tag{8}$$

The performance of the MLP was tested on 441 points on $-1 \le (x_1, x_2) \le 1$, a subset of the train set, and a mean square error of 5.272×10^{-4} was obtained. The recurrent network in Figure 2b, including the trained MLP and the appropriate LHN, was run for four different initial conditions x_0. The convergence rate was α = 0.1, the convergence criterion was $\|x_{k+1} - x_k\| \le 10^{-5}$. The four resulting trajectories in both input space ***X*** and output space ***Y*** are shown in Figure 3, the corresponding data are listed in Table 1.

Table 1: Data corresponding to the four trajectories.

trajectory	x_0	x_f	y_0	y_{err} (x10^{-3})
#1	(0.4 , 0.4)	(0.4450, -0.3363)	(-0.9660 , 0.0248)	(-0.1210 , -0.0068)
#2	(-0.3, 0.3)	(-0.4150 , 0.9908)	(-0.6577 , 0.4295)	(-0.0573, 0.0884)
#3	(0.2 , 0.8)	(-0.4150 , 0.9909)	(0.8807 , -0.0657)	(-0.1046 , -00175)
#4	(0.0 , -0.2)	(0.4449 , -0.3364)	(-0.4693 , 0.3384)	(-0.0350 , 0.0793)

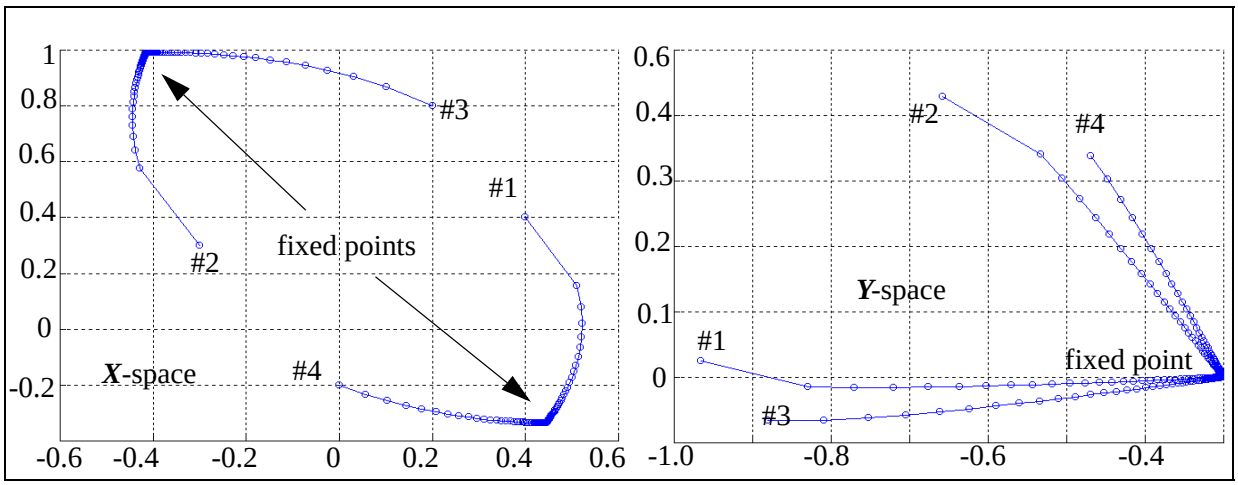

Figure 3 Solving the example equation using the recurrent network in Figure 2b: trajectories from four different initial conditions (numbered) to fixed points ***x*** which satisfy ***y***=***f***(***x***).

5 References

[1] R.L. Burden and J.D. Faires, *Numerical Analysis*, PWS-Kent, Boston, MA, 1988.

[2] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sciences USA*, Vol. 81, pp. 3088-3092, 1984.

[3] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol. 2, pp. 359-366, 1989.

[4] K. Mathia and R. Saeks, "Inverse Kinematics via Linear Dynamic Networks," *Proc. World Congress on Neural Networks*, San Diego, Vol. 2, pp. 47-53, June 1994.

[5] K. Mathia, R. Saeks and G. Lendaris, "Linear Hopfield Networks, Inverse Kinematics and Constrained Optimization," *Proc. IEEE Conf. Sys., Men, and Cyber.*, Vol. 2, pp. 1269-1273, Oct. 1994.

[6] E. Rumelhart and J.L. McClelland (eds.), *Parallel Distributed Processing*, Vol. 1, Chapter 8, MIT Press, Cambridge, MA, 1986.

[7] P.J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. The-

sis, Harvard University, Cambridge, MA, 1974.