Using Neural Networks to Estimate Regions of Stability

Enrique D. Ferreira and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Av., Pittsburgh, PA, 15213-3890 USA
edf/krogh@ece.cmu.edu

Abstract

This paper presents a new method to estimate the region of stability of an asymptotically stable equilibrium point of an autonomous nonlinear system using a neural network. In contrast to model-based analytical methods, this approach uses empirical data from the system to train the neural network. The neural network results are compared with estimates obtained by previously proposed methods for some sample two dimensional problems and for an inverted pendulum.

1. Introduction

The problem of estimating the region of stability for the stable equilibrium of autonomous nonlinear systems is fundamental in the theory of dynamic systems, and has been studied for many years [15] [5]. In applications, knowledge of regions of stability is essential for the safe operation of many complex dynamic systems, such as power systems and nuclear reactors [14]. Despite many years of theoretical attention to this problem, and its clear practical importance, the existing methods for estimating stability regions remain limited. They often can be applied only to low-dimensional systems, and they are generally very conservative.

This paper presents a new method to estimate the region of attraction of a stable equilibrium for an autonomous nonlinear system using a neural network. The motivation for using a neural network to estimate the region of stability is threefold. First, as a universal approximator to nonlinear functions, a neural network imposes no a priori constraints on the shape of the boundary estimation. This is in contrast to analytical methods based on Lyapunov techniques which restrict the form of the boundary to the particular parameterization chosen for the Lyapunov function.

The second motivation for using a neural network is that it can be trained using empirical data from the actual system. The proposed neural network approach does not require an analytical model of the system dynamics. This is particularly appealing for the application for which this method is being developed, namely, to determine switching rules for the SIMPLEX real-time control architecture which supports fail-safe operation of dynamic systems when the control laws are being changed and tested on line [10]. These switching rules require estimates of the stability regions for closed-loop systems under various control laws, and since real dynamic systems are being controlled, the rules need to be correct for the actual dynamics, not just for models of the systems.

Thirdly, a neural network computes a classification result quickly. In the context of the SIMPLEX architecture, it must be known in real time whether the current system state is in the stability regions for the given controllers. Although it may take considerable time to train the neural network, either off line or on line, the decision about whether or not the current state is within the region of stability must be made within the sampling rate of the real-time control system.

2. Problem Formulation

Let the state-constrained nonlinear autonomous system be:

$$\begin{aligned} \dot{x} &= f(x) \\ x &\in D \\ f(0) &= 0 \end{aligned} \tag{1}$$

with $x = 0$ being an asymptotically stable equilibrium point. The nonlinear function f is assumed continuous and smooth on the constrained set D. Let us define the trajectory of the system (1) with the initial state x_o at time t_o as $x(t, t_o, x_o)$. The stability region R for $x = 0$ is a connected, invariant set defined as:

$$R = \{x_o : \; x(t, t_o, x_o) \in D \; \forall t \geq t_o \text{ and } lim_{t \to \infty} x(t, t_o, x_o) = 0\}$$

The problem of interest is to design a classifier that determines whether a given state is in R. This classifier should be conservative, but as accurate as possible, and fast enough to be used on-line.

Several methods have been proposed to compute inner (i.e., conservative) approximations to the boundary of R (see e.g.[1] and references therein). All of these methods assume a model of the system dynamics is given. Consequently, for real applications the system dynamics must be known precisely or measures must be taken to make the boundary estimate conservative enough to account for model uncertainties.

Most of the proposed methods involve the computation of a Lyapunov function for the system. For example, Davison and Kurak [2] developed an algorithm to find a quadratic Lyapunov function that maximizes the volume of the stability region. Vannelli and Vidyasagar [12] estimated the region of stability from the development of a "maximal" rational Lyapunov function. Noldus and Loccufier [5] proposed a technique for improving a given Lyapunov function estimate by integrating the system dynamics backwards in time, away from the Lyapunov boundary at specific points, to enlarge the estimate of R.

Marpaka [3] proposed that for power system applications where fast real-time estimates are needed, one could train a feedforward neural network to characterize a *given* Lyapunov function. Although this proposal addresses the problem of on-line computation via a neural network, it does not advance any methods for actually obtaining the Lyapunov function itself.

3. Stability Region Estimation

The ability of neural networks to classify information and to approximate functions to a specific degree of accuracy is well known. Their design is not an automatic procedure, however. The network architecture and the training methods to be used must be selected with good engineering judgment, and research continues to provide new insights for making these decisions. The selection of the network size and training data is also crucial. Recent results by Rao et al. [7] state a bound on the number of samples needed to get specific probability of error in the approximation depending on the smoothness of the function and the learning parameters for feedforward networks. The method developed by Sanner and Slotine [8], based on the smoothness of the function being approximated and using Gaussian radial basis functions, leads to very accurate approximations, but has the disadvantage that network sizes can get very large, growing at the same rate as the data needed. Further developments using wavelets seem to be very promising for reducing the required network size [9].

3.1. Neural network architecture

Figure 1 shows the architecture of the proposed neural network to estimate stability regions. A multilayer feedforward network was selected for its capacity as an universal approximator with a size that is small relative to the size of the data set [8]. Even though one hidden layer is sufficient to achieve accurate approximations, the use of two hidden layers plus a linear input-output component gives more robust results [4], albeit at the cost of longer training.

The inputs to the neural network are the components of the system state vector, $\mathbf{x}$, and the output is a two-dimensional vector, $\mathbf{y}$. The components of $\mathbf{y}$ will range roughly between -1 and 1 in the region of approximation, giving an indication of the membership of the state vector in R. The ideal output values are $(y_1, y_2) = (1, -1)$ when $\mathbf{x}$ is R and $(y_1, y_2) = (-1, 1)$ when $\mathbf{x}$ is not in R. To make a distinct classification in the non-ideal case (i.e., when the components of $\mathbf{y}$ are not equal to ± 1), two positive threshold parameters, θ and δ, are selected to implement the following decision rule:

Declare $\mathbf{x}$ belongs to if:

$$1. \quad y_1(\mathbf{x}) - y_2(\mathbf{x}) > \theta$$
$$2. \quad \|\nabla \ (y_1(\mathbf{x}) - y_2(\mathbf{x}))\| < \delta$$

where ∇ denotes the gradient with respect to $\mathbf{x}$.

The parameter θ is chosen after the network is trained so that the classification is correct for all the training and validation data. The parameter δ is chosen much less than the maximum of the norm of the gradient of the network output over the domain. The second rule does not accept x as being in R in regions where the outputs are changing value, that is, near the threshold of the estimate of the region boundary.

The relationship between inputs and outputs for the network in figure 1 is given by

$$\mathbf{y} = \mathbf{b}_o + W_o \mathbf{x} + W_3 \ g(W_2 \ g(W_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2), \qquad (2)$$

where $W_i, \mathbf{b}_i$ are the weight matrices and threshold vectors, respectively. The nonlinear function $g(.)$ for the network hidden units is the hyperbolic tangent function. The output units are linear. The approximation is then a combination involving linear and nonlinear functions.

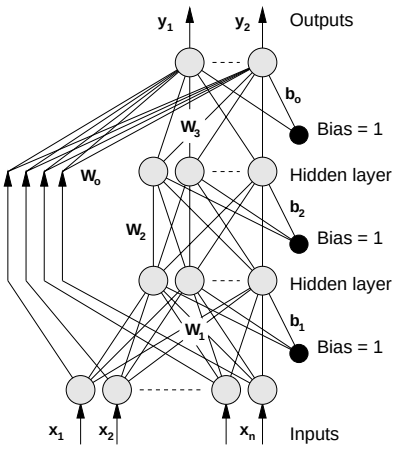

Figure 1: Neural network scheme used.

3.2. Training procedure

Two types of training procedures are used:

1. *Supervised.* A priori knowledge of the desired output of the network for a set of given inputs is used to adjust the parameters of the network to reproduce the known input-output relationship. Often, we already have

data from the system that can be used to train the network off-line.

2. *Reinforcement learning.* In an on-line situation we do not know a desired value, i.e. if the input state belongs to the stability region or not, beforehand. Therefore a temporal differences method (TD_λ) [11] is applied to generate a prediction of the desired outputs for the network. The update equation for the network weights w at step k is

$$w_k = \eta(P_{k+1} - P_k).\sum_{i=1}^{k} \lambda^{k-i} \nabla_w P_i,\ k = 1, \ldots, N.$$

$$P_k = \mathbf{y}_k,\ k = 1, \ldots, N. \qquad P_{N+1} \equiv \hat{\mathbf{y}}$$

with $\mathbf{y}_k$ the network output at step k and $\hat{\mathbf{y}}$ known data. This algorithm is much faster than just waiting until the end of an experiment to get precise data to train the network [13].

The other element of the training procedure is the design of the data to use with the algorithms described above. For supervised learning, let us assume the function $h(\mathbf{x})$ to be approximate by the network satisfies a Lipschitz condition:

$$\|h(\mathbf{x}_1) - h(\mathbf{x}_2)\| \leq L\ \|\mathbf{x}_1 - \mathbf{x}_2\|$$

In an ideal situation in which experiments are possible and cheap, we can define a fine grid to cover the variation of the function by means of the sampling theorem. Training samples are taken in a square mesh inside the region selected to approximate the function. For reinforcement learning the initial states are taken uniformly random in the region to investigate. Additional supervised training can be performed with representative patterns to improve the quality of on-line learning, and sliding windows over the trajectories can be applied to reduce the computation [6].

A conjugate-gradient, Polak-Ribiere minimization procedure is used. The initial values of the weights are randomly chosen. Early stopping using a test set is used to prevent overtraining. A pruning algorithm is run to optimize the size of the network by getting rid of the hidden units which do not correlate enough with the output.

4. Some 2-D examples and comparisons

In this section we examine the results of the proposed method for simple second order systems and compare the results with the ones obtained by Davison and Kurak [2] and Vannelli and Vidyasagar [12]. In these examples there is no need to apply reinforcement learning because we already know the real region of stability.

4.1. Van der Pol equation

The state equations are:

$$\begin{aligned} \dot{x}_1 &= -x_2 \\ \dot{x}_2 &= x_1 - x_2 + x_1^2 x_2 \end{aligned} \tag{3}$$

Knowing the region of stability in this case, the limits in the state space were taken as:

$$|x_1| \leq 2.5, \qquad |x_2| \leq 3.0$$

The error values and parameters were averaged over 5 runs. Table 1 shows the results of the proposed method. Figure 2 shows the estimation of that region according to the three methods. The network approximation is closer to the actual stability region than the other methods.

Training samples:	3721
Testing samples:	961
Units in each layer:	2 - 10 - 2 - 2
Training set MSE:	0.105
Test set MSE:	0.152
$\theta,\ \delta$:	0.9 / 1.1

Table 1: Van der Pol equation: Estimation results.

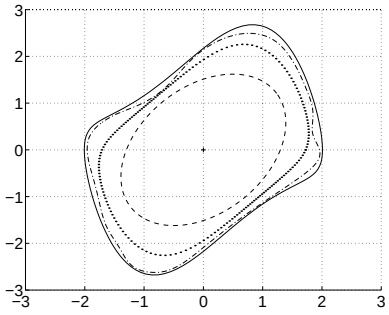

Figure 2: Van der Pol equation: Estimates comparison.

solid:	actual stability region.
dotted:	Vannelli and Vidyasagar method
dashed:	Davison and Kurak method
dash-dotted:	neural network estimation.

4.2. Second example

Taken also from [2], the state equations are:

$$\begin{aligned} \dot{x}_1 &= -x_1 + 2x_1^2 x_2 \\ \dot{x}_2 &= -x_2 \end{aligned} \tag{4}$$

In this case the region of stability is unbounded:

$$R = \{\mathbf{x} :\ x_1.x_2 < 1\}$$

The method developed in [12] gives a perfect estimation of the region. The method reported in this paper is essentially designed for a bounded region. Therefore, the intersection of the stability region and the region, given by

$$|x_1| \leq 3.0, \qquad |x_2| \leq 3.0$$

was estimated. Table 2 shows the results of the proposed

Training samples:	3721
Testing samples:	1024
Units in each layer:	2 - 8 - 2 - 2
Training set MSE:	0.0128
Test set MSE:	0.0144
θ, δ :	0.2 / 1.3

Table 2: Network estimation results

method. Figure 3 shows the estimation of the stability region according to the three methods. Note that the neural network approximation is very close to the hyperbole.

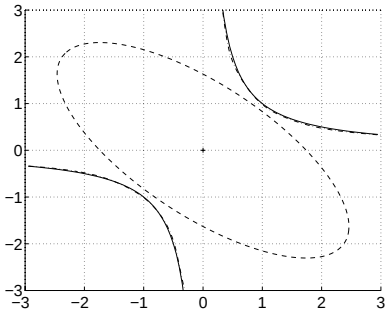

Figure 3: Comparison between estimates.

solid:	actual stability region and Vannelli and Vidyasagar method
dashed:	Davison and Kurak method
dash-dotted:	Neural network estimation*

* This estimation is very close to the hyperbole.

5. The Inverted Pendulum

As a higher dimensional example, we consider the inverted pendulum (IP). A physical IP has served as a standard laboratory example for the SIMPLEX architecture described in the introduction. Here we present results of the neural network estimation of the stability region based on data from a simulation model using both supervised and reinforcement learning. The IP model equations are:

$$\begin{aligned} J_t\ddot{x} + \frac{1}{2}\,ml\,cos\alpha\,\ddot{\alpha} + B_r\dot{x} - \frac{1}{2}\,ml\,sin\alpha\,\dot{\alpha}^2 &= F \\ \frac{1}{2}\,m\,cos\alpha\,\ddot{x} + \frac{1}{3}ml\ddot{\alpha} - \frac{1}{2}\,mg\,sin\alpha &= 0 \end{aligned}$$

with the following constraints:

$$|F| \le F_{max}, \quad |x| \le x_{max}, \quad |\dot{x}| \le v_{max}$$

The parameters take the values:

J_t	0.6650	Kg	g	9.8	m/s^2
m	0.21	Kg	F_{max}	2	N
l	0.61	m	x_{max}	0.45	m
B_r	0.1	Kg/s	v_{max}	1	m/s

Figure 4 shows a model scheme for a standard IP. Supervised and reinforcement learning is applied. Table 3 summarized some parameters calculated for the network and some results obtained on the simulator. Figures 5 and 6 show some cuts of the region found for the model compared to the best linear region based on constraints for individual states variables, calculated by maximizing the volume of the hypercube defined by the constraints.

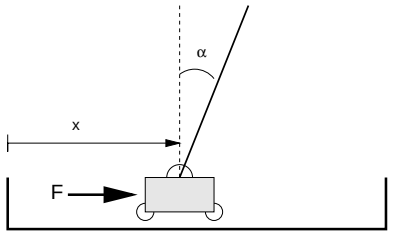

Figure 4: Inverted pendulum model scheme.

Training samples:	1000
Testing samples:	300
Units in each layer:	4 - 12 - 6 - 2
Training set MSE:	0.01
Test set MSE:	0.05
θ, δ :	1.8 / 0.9

Table 3: Network estimation results

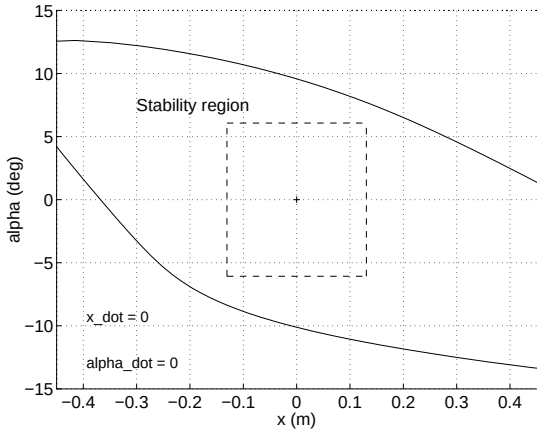

Figure 5: State space cut showing the estimated stable region in the (x, α) plane for $\dot{x} = \dot{\alpha} = 0$.

solid	neural network estimation.
dashed:	best linear constraints.

6. Conclusions

A method to estimate the stability region of an autonomous nonlinear system using neural networks is presented. Although the model of the system is not used directly, any a priori knowledge might be used for train-

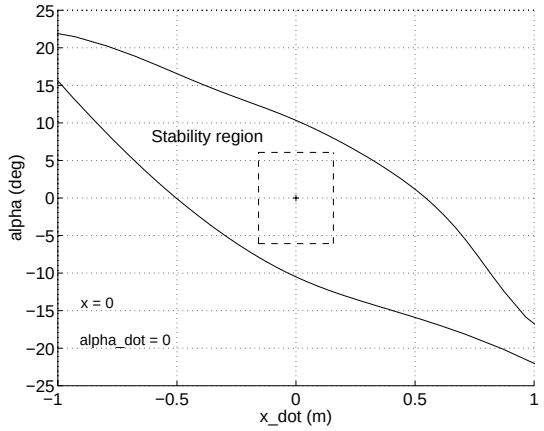

Figure 6: State space cut showing the estimated stable region in the $(\dot{x}, \alpha)$ plane for $x = \dot{\alpha} = 0$.

solid	neural network estimation.
dashed:	best linear constraints.

ing. Preliminary results in a simulation environment with a control benchmark system as an inverted pendulum are shown with satisfactory results.

In applications, the knowledge of a region of stability can have several uses. If we think of the nonlinear autonomous system as a closed-loop system with state and input constraints, the stability region estimation can allow us to make a decision to shutdown the system or change controllers when the system goes out of the stable region. It is necessary to perform experiments to estimate the region of stability, but actual data from the system operation can be used as training data for the network to improve its estimation.

Further development needs to be done in setting up the conditions on the nonlinearities of the system to determine precise bounds on the parameters involved in the decision process to allow a more accurate estimation. The possibility of using recurrent networks needs to be studied considering a different strategy with emphasis in the use of on-line development.

7. Acknowledgments

This research is been supported by CONICYT-IBD, OAS and the Software Engineering Institute at Carnegie-Mellon University.

References

[1] H-D. Chiang, M.W. Hirsch, and F. Wu. Stability regions of nonlinear autonomous dynamical systems. *IEEE Trans. on Automatic Control*, 33(1):16–27, Jan 1988.

[2] E.J. Davison and E.M. Kurak. A computational method for determining quadratic lyapunov functions for nonlinear systems. *Automatica*, 7:627–636, 1971.

[3] D.R. Marpaka. Recognition of stability domains of nonlinear dynamical systems using multilayer feed-forward artificial neural networks. In *XXIII Southeastern Symp. on Syst. Theory*, pages 212–217, Columbia, SC, USA, Mar 1991. IEEE.

[4] P.M. Mills, A.Y. Zomaya, and M.O. Tade. *Neuro-Adaptive Process Control*. John Wiley & Sons Ltd., 1996.

[5] E. Noldus and M. Loccufier. A new trajectory reversing method for the estimation of asymptotic stability regions. *Int. J. Control*, 61(4):917–932, 1995.

[6] D.A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. Technical report, Computer Science Dept. Carnegie Mellon University, 1989.

[7] N.S.V. Rao, V. Protopopescu, R.C. Mann, E.M. Oblow, and S.S. Iyengar. Learning algorithms for feedforward networks based on finite samples. *IEEE Trans. on Neural Networks*, 7(4):926–940, Jul 1996.

[8] R.M. Sanner and J-J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Trans. on Neural Networks*, 3(6):837–863, 1992.

[9] R.M. Sanner and J-J. E. Slotine. Structurally dynamic wavelet networks for the adaptive control of uncertain robotic systems. In *Proc. 34th CDC*, pages 2460–2467, New Orleans, LA, Dec 1995. IEEE.

[10] L. Sha, R. Rajkumar, and M. Gagliardi. The simplex architecture: An approach to build evolving industrial computing systems. In *Proc. of The ISSAT Conference on Reliability*, 1994.

[11] R.S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[12] A. Vannelli and M. Vidyasagar. Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21:69–80, Jan 1985.

[13] P. Werbos. A menu of designs in reinforcement learning over time. In W.T. Miller III, R.S. Sutton, and P. Werbos, editors, *Neural Networks for control*, chapter 3. MIT Press, 2nd edition, 1991.

[14] J. Zaborsky, G. Huang, B. Zheng, and T-C. Leung. On the phase portrait of a class of large nonlinear dynamic systems such as the power system. *IEEE Trans. on Automatic Control*, 33(1):4–15, Jan 1988.

[15] V.I. Zubov. *Methods of A. M. Lyapunov and their application*. P.Noordhoff Ltd., 1964.