Using Artificial Neural Networks and Support Vector Regression to Model the Lyapunov Exponent

Adam Maus*

April 13, 2009

Abstract:

Finding the salient patterns in chaotic data has been the holy grail of Chaos Theory. Examples of chaotic data include the fluctuations of the stock market, weather, and many other natural systems. Real world data has proven to be extremely difficult to predict due to it high dimensionality and the potential for noise. It has been shown that artificial neural networks have been able to accurately calculate the Lyapunov Exponent, a feature that determines the divergence of two initially close trajectories and thus chaos. A variation of the support vector machine called support vector regression is used in function approximation and applying this tool to chaotic data may yield another model that can also be used in the prediction of the Lyapunov Exponent. Since support vector machines have been found to arrive at a prediction much faster than the artificial neural network, it may prove to be the model to use in time sensitive applications.

Introduction

Nonlinear prediction attempts to describe the inherent qualities of a given dynamical system using mathematical models. Models can range in complexity from a simple set of equations such as the logistic equation used to model the carrying capacity of a species in an ecosystem [1] to the creation of artificial neural network models with hundreds of parameters used in forecasting exchange rates [2]. These dynamical systems as well as the rise and fall of populations, the stock market, and the weather exhibit chaos where there is a sensitive dependence on initial conditions. Since two initially close conditions will diverge exponentially fast, chaotic dynamics pose a formidable challenge for long term forecasting.

* University of Wisconsin – Madison
Contact Email: amaus@wisc.edu

Aside from long term forecasts, one goal of nonlinear prediction is to create a mathematical model that accurately represents data take from a system. The hope is that by creating an accurate model, underlying features in the system will be replicated as well. One such feature, the largest Lyapunov exponent (LE) is valuable in determining whether or not a given system is chaotic. Its value describes the average rate at which predictability is lost.

The largest Lyapunov exponent is usually geometrically averaged along the orbit and is always a real number. This Lyapunov exponent is calculated numerically using the procedure in [3]. When calculating the LE for a model, we run fixed length forecasts using the models created by the artificial neural network and support vector regression model on training data. These forecasts are fixed at 5000 time steps, 5000 was chosen because it gives a good approximation of the true LE and is

quicker to calculate than one million time steps to find the true LE of the models and systems being studied.

The forecast from a model can converge to a number of different attractors that represent varying LEs. A forecast that converges exponentially fast to a fixed point will have an LE with a value of negative infinity. The limit cycle is a case that is often hard to distinguish without the calculation of the LE because the data may be following a strict path with a large number of periods or steps before repeating. The LE of a limit cycle ranges between negative infinity and 0. An LE of zero denotes a system such as

$$x_{n+1} = 3x_n\left(1 - x_n\right),$$

requiring an infinite number of points to converge to an attractor. Dynamical systems such as the Hénon map [4] have a positive LE and "converge" to what are called strange attractors. These attractors appear to follow an orbit but it never repeats. The Hénon map whose form

$$x_{n+1} = 1 - 1.4x_n^2 - 0.3x_{n-1} \ ,$$

can be initialized with x = (.1, .1) and produces a strange attractor plotted by x_n versus x_{n-1} as in Figure 1.

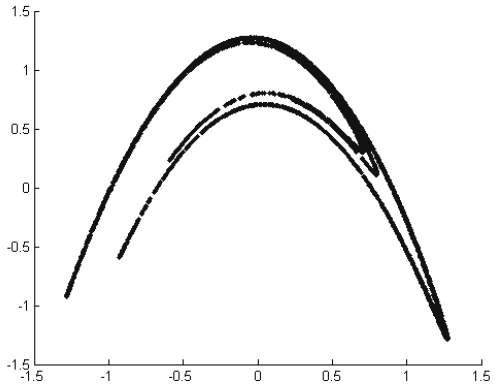

FIG 1: Strange Attractor of the Hénon Map

The LE calculated for the Hénon map using 5000 time steps equals .42093 and the goal of this paper is to describe how neural networks and support vector regression models can be used to calculate the LE for this map and other similar dynamical systems.

Models Studied

Artificial neural networks have been used in past for modeling time series [5], nonlinear prediction [6], and analyzing the underlying features of data [7]. Hornik *et al*. [8] claimed that these models were universal approximators with the ability to represent any function with an arbitrary number of hidden units. This model, seen schematically in Fig. 2, used to replicate the LE uses a single layer of hidden units to perform function approximation and takes the form,

$$\hat{x}_k = \sum_{i=1}^{n} b_i \tanh\left(a_{i0} + \sum_{j=1}^{d} a_{ij} x_{n-j}\right),$$

where a is an $n \times d$ matrix of coefficients, and b is represented by a vector of length the number of neurons n. The a matrix represents the connection strengths of the neurons, and the b vector is used to modulate the strength of a particular neuron i. In this model, a_{i0} represents the bias term that is commonly used in neural network architectures.

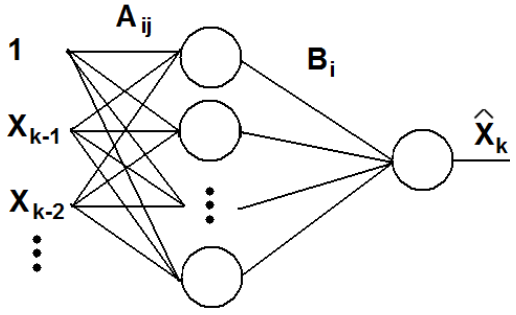

FIG. 2: Single Layer, Feed-Forward Neural Network

The neural network uses the *d* last points from a scalar time series *x* in the prediction of the next step *k*. In this model, *d* is considered the dimension or embedding of the neural network. It represents the number of inputs or previous time lags used to predict each subsequent value in the time series.

The weights in *a* and *b* are updated using a variant of the simulated annealing algorithm where temperature is held constant at 1, effectively removing the term. For each coefficient, a Gaussian neighborhood is searched whose size is changed based on progress in training. The coefficients are chosen to minimize the mean-square prediction error,

$$e = \frac{\sum_{i=1}^{c} (\hat{x}_i - x_i)^2}{c}$$

The other model studied, Support Vector Machines have traditionally been used in classification problems; in 1996 Vapnik *et al*. [9] extended this model to perform robust function approximation that is insensitive to small changes in the data. The LibSVM Toolbox for Support Vector Machines by Chang and Lin [10] was used in this project.

Vapnik claimed that ε-Insensitive Support Vector Regression would provide effective estimation of functions [11] and SVR has also been used in forecasting the stock price change with the help of analysts [12].

As Smola stated in [12] SVR is used to approximate a function $f(x)$ that has at most ε deviation from the targets x_k. This effectively removes certain data points from the model's approximation of the data. A function is found that can approximate just a subset of the actual data and because of the chaotic nature of the data being studied; using a subset of points should change the model's ability to replicate the same chaotic dynamics of the system being studied. However, we found that it performs just as well as neural networks in approximating the LE.

The function that the SVR model produces takes the form

$$f(x) = \langle w, x \rangle + b,$$

where $\langle w, x \rangle$ denotes the dot product of *w* and *x*. Calculation of *w* and *b* involves minimizing *w* through a convex optimization problem [13] that minimizes a Lagrangian function [14]. The computation of *b* can be performed in different ways. LibSVM computed *b* by finding the average value of the support vectors with Lagrange multipliers between 0 and the user-defined variable *C*.

Once a model was produced, the forecast of *p* points took the form

$$\hat{x}_p = b + \sum_{i=1}^{s} w_i K,$$

where *s* represents the number of support vectors used in the model. In the LibSVM toolbox, this was automatically calculated. *K* represents the kernel used in the model and the approximation. In this experiment, *K* took the form of a Gaussian Radial Basis function with an adjustable parameter σ. The training data and forecast were initialized with data from the Hénon map. Two time series were used to train the model. One time series served as training data for the model, the other time series served the test time series that predicted the forecasting error. After obtaining a model with a low forecasting error compared to the system being studied, forecasting was used to calculate the LE.

During the training of the SVR model, we optimize the model on the set of

data but there are a number of parameters to train to create a low test error. The size of ε, the cost of vectors on the wrong side of the function *C*, the kernel parameters, etc, must all be chosen. Since each parameter can be any real number, a training schedule was created that was similar to the neural networks training. The parameters trained *C*, ε, σ, *p*, were initialized with values .001, .01, .1, and .1, respectively. To train the parameters, the parameters were varied according to a Gaussian neighborhood whose size depends on the progress being made in training the model.

To objectively choose one model over another the mean-square prediction error was used to compare the forecast of the SVR and test data with one caveat. The size of the test time series used was effectively shortened by a weighting scale created so the model focused on replicating the first points in the test time series.

$$e' = \frac{\sum_{i=1}^{c} \psi_i \left(\hat{x}_i - x_i \right)^2}{c}$$

The ψ_1 was chosen arbitrarily for this project and set to seven and for each additional *k*, ψ_k was halved, though further research may find that ψ can be optimized for the system being studied. This scheme was created because of the chaotic nature of both the model created and the data being modeled, sensitive dependence on initial conditions would cause the two sets of data to be incomparable after just tens of steps. If we tried to minimize the mean square error between the entire forecast and the test data, it eventually led to a model that found the average between the high and low peaks in the test data.

Numerical Results

After training the SVR model on 400 points, the LE was calculated and found to be .41288 for the Hénon map with a test error of 4.6866 x 10^{-5}. The strange attractor of the SVR model produced was visually indistinguishable. It is important to note for all cases that different training instances lead to different results and these were chosen because they were the lowest mean square errors found.

The neural network was trained for one million epochs with c = 400 points taken from the Hénon Map. After training, the LE of the model was calculated. The LE was found to be .38431 with a mean-square error of 4.646 x 10^{-5}. Figure 3 shows the strange attractor produced by this network, as seen, the neural network had some trouble reproducing the attractor for the Hénon map exactly though other better trained neural networks may perform better. Likewise, the difference between the true LE of the system and the neural network may arise from stopping the training too soon. A more accurate model seems to correlate with a more accurate LE of the model to the test system.

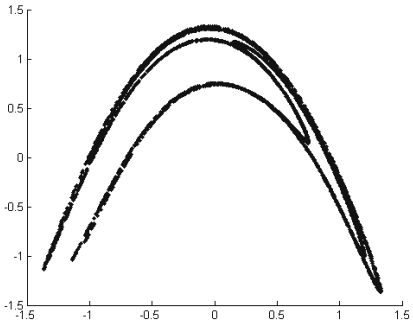

FIG. 3: The Hénon map attractor reproduced by the Neural Network

Two other systems were studied using the Neural Network (NN) and SVR models, the Logistic map (LM) [15],

$$x_{n+1} = 4x_n(1 - x_n)$$

and the delayed Hénon map (DHM) [16],

$$x_{n+1} = 1 - 1.6x_n^2 - 0.1x_{n-4}$$

which led to the following table of LEs.

	Actual	NN	SVR
LM	.69526	.63744	.68923
DHM	.37038	.35030	.35550

Table 1: The LE approximations using 5000 steps in the forecast of each system and model

The neural network and SVR model both produced strange attractors for the delayed Hénon map, Fig. 4 and Fig. 5, respectively that were visually similar to the actual strange attractor, Fig. 6. The Neural Network and SVR model achieved a mean square error of 5.6792 x 10^{-6} and 5.4632 x 10^{-6}, respectively.

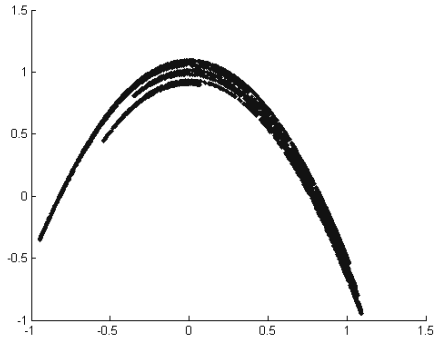

FIG. 4: The delayed Hénon map attractor reproduced by the Neural Network

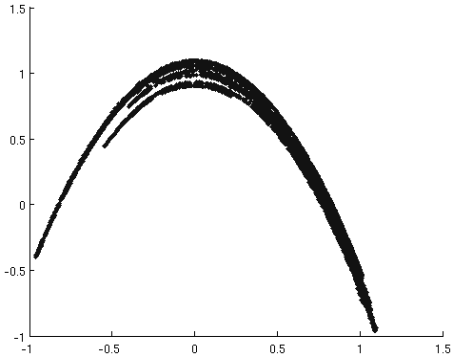

FIG. 5: The delayed Hénon map attractor reproduced by the SVR model

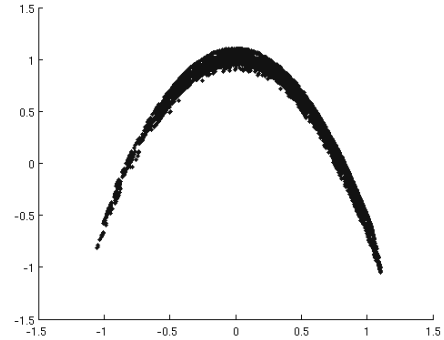

FIG. 6: Strange Attractor for delayed Hénon map

The SVR model did not reconstruct the exact structure of the strange attractor for the Logistic map but this could be due to inadequate training, the error was .004256. The neural network was able to reconstruct the attractor and the LE with an error of 3.9735 x 10^{-6}. The attractor of the logistic map and the attractor produced by the SVR model are seen in Fig. 7 and 8, respectively.

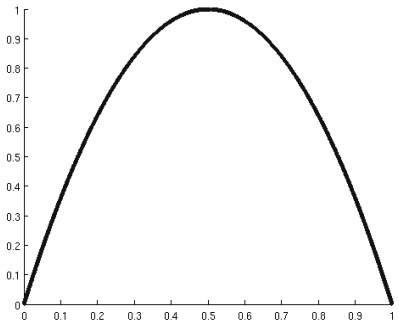

FIG. 7: Strange Attractor for Logistic map

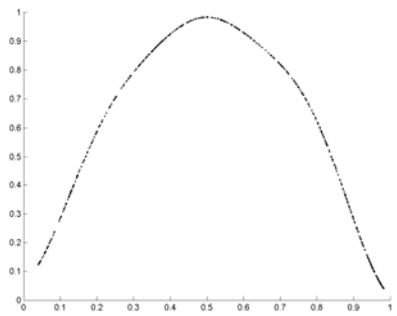

FIG. 8: The Logistic map attractor reproduced by the SVR model

Though both models produce mean square errors, it is not fair to compare them using these values. The weighting on the errors in SVR model was different from that of the Neural Network. These objective functions were simply used to train each model. If a comparison was needed, a more accurate measurement would be to identify the difference between the true Lyapunov exponent of the system being studied and the Lyapunov exponent produced by each model.

Discussions and Conclusion

Artificial neural networks serve as excellent models for reconstructing the data and studying the LE for the Hénon Map. If we applied these models on data taken from a system whose LE is unknown, they would perform well given that they are adequately trained. Certain measures could be taken to ensure that the correct LE is found which include cross validation and ensembles of neural networks among other ways to provide that the neural network does not overfit the data it is given. One drawback of this neural network model is that they require many more calculations than SVR and unlike SVR (prior to training the parameters), neural networks often fail to find the global optimum or absolute best way to model the data therefore it is imperative to train many different networks on the same data.

Support Vector Regression performed much better than expected. Mattera *et al*. found that reconstructing the strange attractor and the chaotic dynamics for a given system using SVR is not always possible [17] but by using a weighting system on the forecast of the data, it is possible to train the parameters for these models to produce an accurate representation of the data.

The number of free parameters created a challenge for trying to calculate the LE, a training schedule was created to alter these parameters and reconstruct both the chaotic dynamics and the strange attractor. Like the neural network, using this type of training causes the model parameters to fall into local optimums, one way to overcome this is to use an ensemble of SVR models to find an average Lyapunov exponent.

One important aspect of time series analysis involves determining the embedding dimension for a model. For the attractor to be reconstructed it must be embedded in an appropriate dimension so it can be unfolded so one preimage does not produce two different images. A sufficient condition for the embedding dimension was provided by Takens [18] who showed that complete unfolding is guaranteed if the time-delayed embedding space has a dimension at least one greater than twice the dimension of the original state space.

For the purpose of this project, the data was embedded in a space that was guaranteed to unfold it, by looking at the time-delayed equations of each system, the embedding could be deduced. For the Hénon map, the embedding was 2, for the delayed Hénon map, 4, and for the Logistic map, 1. For an unknown system, embedding can be calculated using an algorithm known as false nearest neighbors [19].

Support Vector Regression has shown to be useful in function approximation, forecasting, and reconstruction of chaotic dynamics for a given system. Support Vector Regression creates a model that uses far less points than the artificial neural network. Training the SVR model parameters takes less time than the neural network and in some cases it arrives at a better approximation of the LE for the system being studied. SVR serves as another model to approximate the LE for a dynamical system and it would be interesting to test SVR more rigorously on continuous systems and real-world data.

References

[1] E. Allman and J. Rhodes, *Mathematical Models in Biology: An Introduction* (Cambridge, New York, 2004).

[2] L. Yu, S. Wang, and K. K. Lai, *Foreign-Exchange-Rate Forecasting with Artificial Neural Networks* (Springer, New York, 2007).

[3] J. C. Sprott, *Chaos and Time-Series Analysis* (Oxford, New York, 2003).

[4] M. Hénon, Comm. Math. Phys. **12**, 1 (1976).

[5] S. Troncia, M. Gionab, and R. Barattia, Neurocomputing **55**, 3-4 (2003).

[6] G. Zhang, B. Patuwo, and M. Hu, Int. J. Forecasting **14**, 1 (1998).

[7] J. Principe, A. Rathie, and J. Kuo, Int. J. Bifurcation Chaos **2**, 4 (1992).

[8] K. Hornik, M. Stinchocombe, and H. White, Neural Networks **2**, 5 (1989).

[9] V. Vapnik, S.E. Golowich, and A. Smola, in *Advances in Neural Information Processing Systems, San Mateo, 1996*, edited by M. Mozer, M. Jordan, and T. Petsche, p. 155.

[10] C. Chang and C. Lin, *LibSVM: a library for support vector machines* (2001).

[11] V. Vapnik. *The Nature of Statistical Learning Theory*. (Springer, New York, 1995).

[12] Z. Zhang, C. Shi, S. Zhang, and Z. Shi, in *Advances in Neural Networks -ISNN, Chengdu, 2006,* edited by J. Wang, Z. Yi, J. Zurada, B. Lu, H. Yin.

[13] A. Smola and B. Scholkopf. Stat. Comput. **14**, 3 (2004).

[14] S. Haykin *Neural Networks, a Comphrensive Foundation.* (Prentice Hall, New Jersey, 1992).

[15] R. May, Nature, **261** (1976).

[16] J. C. Sprott, Electron. J. Theor. Phys. **3**, 12 (2006).

[17] D. Mattera and S. Haykin in *Advances in Kernel Methods: Support Vector Learning*, edited by B. Schölkopf, C. J. Burges, and A. J. Smola, p. 211.

[18] F. Takens, *Detecting strange attractors in turbulence* (Springer, Berlin, 1981).

[19] H. Abarbanel, *Analysis of Observed Chaotic Data* (Springer, New York, 1996