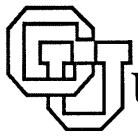
EVALUATING NEURAL NETWORK PREDICTORS BY BOOTSTRAPPING

Andreas S. Weigend and Blake LeBaron

CU-CS-725-94



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Evaluating Neural Network Predictors by Bootstrapping

Andreas S. Weigend
Department of Computer Science
and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430
andreas@cs.colorado.edu*

Blake LeBaron
Department of Economics
University of Wisconsin
1180 Observatory Drive
Madison, WI 53713
blebaron@facstaff.wisc.edu

Abstract—**We present a new method, inspired by the bootstrap, whose goal it is to determine the quality and reliability of a neural network predictor. Our method leads to more robust forecasting along with a large amount of statistical information on forecast performance that we exploit. We exhibit the method in the context of multivariate time series prediction on financial data from the New York Stock Exchange. It turns out that the variation due to different resamplings (i.e., splits between training, cross-validation, and test sets) is significantly larger than the variation due to different network conditions (such as architecture and initial weights). Furthermore, this method allows us to forecast a probability distribution, as opposed to the traditional case of just a single value at each time step. We demonstrate this on a strictly held-out test set that includes the 1987 stock market crash. We also compare the performance of the class of neural networks to identically bootstrapped linear models.**

1 Introduction

Training a network on a time series is not hard, but once we have a network, how can we evaluate the accuracy of the predictions? On the one hand, if the time series is fairly long (1,000 points or longer), and if it is fairly clean (noise of less than one per cent of the signal), the evaluation is relatively easy, since only very few functions will fit some held-back data very well. This regime can be described as a "right-by-$(1-\epsilon)$-regime." On the other hand, for short and/or very noisy time series, one can only hope to be right with a probability of $(0.5+\epsilon)$. In such a noisy regime, many functions will be indistinguishable in their forecasting quality. This is the regime financial markets are in; it is well known that random predictions, or random trading strategies, can yield deceptively long sequences of good predictions or profitable trades. When connectionist techniques are used, additional choices (such as the architecture, training procedure, and the random initialization of the network) make the evaluation even harder.

A standard procedure for evaluating the performance of a model would be to split the data into one training set (used for gradient descent in the parameters), one cross-validation set (used to determine the stopping point to avoid overfitting, and/or used to set additional parameters, such as for weight-elimination), and one or more test sets [Weigend *et al.*, 1990]. In our experience, this approach can be very sensitive to the specific sample splitting. Furthermore, if the specific test set is given and the data is sparse (as in the widely used sunspot example), tests of forecast reliability may not reflect a good picture of sample variability, or potential changes in model (mis-)specification.

This paper addresses these problems with a resampling method ("bootstrapping"). The method we present combines the purity of splitting the data into three disjoint sets (including the untouched test set, only used to obtain the final distribution of performance) with the power of a resampling procedure. This allows us to get a better statistical picture of forecast performance, including the ability to estimate the effect of the randomness of initial conditions vs. the randomness of the splits of the data.

All bootstrap methods that have been suggested so far (see [Efron and Tibshirani, 1993] for an introduction, and [Tibshirani, 1994] for the application of standard methods to neural networks in a cross-sectional context) do not split the data in order to evaluate the performance of the model. When working with very noisy data (such as in the financial domain), an evaluation that mixes data for training and testing can lead overly optimistic results.

To demonstrate our method, we wanted to use a data set that lies somewhere between simple noise-free function fitting, and a sequence of true random numbers where no model has a chance. We picked the daily trading volume on the New York Stock Exchange;[1] on this data set, predictions explain about half of the variance. Section 2 of this paper describes the method and the data set, Section 3 presents the empirical results of our study. Conclusions are drawn in Section 4.

---

*http://www.cs.colorado.edu/homes/andreas/public_html/Home.html

[1] Although forecasting prices is a potentially more lucrative target, volume actually is interesting to the economist whose goal is to understand how markets function. The data used here are available by anonymous ftp at ftp.cs.colorado.edu in /pub/Time-Series.

2 Experimental Design

2.1 Bootstrapping Methodology

Randomness naturally enters in two ways in neural network modeling: in the splitting of the data, and in choices about the network initialization, architecture, and training. Since there is no one best split of the data or obvious choice for the initial weights etc., we draw many realizations in order to understand the impact of these choices.[2] CPU intensive bootstrapping methods allow us to evaluate the performance, reliability and robustness of the connectionist approach, and to compare it with linear modeling. Bootstrapping involves generating empirical distributions for statistics of interest through random resampling. We combine bootstrapping along with random network selection and initialization.

A standard procedure for finding a good neural network is to split a time series into three parts: training, cross-validation, and test sets. The training set is used for parameter estimation (in simple backpropagation, by updating the parameter by gradient descent on some cost function). In order to avoid overfitting, a useful procedure is to use a network sufficiently large for the task, to monitor (during training) the performance on the separate cross-validation set, and finally to employ that network that corresponds to the minimum on the cross-validation set for future purposes (such as the evaluation on the test set). The usual procedure fixes these sets. Statistics can be estimated in the test set, but this leaves one question wide open: *What is the variation in performance as we vary training, cross-validation, and test sets?* Furthermore, the usual procedure doesn't tell us how stable the performance is with respect to network parameters. In this paper, we will vary both the data partitions and network parameters in order to find out more about the distributions of forecast errors.

In our example, we have some 6200 patterns, each made up of a few past values of a number of time series (for details of how the patterns are constructed, see Section 2.2 below). We first build the test set by randomly picking 1500 patterns with replacement. The patterns used in this specific test set are then removed from the pool. From the remaining patterns, we then randomly set aside 1500 patterns as the cross-validation set (these are picked without replacement, and also removed from the pool). The remaining patterns then constitute the training set.

We use simple, fully connected feedforward networks with one hidden layer of tanh units and a linear output unit. However, in order to include variations over reasonable choices for network and learning parameters, a number of network characteristics are also draw randomly at the beginning of each run.[3] The cost function is the squared difference between network output and target, summed over all patterns in the training set. Most results are given in terms of "$(1 - R^2)$", i.e., one minus the squared correlation coefficient between forecast and target. Figure 1 also gives the normalized mean squared error, NMSE, i.e., the mean squared error divided by the overall variance of the target.[4]

2.2 Data Set

Our forecasting target is daily total trading volume on the New York Stock Exchange (NYSE). We believe that this series adds two interesting dimensions to forecasting in a nonlinear setting. First, while many papers have tried neural network approaches to forecasting prices, few have attempted forecasting trading volume. Second, volume differs from many other financial series in that it contains more forecastable structure than typical price series.

We use the daily measure of aggregate turnover on the NYSE which is total volume divided by shares outstanding, or the fraction of shares traded that day. This series is not stationary. We detrend it by dividing by a 100-day moving average of past turnover. In other words, we compare the volume today with the average volume over the last 100 trading days. The distribution of this series is still very skewed. We then take the logarithm to get a less skewed distribution.[5] We refer to this transformed series as v_t. Another input is the first difference of the logarithm of the level of the Dow Jones Industrials Index as

[2] We here bootstrap *pairs*. An alternative is to bootstrap *residuals*. That method is less appropriate here, since we are not interested in global error bars but in the variation due to different subsets of such input-output pairs.

[3] In detail, the network architecture is chosen uniformly over 2 to 6 hidden units. The learning rate is chosen uniformly over $[1, 20] \times 10^{-4}$, no momentum. The weight-range w of the initial weights is drawn between $[0.25, 2.5]$. The individual weights are then initialized randomly from a uniform distribution over $[w/i, -w/i]$ where i is the number of connections coming into a unit ("fan-in"). The batchsize (how many patterns are presented until the weights are updated) is drawn uniformly from $[20, 180]$. All inputs are scaled to have zero mean and unit variance as estimated over the entire data set. No significant correlation was found between performance and these parameters. We also tried conjugate gradient in training; the results do not differ from fixed stepsize gradient descent presented here.

[4] The two measures are usually fairly close. For example, let $d_t = \rho y_t + \epsilon_t$, where d_t is the desired value (target), y_t is the forecast, and ϵ_t is independent forecast error. If the means and variances of x_t and d_t are close, then ρ can be approximated by the correlation between forecast and target. Also, for the variances we would get, $\sigma_d^2 = \rho^2 \sigma_y^2 + \sigma_\epsilon^2$. Finally, assuming that the variances of d_t and y_t are equal gives $1 - \rho^2 = (\sigma_\epsilon^2)/(\sigma_d^2)$, or in our notation $1 - R^2$.

[5] In principle, the match ought to be between the "error model" (i.e., exp{cost function}) and the errors of the series (i.e., the residual errors after forecasting, as opposed the unconditional distribution of the series). For these noisy data, however, where we can only hope to predict about half of the variance, the errors and the series itself are quite correlated. This motivates our choice of taking the logarithm of the detrended volume as target values. This can also be viewed as suppressing the importance of very large changes and suggesting to the model to pay more attention to small changes.

a measure of stock returns, r_t. We use daily data from December 3rd, 1962, through September 16th, 1987, corresponding to 6230 days.[6]

Volume movements are connected to stock return movements in interesting ways (e.g., [Gallant *et al.*, 1993], [Karpov, 1987], [LeBaron, 1992]). One of these features is that volume is related to stock price volatility, the absolute magnitude of daily price movements. Also, volume tends to be higher in rising markets. For these reasons we chose several lagged returns and volume variables as predictors. The predictor vector (i.e., the 12 values presented to the network as inputs for each pattern) is given by

$$\{v_{t-1,2,3}, r_{t-1,2,3}, |r_{t-1,2,3}|, \log(\sigma^2_{t-1,2,3})\}.$$

Here, σ_t is another estimate of return variability. It is defined recursively as

$$\sigma_t^2 = \beta\sigma_{t-1}^2 + (1-\beta)r_t^2 \qquad \text{with } \beta = 0.9,$$

similar to variance estimates from autoregressive conditional heteroskedastic models often used in financial time series and summarized in [Bollerslev *et al.*, 1990].

3 Empirical Results

3.1 Learning Curves and Overfitting

Figure 1 shows the set of learning curves for a typical run, for the three sets, both in terms of $(1 - R^2)$ and the normalized mean squared error. Although similar, the correlation coefficient corrects for the case that the network output and desired value have different means, whereas the squared error does not and is thus higher than $(1 - R^2)$ (see also footnote 4).

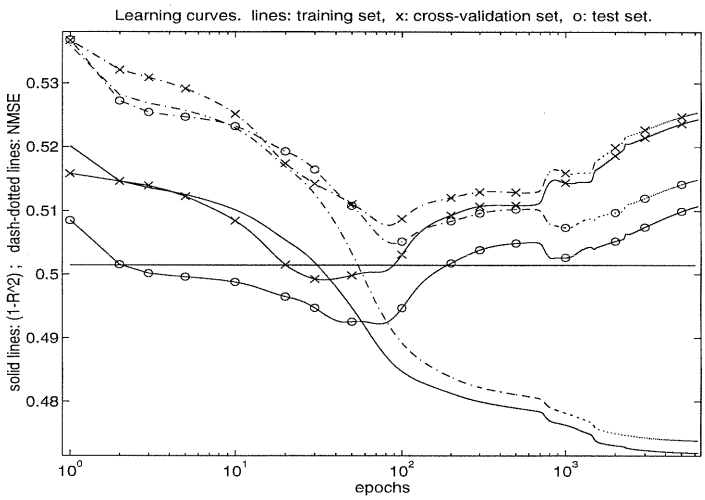


Figure 1: Learning curves of one specific network on one specific split. There are three pairs of curves. The first pair (monotonically decreasing) gives the performance on the training set, the second pair (denoted by "×") on the cross-validation set, and the third pair ("o") on the test set. The three solid lines plot the $(1 - R^2)$ measure; the three dash-dotted lines give the normalized mean squared error (NMSE).

Note the clear increase of cross-validation and test errors after passing through minima (overfitting). Around epoch 800, for example, the network extracts a feature of the training set that helps the test set, but hurts the cross validation set. Note also that the minima of the cross-validation set and the test set do not occur at the same epoch. To be methodologically clean we use, for each run, the performance value on the test set at that epoch that has the minimum of the cross-validation set. The straight line gives the test performance of a linear model estimated on the union of training and cross-validation sets.

3.2 Linear vs. Nonlinear Comparison

One of the most important goals of any exploration of a nonlinear forecasting method is to demonstrate improvements over linear forecasts. Traditionally, one or a few networks are trained on one split of the data. We will see in this section that results obtained in this manner can be misleading; on our data, such a network can easily be a few percent better, but also a few percent worse than the linear model, depending on the split. Thus, instead of just comparing forecasts in one out-of-sample time period, we bootstrap the distributions of forecast performance for both the neural network and linear forecasts. This allows a reliable statistical comparison of the two models.

[6] A "super test set" (the period from September 17th through October 19th, 1987 that contains the 1987 crash) is set aside for some final out-of-sample forecasting experiments, described in Section 3.4.

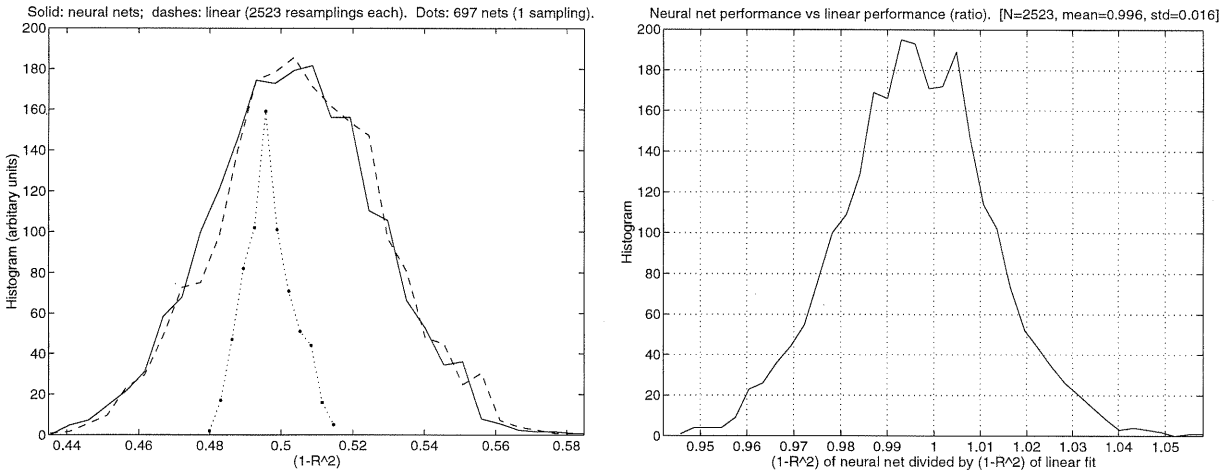


Figure 2: (a) [left]: Histograms of $(1 - R^2)$ forecast performance. The solid line shows the distribution of the networks, the dashed lines of linear model, both estimated on 2523 different resamplings of the available data. The dotted line takes just one split of the data and describes the distribution of 697 networks. The fact that the width of the dotted histogram is a lot smaller than the width of the other two indicates that the randomness in the splitting of the data generates more variability than the randomness in network initialization does.
Figure 2 (b) [right]: Histogram of the ratio of $(1 - R^2)$-neural network performance divided by $(1 - R^2)$-linear performance for 2523 resamplings.

For comparison we fit for each split a linear model to exactly the same targets and predictors used for the neural network. Parameters are estimated using the union of the same training and cross-validation set, and $(1 - R^2)$ is estimated over the same test set from each bootstrap resampling.

The empirical density from 2523 bootstrap resamplings of the forecast performance is shown in Figure 2a. The sold line and the dashes display the performance of the neural net and the linear models, respectively. It is clear from this picture that distinguishing between the two forecasts is going to be difficult, if possible at all. (When we embarked on this experiment, we were indeed hoping for a clean separation that would have indicated nonlinear structure in this series!) To focus on the comparison, we plot in Figure 2b the run-by-run ratio of the two forecast performance measures. This ratio is estimated for each of the bootstrap samples and recorded. If the neural network were consistently outperforming the linear model then this ratio would be much less than unity. This histogram shows that it is not very likely that the neural network will do better than a linear model in most cases.[7]

3.3 Variability over Random Networks

Our procedure randomizes both over data samples and over network architectures and initial parameters. An important question is: *How much of the variability is due to the data set resampling, and how much is due to the network parameters?* Viewed from a different angle: for a given split, how much overfitting (or "data-mining") would connectionists be likely to engage in were they to optimize their network architecture etc. for that split? If great gains were possible by tinkering with network parameters for each split, we should observe a lot of variability in forecast performance over randomly initialized networks on a given data set split. However, the dotted line in Figure 2a shows a representative density for 697 randomly drawn nets, all trained and tested using the same training, cross-validation, and test sets. *Forecast variation due to changes in network structure is small relative to the sample variation.*

Another perspective on the correlation of forecast performance between neural networks and linear models is given in Figure 3, a scatter plot of the performance of the nonlinear vs. the nonlinear model,

$$\{(1 - R_L^2)^i, (1 - R_{NN}^2)^i\},$$

for each bootstrap sample, i. If the networks are picking up much of the same structure as the linear forecasts, we will see a strong correlation between the two. This is indeed the case as can be seen in Figure 3 where the correlation between forecast errors is 0.936.

The main result for an economist interested in linear vs. nonlinear structure is that models that can in principle express any nonlinear function (feedforward neural networks with tanh hidden units), are not able to extract nonlinear features from the volume series that generalize out-of-sample.

[7] The average ratio in Figure 2b is 0.996 ± 0.016. On the one hand, this is significantly different from 1, with a t statistic of 12.6, indicating a significant, but small improvement in overall forecast performance. On the other hand, when we compare the forecast performance using squared forecast errors, we find that the average ratio (over the same 2523 runs) is larger than unity, 1.003 ± 0.020 (the confidence intervals are statistical errors of one standard deviation).

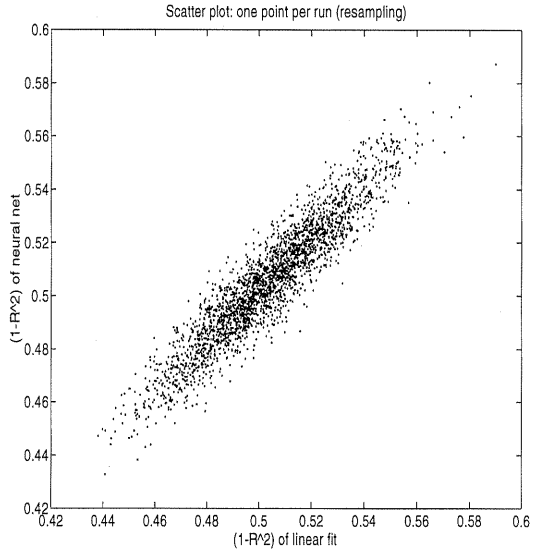


Figure 3: Forecast correlation. For each of the 2523 runs (i.e., different splits of the data) one point is entered into this scatter plot. Its location is determined by the performance of the neural network vs. the linear model. Note that the point cloud is quite elongated, corroborating the finding that there is more variation due to the randomness in data splits than the variation that results from the randomness in the initial conditions.

3.4 Probability Density of the Forecasts

Now that we have an entire ensemble of neural network predictors, we can present in the final part of the paper a new way of showing how all these networks can give us a fresh view on forecast confidence: the idea is to use each of the 2523 networks to make (single-step) predictions on a sample that had been set aside throughout (i.e., never used during training, cross-validating or testing). The time period of this sample starts immediately after the time period considered so far, i.e., it starts on September 17th, 1987, and includes the crash of October 19th, 1987, a day with unusually large price movement and trading volume.

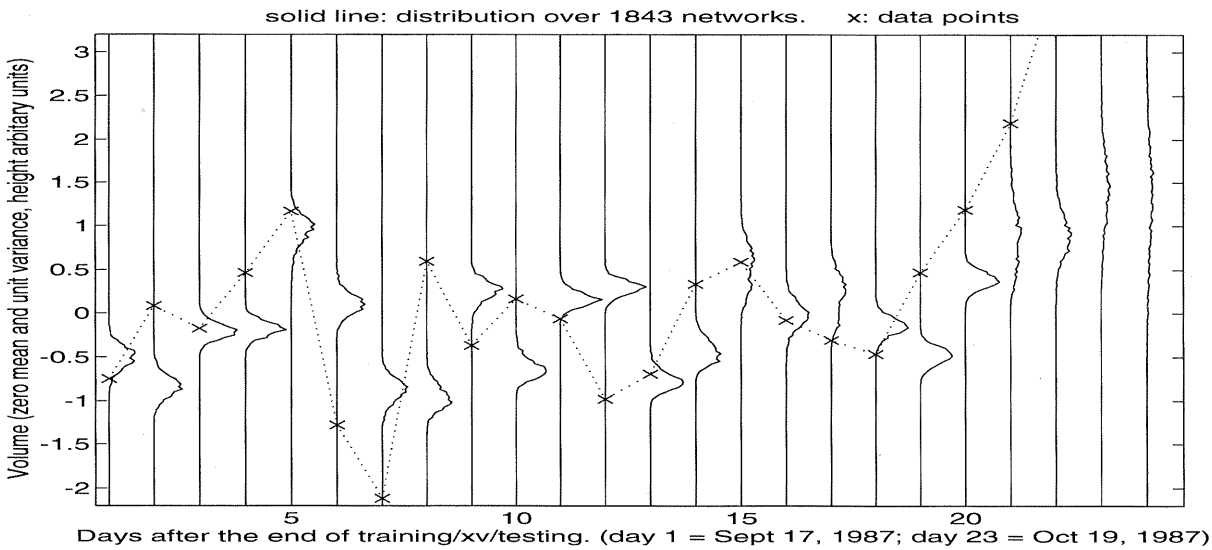


Figure 4: Confidence intervals on individual days on the "super-test-set". The October 1987 stock market crash occurred on Day 23 on this scale; the value for the volume went off scale.

Figure 4 displays, for each day, the density of all the neural network predictions. The actual data points are marked with $\times$. (The data points for the stock market crash, October 19th and 20th, 1989 are missing since they are off the scale.) There are a few interesting features contained in the figure. We see that the predictors in many cases are biased high or low, indicating generally bad forecast performance. (Explaining 50 percent of the variance of the data means that there still remains 50 percent unexplained!) Second, the fact that for many of the days the width of the distribution is quite small suggests that even

the smartest selection or combination of forecasts cannot yield much improvement. Finally, we can see how the models' predictions begin to spread apart as the period of the crash is reached.

Forecast uncertainty can come from a number of sources, such as model uncertainty (which can be further split into uncertainty over parameters and specifications), and stochasticity (e.g., outside shocks, measurement noise).[8] E.g., when approximating a noisy dynamical system, $x_t = f(x_{t-1}, \theta) + \epsilon_t$, forecast uncertainty comes from both the additive noise and the fact that we don't have the true model. In our experiments we are looking at the distribution of forecasts over fitted models and parameters. This gives some feel for the reliability of our forecasting models, but it is not a true "confidence region" about where the process is going to go. A method to estimate true confidence regions that depend on the location in input space ("local error bars") is given by [Weigend and Nix, 1994]. They use a maximum likelihood framework to train a network with two output units, one giving the prediction, the other one the error bar. A method to estimate the probability distribution of the next value—a rather data-hungry enterprise—is described by [Srivastava and Weigend, 1994]. They use "fractional binning" to estimate the conditional density of the stochastic process. Both these methods take all noise sources into account *except* the one considered here.

4 Conclusions

In this paper we demonstrated the usefulness of a new bootstrap approach to generating and testing time series forecasts. The procedure should eventually allow us to find more robust networks that are less sensitive to nonstationarities present in time series. Also, it allows us to be more confident about what we can conclude about the reliability of the forecasts.

We applied our procedure to trading volume. Contrary to our expectation, there was no improvement over linear models. Given the method and a sufficiently large number of simulations, we can be pretty confident that this is a fairly general property for this time series and the class of models we considered. This does not rule out the possibility of forecast improvements using other nonlinear structures or additional forecast variables.

The simulations gave us important insights into the variability of forecast performance over changes in subsamples and network structure. For our example, most of the variability in forecast performance was clearly coming from sample variation and not from model variation. This tells us that for this series there is probably little hope in fine tuning the networks we used.

In summary, when examining relatively noisy time series we feel that procedures such as bootstrapping are extremely useful in getting a clearer picture of what is real and what is noise.

Acknowledgements

Andreas Weigend acknowledges support from the National Science Foundation under Grant No. RIA ECS-9309786, and discussions with William Finnoff and Rob Tibshirani. Blake LeBaron acknowledges support from the Alfred Sloan Foundation, and the Wisconsin Alumni Research Foundation.

References

[Bollerslev *et al.*, 1990] T. Bollerslev, R. Y. Chou, N. Jayaraman, and K. F. Kroner. ARCH modeling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, 52(1):5–60, 1990.

[Efron and Tibshirani, 1993] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap.* Chapman and Hall, 1993.

[Gallant *et al.*, 1993] A. R. Gallant, P. E. Rossi, and G. Tauchen. Nonlinear dynamic structures. *Econometrica*, 61:871–908, 1993.

[Karpov, 1987] J. M. Karpov. The relation between price changes and trading volume: A survey. *Journal of Financial and Quantitative Analysis*, 22:109–126, 1987.

[LeBaron, 1992] B. LeBaron. Persistance of the Dow Jones index on rising volume. Technical report, University of Wisconsin - Madison, Madison, Wisconsin, 1992.

[Srivastava and Weigend, 1994] A. N. Srivastava and A. S. Weigend. Computing the probability density in connectionist regression. In *IEEE International Conference on Neural Networks, Orlando, FL.* ICNN, 1994.

[Tibshirani, 1994] R. Tibshirani. A comparison of some error estimates for neural network models. Technical report, Dept. of Preventive Medicine and Biostatistics, University of Toronto, Toronto, Ontario, 1994.

[Weigend and Nix, 1994] A. S. Weigend and D. A. Nix. Predictions with confidence intervals (local error bars). Technical report, CU–CS–724–94, Computer Science Department, 1994.

[Weigend *et al.*, 1990] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.

[8] Other sources, important in nonlinear dynamical systems with low noise, such as the divergence of nearby trajectories, are less important in the present case of noisy financial data.