Letters

A Bootstrap Evaluation of the Effect of Data Splitting on Financial Time Series

Blake LeBaron and Andreas S. Weigend

***Abstract*—This letter exposes problems of the commonly used technique of splitting the available data into training, validation, and test sets that are held fixed, warns about drawing too strong conclusions from such static splits, and shows potential pitfalls of ignoring variability across splits. Using a bootstrap or resampling method, we compare the uncertainty in the solution stemming from the data splitting with neural-network specific uncertainties (parameter initialization, choice of number of hidden units, etc.). We present two results on data from the New York Stock Exchange. First, the variation due to different resamplings is significantly larger than the variation due to different network conditions. This result implies that it is important to not over-interpret a model (or an ensemble of models) estimated on one specific split of the data. Second, on each split, the neural-network solution with early stopping is very close to a linear model; no significant nonlinearities are extracted.**

***Index Terms*—Bootstrap, financial forecasting, linear bias of early stopping, model evaluation, model merging, model uncertainty, resampling, superposition of forecasts, time series prediction.**

I. Introduction

Training a network on a time series is not hard, but once we have a network, how much can we trust the forecasts for truly new data? On the one hand, if the time series is fairly long (above a few thousand points), and if it is fairly clean (noise of less than 1% of the signal), the evaluation of a model is relatively easy, since only very few functions will fit some held-back data very well. This regime can be described as a "right-with-probability-$(1-\epsilon)$-regime." On the other hand, for very noisy and/or very short time series, one can only hope to be right on new data with a probability of $(0.5+\epsilon)$. An example would be the forecast of the direction of a stock price movement. It is well known that random predictions, or random trading strategies, can yield deceptively long sequences of good predictions or profitable trades. In such noisy problems, many functions will be indistinguishable in their forecasting quality. 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. Evaluating a model for noisy time series can be more work than estimating the parameters.

A standard procedure for evaluating the performance of a model is to split the data into one training set (used for the parameter estimation, e.g., through gradient descent or second-order methods), one validation set (used to determine the stopping point before overfitting occurs, and/or used to set additional parameters or hyperparameters, such as the importance given to penalize model complexity), and one or more test sets. This procedure has been used for many years in the connectionist community, see, e.g., [24]. Our more recent experience has found this approach, along with conclusions drawn from it, to be very sensitive to the specific splitting of the data. Therefore, usual tests of forecast reliability can easily be overly optimistic.

This article addresses these problems with a bootstrap method. The approach we present combines the purity of splitting the data into three disjoint sets with the power of a resampling procedure, giving a better statistical picture of forecast variability, including the ability to estimate the effect of the randomness of the splits of the data versus the randomness of initial conditions of the network.

This is not the first article that uses the bootstrap in a connectionist context. Weigend *et al.* [25] used the *bootstrapping of residuals* to evaluate the forecasting power of a neural net for exchange rate forecasts, and Connor [6] also bootstrapped residuals to obtain error bars for the iterated time series predictions. The goals were different from the goal of the work reported here. In this article we *resample pairs* which will be clarified in Section II-A. Resampling pairs was first suggested by Efron [8], and first used in the connectionist community by Paass [20] on the example of noisy exclusive OR. Tibshirani [21] applied the bootstrap machinery to networks in a cross-sectional context. However, none of these articles evaluate the effect of using the common, simple, static sample split on the performance reliability.

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[1] on the New York Stock Exchange, where predictions can explain about half of the variance. Section II of this article describes the method and the data set, Section III presents the empirical results of the study, Section IV discusses other sources of uncertainty not captured by the bootstrap, and Section V draws some conclusions.

II. Experimental Design

A. Bootstrapping Methodology

Randomness enters naturally in two ways in neural-network modeling: in the splitting of the data, and in choices about the network initialization, architecture, and training. A standard procedure for finding a good network is to split the patterns derived from a time series into three sets: training, 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 common procedure is to use a network sufficiently large for the task, to monitor (during training) the performance on the separate validation set, and finally to choose the network that corresponds to the minimum on the validation set, and employ it for future purposes such as the evaluation on the test set. These sets have no patterns in common. The usual procedure fixes these sets. As many statistical quantities as desired can be estimated in the test set, but this leaves one question wide open: *What is the variation in performance as we vary training, validation, and test sets?* This is an important question since real-world problems do not come

Manuscript received July 11, 1994; revised October 18, 1997. B. LeBaron was supported by the Alfred Sloan Foundation and the Wisconsin Alumni Research Foundation. A. S. Weigend was supported by the National Science Foundation and from the Air Force Office of Scientific Research.

B. LeBaron is with the Department of Economics, University of Wisconsin, Madison, WI 53713 USA.

A. S. Weigend is with the Department of Information Systems, Leonard N. Stern School of Business, New York University, New York, NY 10012 USA.

Publisher Item Identifier S 1045-9227(98)01048-0.

[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.

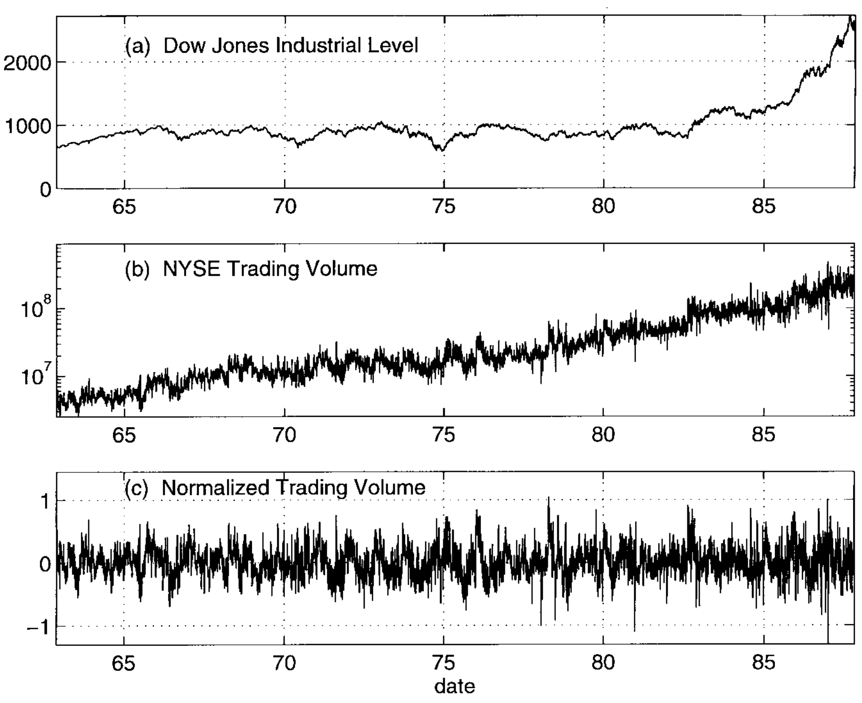

Fig. 1. (a) The level of the Dow Jones Industrial Average from December 1962 to October 1987. (b) The raw trading volume (aggregate turnover) on the NYSE. The nonstationarity is evident; this is a semilogarithmic plot. (c) The series v_t that we use as target: it is obtained by taking the logarithm of the raw value and dividing it by the mean of the last 100 trading days.

with a tag at each pattern saying how it should be used! Also, if we were to only train one network on such a split, this would not tell us how stable the performance is with respect to network choices.

Since there is not just one "best" split of the data or obvious choice for the initial weights etc., we will vary both the data partitions and network parameters in order to find out more about the distributions of forecast errors. We use a computer intensive bootstrapping method 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.

In more detail, in order to understand the impact of the splitting and network choices, we draw a realization of splits and network conditions, and train a complete model on this realization. This is sometimes called bootstrapping *pairs* [7], since the input–output pairs or patterns remain intact, and are resampled as full patterns. This can be contrasted with training one model only, and resampling the errors of that one model to obtain a distribution, called bootstrapping *residuals.* The latter method was used in single-step prediction by Weigend *et al.* [25] in the context of foreign exchange rate predictions. One model was built on one split of the data. Similarly, in an application to load forecasting, Connor [6] trained one single-step prediction network on one split of the data, then resamples from the empirical distribution of the single-step errors and adds these to the inputs in order to obtain estimates of the errors of iterated forecasts. In this residuals bootstrap, the residuals obtained from one specific model are used in rebuilding pairs or patterns to obtain error bars reflecting all sources of error, including model misspecification. In contrast, here we are interested in variation due to sample splits rather than error bars. Every "run" has a different assignment between the sample patterns and the three sets which thus are different for each run.

In the example used in this article, 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 II-B 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 validation set (these are picked without replacement, and are also removed from the pool). The remaining patterns then constitute the training set.[2] For the results presented in the article, we do this 2523 times, training a network each time.

We use 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 drawn randomly at the beginning of each run.[3] The cost function is the squared difference between the network output and the target (expressed as the log-transformed volume, detailed in the next section), 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.

[2]There is no deep theoretical justification for drawing the test data with replacement, and the training and validation set effectively without replacement. Our motivation was to stick to the standard rule of sampling with replacement for the test set. For the training and validation sets, we did not allow for repeated patterns since we wanted the linear fit comparison to be estimated on each nontest data point with even weight, and wanted to use identical sets for the net and the linear fit in each run.

[3]In detail, the network architecture is chosen uniformly over two–six 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 block-size (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 any of these choices.

B. Data Set

We use daily data from the New York Stock Exchange (NYSE) from 3 December, 1962, through 16 September, 1987, corresponding to 6230 days.[4] Our forecasting goal is daily total trading volume, shown in Fig. 1. We believe that this series has two interesting features: First, while many articles 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, Fig. 1(b). 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 obtain a less skewed distribution.[5] We refer to this transformed series as v_t. This target series is shown in Fig. 1(c).

Beside three lagged values of v (a typical autoregressive or AR model), we use three other sets of variables, making it an exogenous or ARX model. We use first differences of the logarithm of the level of the Dow Jones Industrials Index as a measure of relative stock returns, r_t. Furthermore, volume movements are connected to stock return movements in interesting ways ([12], [15]; [10]. One of these features is that volume is related to stock price volatility, sometimes approximated by the absolute magnitude of daily price movements. Furthermore, 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 an estimate of a volatility. It is defined recursively as

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

This represents an exponential filter of the squared returns. This can be interpreted in physical terms as a relaxator: A shock in r_t^2 decays in $-1/\log\beta = 9.5$ days to $1/e = 0.37$ times its initial value.[6] We initialize σ_0^2 to the unconditional variance of the series. The choice of the exponentially smoothed squared returns is motivated by the similarity to variance estimates from autoregressive conditional heteroskedastic (ARCH) models often used in financial time series ([2], [3]).

Summarizing, we use the following inputs for our model.

- Three lags of the past trading volume, $v_{t-1,2,3}$. They are normalized by the 100-day moving average (but not differenced), see Fig. 1(c). Their one-day autocorrelation after normalization is 0.66. (Without our normalization, i.e., taking the raw volume from Fig. 1(b), the overall shift in level over the two decades is responsible for an autocorrelation of 0.95.).
- Three lags each of the relative returns, $r_{t-1,2,3}$. Their one-day autocorrelation is small (0.135), and disappears for two or more lags, as discussed in [15].
- Two estimates of their volatilities, with three lags each:
  a) Absolute value of the relative returns, $|r_{t-1,2,3}|$. Their autocorrelation coefficients are dropping off very slowly, and have values for the first ten lags around 0.16, computed after subtracting the mean of $|r_t|$.[7]
  b) Logarithm of the exponentially smoothed squared returns, $\log(\sigma^2_{t-1,2,3})$. Their one-day autocorrelation is 0.975. It drops off very slowly, primarily due to the smoothing (each value reenters at the next time step attenuated by $\beta = 0.9$), and secondarily due to the already existing autocorrelation of the driving process of r_t^2.

We refer to each of these 12-dimensional predictor vectors with the associated one-dimensional target value as *a pattern.* The correlation coefficients were computed through 19 October, 1987, i.e., excluding the effect the day of the 1987 crash would have. As shown, some of the input dimensions are highly correlated. Despite this high correlation that gives an effective overlap of the patterns in the three sets, we will see in the next section that the performance varies a lot for different random samples out of these overlapping patterns.

III. Empirical Results

A. Learning Curves and Overfitting

Fig. 2 shows the set of learning curves[8] for a typical run, for the three sets, both expressed as one minus the correlation coefficient squared, $(1 - R^2)$, and as the mean squared error divided by the overall variance of the target, NMSE. Differences between these two reasonable performance measures occur when the mean and the variances are not estimated correctly. Whereas the correlation coefficient corrects for these differences (by subtracting the means and dividing by the standard deviations), the squared error does not, and is thus higher than $(1 - R^2)$.

The learning curves in Fig. 2 show performance versus the number of backpropagation iterations. There is a clear increase of validation and test errors after passing through minima, usually called overtraining or overfitting. At some stage (around epoch 800 in this specific run which happened to have a very small learning rate) the network extracts a feature of the training set that helps the test set, but hurts the validation set. The minima of the validation set and the test set do not occur at the same epoch. From each of these sets of learning curves, only a single number is used for the subsequent analysis and comparisons in this letter: the performance value on the test set at that epoch that has the minimum of the validation set.

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

[5] Normalizing with the 250-day mean (of the last trading year) did not remove quite enough of the nonstationarity. Note also that we are using the normalized level of the volume, not a difference version that would correspond to the change in volume. Apart from correcting somewhat for the skewed distribution, the logarithm can be interpreted as emphasizing small values of the volume more than large ones, and, alternatively, as facilitating product interactions between lagged values of the volume, since the inputs are added in the argument of the hidden units, and adding logarithms corresponds to multiplying the original values.

[6] An equivalent box-cart moving average would average the squared returns over 19 days. We chose the exponential average since it does not exhibit the box cart's shadows, i.e., the effect that large shocks show up again with the opposite sign once they drop off the left side of the window.

[7] The $1/e$ decay time of the corresponding AR process is about half a time step. This does not characterize the time scale of the underlying process well: the coefficient is severely underestimated due to the presence of noise in the inputs ("errors-in-variables," see [9] and [5]). Fitting a state-space model with full dynamics to the series $\{|r_t| - \langle|r_t|\rangle\}$ gives an autoregressive coefficient for the dynamics of the state of 0.9915, corresponding to an $1/e$ decay time of approximately five months (117 trading days). For more details of this method and their interpretation for modeling volatilities see [22].

[8] We use the term learning curve to characterize the performance as a function of the iterations of the algorithm. In a different context, typically when an arbitrary number of training patterns can be generated, the term learning curve denotes performance as a function of data set size.

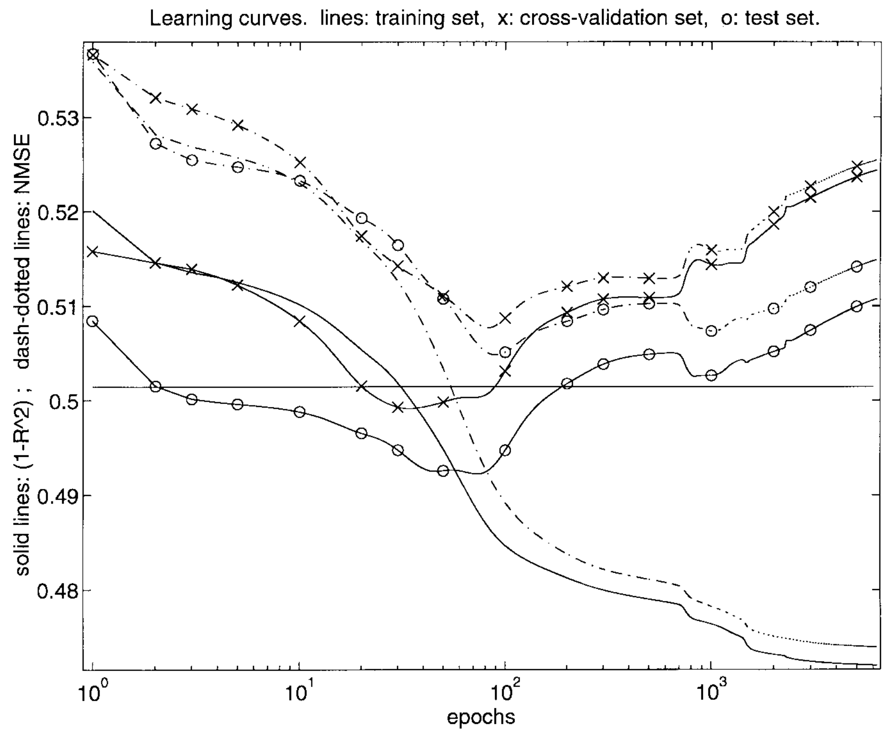

Fig. 2. Learning curves of one specific network on one specific split. They show the performance versus the number of backpropagation iterations. There are three pairs of curves. The first pair (monotonically decreasing) gives the performance on the training set, the second pair (denoted by "$\times$") on the 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). The straight line indicates the test error of a linear model estimated on the union of the training set and the validation set.

B. Linear versus Nonlinear Comparison

One of the most important goals of any exploration of a nonlinear forecasting method is to demonstrate an improvement over linear forecasts. For synthetic data, generated from nonlinear noise-free systems, forecast improvements of several orders of magnitude have been reported: consider the celebrated logistic map which consists only of a second-order component (quadratic term) without any first-order (linear) component. It really should come as no surprise that methods that allow for nonlinearities will vastly outperform the perfectly inadequate linear fit in cases when there is no linear component.

We here focus on high-noise real-world data where the evaluation is much harder, and potential nonlinearities are often masked by noise. In this case, great care needs to be taken to evaluate the nonlinearity of the model: obtained on a single split, depending on the split, a network can easily be a few percent better, but also a few percent worse than the linear model. Thus, instead of just comparing forecasts on one split and one out-of-sample time period, we recommend bootstrapping and reporting the distribution of forecast performance for both the network and linear forecasts. This allows a more meaningful statistical comparison between linear and neural-network models.

In this comparison we fit for each split a linear model to exactly the same patterns (inputs and targets) used for the network. Parameters are estimated using the union of the same training and validation set, and $(1 - R^2)$ is estimated over the same test set from each bootstrap resampling.[9]

The empirical density from 2523 bootstrap resamplings of the forecast performance is shown in Fig. 3. The solid line displays the performance of the networks, and the dashes that of the linear models. It is clear from this picture that distinguishing between the two forecasts is going to be difficult, if possible at all.

To focus on the comparison, Fig. 4 shows a histogram of 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 networks were consistently outperforming the linear models then this ratio would be less than unity. However, this histogram shows that it is not very likely that the network will do better than a linear model in most cases.[10]

Another perspective on the correlation of forecast performance between neural networks and linear models is given in Fig. 5, a scatter plot of the performance of the linear versus the nonlinear model

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

One point is entered 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 in Fig. 5 where the correlation between forecast errors is 0.936.

To summarize this section: When we embarked on this experiment, we were hoping for simple clean evidence for nonlinear structure in

[9]One referee suggested a comparison with an autoregressive moving average (ARMA) models instead of the exogenous autoregressive linear (ARX) we use. ARMA models are indeed more general linear models than AR models. However, for the pairs-bootstrap study presented here, where resampling destroys the sequence of the patterns, it is not possible to feed back errors (the MA part). An ARMA model cannot be used in combination with the pairs-bootstrap; ARX is thus the appropriate linear model class to compare to.

[10]The average ratio in Fig. 4 is 0.996 $\pm$ 0.016. On the one hand, this is significantly different from one, 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 $\pm$ 0.020 (the confidence intervals are statistical errors of one standard deviation). This leads us to the conclusion that there is no relevant difference between the nets and the linear fits.

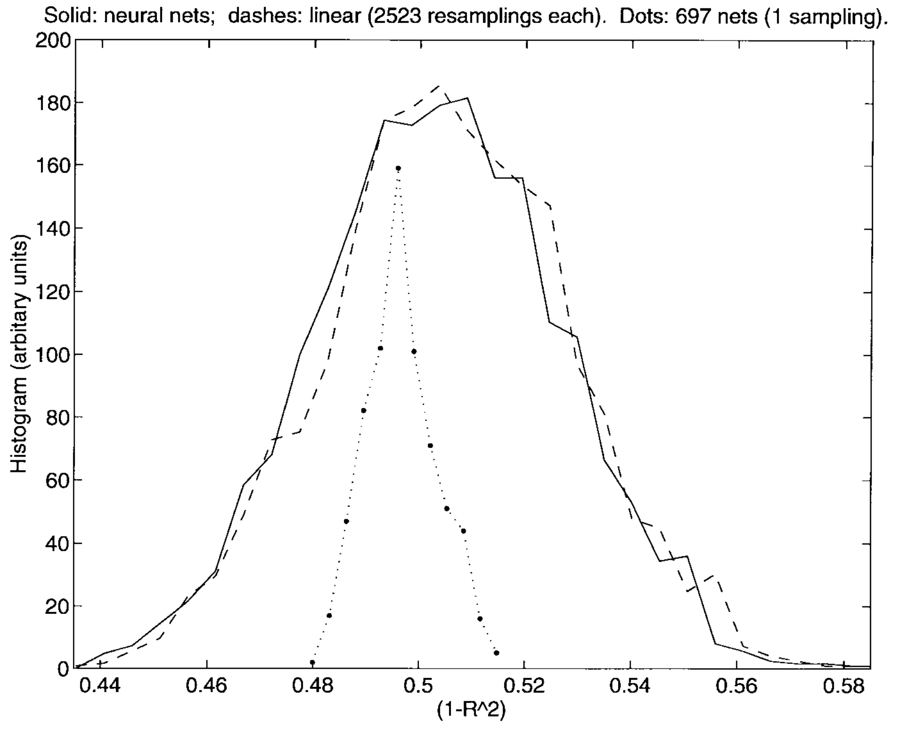

Fig. 3. Histograms of $(1 - R^2)$ forecast performance. The solid line shows the distribution of the networks, the dashed lines of linear model, both estimated on 2523 different resamplings of the available data. The dotted line takes just one split of the data and describes the distribution of 697 networks. The fact that the width of the dotted histogram is clearly 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.

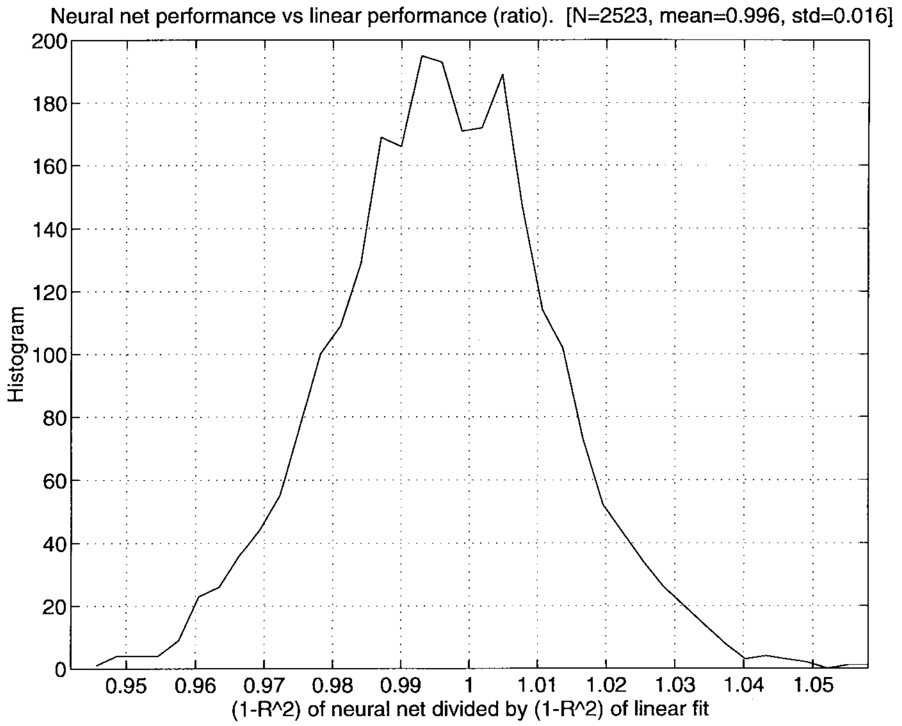

Fig. 4. Histogram of the ratio of $(1 - R^2)$ network performance divided by $(1 - R^2)$-linear performance for 2523 resamplings. Each entry in this histogram corresponds to the performance ratio of one network and one linear model trained on one specific split.

the volume of the NYSE, of high interest for economists. What we found instead is that possible underlying nonlinearities are not easily discovered—using a model class celebrated for its ability to express any nonlinear function (feedforward networks with $\tanh$ hidden units with a squared error cost function) did not reveal such structure. Since this article focuses on the variation due to different splitting of the data, we did not use explore alternatives to early stopping that avoid the bias toward a linear solution, such as weight-elimination or pruning; those are interesting experiments and the data is available from the authors' web sites. Furthermore, we did not use computer

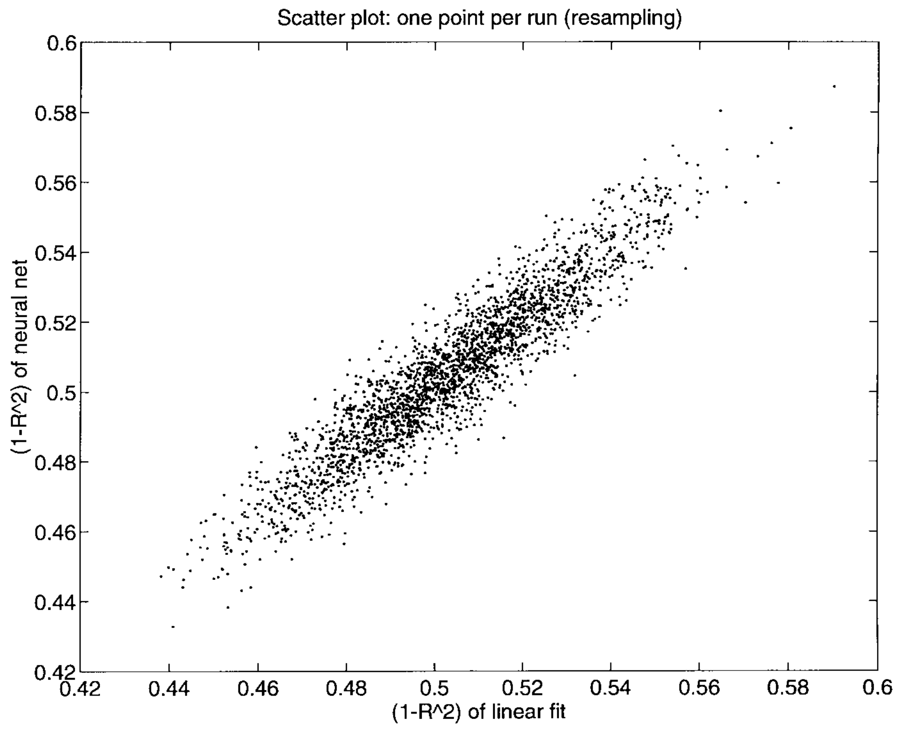

Fig. 5. Scatter plot of the prediction errors. For each of the 2523 runs (i.e., different splits of the data) one point is entered into this scatter plot. Its location is determined by the performance of the network versus the linear model. Note that the point cloud is much more stretched out along the 45 $^\circ$line than orthogonal to it, again indicating that there is more variation due to the randomness in data splits than the variation that results from the randomness in the initial conditions.

generated nonlinear data, since generating an arbitrarily large number of noise-free data points of an ergodic system will typically (for any split of the data) give very close neighbors between the different sets and not constitute a serious test for the real-world problem of noisy data of finite record length, that we typically find in economics, finance, and business, perhaps slightly nonlinear.

C. 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 model overfitting 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 Fig. 3 shows a representative density for 697 randomly drawn nets, all trained and tested using the same training, validation, and test sets. The answer to the question is: *The variations of the forecasts due to changes in network structure are small relative to the variations due to sample splitting.*

D. Probability Density of the Forecasts

Now that we have an entire ensemble of neural-network predictors, we can investigate how all these networks can give us a fresh view on the old idea of combining forecasts by looking at the scale of the variations compared to the noise inherent in the problem. We use each of the networks to make predictions on a sample that had been set aside throughout (i.e., never used during training, validating, or testing). The time period of this sample starts immediately after the time period considered so far, i.e., it starts on 17 September, 1987, and includes the crash of 19 October, 1987, a day with unusually large price movement and trading volume.

Fig. 6 displays, for each day, the density of all the network predictions. They are single-step forecasts: the input for tomorrow's prediction contains today's observed values. The solid lines are the histograms of the individual raw predictions; they have not been convolved with any added uncertainties. The actual data points are marked with $\times$. (The data points for the stock market crash, 19 and 20 October, 1989, are missing since they are off the scale.) A few interesting features are contained in the figure. First, we see that the forecasts in many cases are biased high or low, indicating generally mediocre forecast performance. (Explaining 50% of the variance of the data means that there still remains 50% unexplained!) Second, the fact that for many of the days the width of the distribution is quite small and quite far away from the actual value suggests that even the smartest selection or combination of forecasts cannot yield much improvement.[11] Finally, we can see how the models' predictions begin to spread apart as the period of the crash is reached. The main reason for this spreading is that the inputs wander off into regions where the network has never seen training points. Regression neural networks do not spend any resources on modeling the density of the inputs—moving away from region of interpolation to extrapolation manifests itself indirectly through deteriorating performance. Thanks to the benevolent nature of $\tanh$ hidden units, the output remains bounded even for thus far unexplored regions.

[11]The idea of combining of forecasts [1], based on the idea that superposition helps to the degree that the errors are uncorrelated, has recently reached the connectionist community, see, e.g., [11]. This article presents, on a practical example, the limitations of averaging for noisy data: the empirical densities show that averaging over all the splits we did (by taking the mean, median or any convex, possibly even adaptive, combination of the 1843 individual models) will not improve the predictions dramatically.

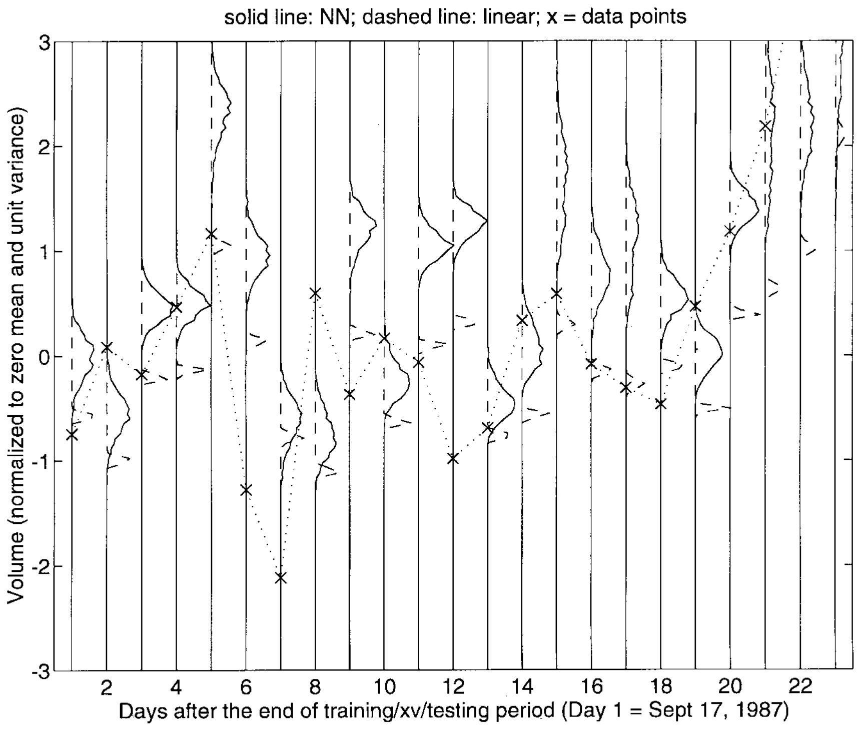

Fig. 6. One-day ahead densities for the days on the "super-test-set." The October 1987 stock market crash occurred on day 23 on this scale; the value for the volume went off scale.

IV. Relation to Other Sources of Uncertainty

Forecast uncertainty can come from many sources. We focused on the uncertainty obtained from the specific splits, that can be called splitting uncertainty. In the larger picture, its size is relatively small compared to all the noise sources that contribute to the normalized squared error of about 50%, or a correlation coefficient of about

$$R = \sqrt{(1-0.5)} = 0.7.$$

We here briefly describe the effect of other sources of uncertainty:[12]

- Noisy targets. An appropriately trained network outputs expected values. Gradient descent in a squared error cost function can be interpreted in a maximum likelihood framework as the observed values being normal distributed around the predicted values with constant noise level. This assumption can be relaxed, first by allowing a Gaussian with locally varying widths, then by modeling the output distribution with potentially multimodal functions:

  a) Heteroskedasticity ("local error bars"). Nix and Weigend [19] described a method to train a network with two output units, the first giving the prediction, the second the error bar. Those two numbers, both functions of the input space, parameterize a Gaussian and can be used for unimodal densities. This method is more flexible than the constant variance assumption, but not appropriate for multimodal output densities.

  b) The assumption of a single Gaussian can be generalized to a mixture of Gaussians that allow prediction of more general densities. Jacobs *et al.* [11] introduced Gaussian mixture models to the connectionist community, and Weigend *et al.* [25] applied them to time series prediction. As an alternative, rather than using this mixture of Gaussians with varying centers, Weigend and Srivastava [23] introduced a fuzzy-logic like superposition of tent-functions at fixed centers to model potentially multimodal densities.

- Noisy inputs (observational noise). This important noise source in autoregression of noisy time series is well known in statistics and econometrics (see Section II-B) but less well known in the connectionist community [27]. If the levels of the noise for each input is known, the effect can always be emulated with a Monte Carlo simulation of forward passes with slightly different values for the inputs in order to build a density reflecting input fluctuations.
- Parameter noise (uncertainty in the weights). From a Bayesian perspective, model parameters are never known exactly but also have some uncertainty that translates into an uncertainty of the prediction ([4], [17] [18]). While an analytic approximation is only available for simple cases, it is always possible to obtain a distribution by generating forward passes through networks with slightly different weight values.
- Regions of low input density (extrapolation). At the end of the Section III-D we discussed the uncertainty for the 1987 crash stemming from too few patterns in the vicinity of the pattern for which the prediction is to be made. For the data set used in this paper this is not an important source here since adjacent input patterns are highly correlated, implying that in most cases the network will have encountered nearby neighbors in the training set. This yields, however, to an overly optimistic interpretation of the performance.

The specific values of the prediction performance should not be overinterpreted. As typically done in cross-sectional bootstraps, we pick the validation and test patterns interspersed with the training

[12] 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.

data in order to obtain an indication of the variation of the subsample selection. Care has to be taken, however, in interpreting the results as accurate estimates of the generalization performance for truly future data. If there is a strong overlap from one pattern to the next (imagine a problem where all inputs are highly smoothed, like the exponentially filtered volatility estimate we use, or, even without smoothing, if the data is sampled with a frequency a lot faster than the dynamics of the system!) the chances are high that for a given test pattern, there will be very similar training patterns adjacent in time. In this case, more accurate estimates of the performance on future data might be obtained by bootstrapping blocks of data ([13], [16]). Note, however, that these blocks are still taken from the entire period. So, if the dynamics is truly nonstationarity, this blocks-bootstrapping will still give overly optimistic results. To avoid fooling oneself on financial data, we strongly recommend using only data for testing that arrived after the end of the training and validation period (whether these two are interspersed, blocked, or sequential).

V. Conclusions

This article demonstrated the usefulness of a pairs bootstrap approach to generating and testing time series forecasts. We then applied the procedure to trading volume. Contrary to our expectation, no improvement over linear models could be obtained with a standard network trained with backpropagation and regularized by early stopping. This does not rule out the possibility of forecast improvements using additional forecast variables, or by using pruning and weight-elimination techniques.

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. This is an example of an application where we feel that procedures such as bootstrapping are extremely useful in getting a clearer picture of what might be real and what is noise.

Acknowledgment

Andreas Weigend acknowledges the hospitality of the Wirtschaftswissenschaftliche Fakultät der Humboldt Universität Berlin during summer 1994. The computation of the time constant of the volatility persistence of this data set using a fully dynamic state-space model is joint work with J. Timmer; see [22].

References

[1] J. M. Bates and C. W. J. Granger, "The combination of forecasts," *Operations Res. Quarterly*, vol. 20, pp. 451–468, 1969.
[2] T. Bollerslev, R. Y. Chou, N. Jayaraman, and K. F. Kronner, "ARCH modeling in finance: A review of the theory and empirical evidence," *J. Econometrics*, vol. 52, no. 1, pp. 5–60, 1990.
[3] T. Bollerslev, R. F. Engle, and D. B. Nelson, "ARCH models," in *Handbook of Econometrics*, vol. 4. New York: North-Holland, 1995.
[4] W. L. Buntine and A. S. Weigend, "Bayesian backpropagation," *Complex Syst.*, vol. 5, pp. 603–643, 1991.
[5] R. Carroll, D. Ruppert, and L. Stefanski, *Measurement Error in Nonliner Models.* London: Chapman and Hall, 1995.
[6] J. T. Connor, "Boostrap methods in neural-network time series prediction," in *Int. Wkshp. Applicat. Neural Networks in Telecommun.*, J. Alspector, R. Goodman, and T. X. Brown, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. 125–131.
[7] B. Efron and R. Tibshirani, *An Introduction to the Boostrap.* New York: Chapman and Hall, 1993.
[8] B. Efron, *The Jackknife, The Boostrap, and Other Resampling Plans*, vol. 38 of *CBMS-NSF Regional Conf. Series in Appl. Math.* Philadelphia, PA: SIAM, 1982.
[9] W. A. Fuller, *Measurement Error Models.* New York: Wiley.
[10] A. R. Gallant, P. E. Rossi, and G. Tauchen, "Nonlinear dynamic structures," *Econometrica*, vol. 61, pp. 871–908, 1993.
[11] R. A. Jacobs, "Methods for combining experts' probability assessments," *Neural Computa.*, vol. 7, pp. 867–888, 1995.
[12] J. M. Karpov, "The relation between price changes and trading volume: A survey," *J. Financial and Quantitative Anal.*, vol. 22, pp. 109–126, 1987.
[13] H. R. Kunsch, "The jackknife and the bootstrap for general stationary observations," *Ann. Statist.*, vol. 17, no. 3, pp. 1217–1241, 1989.
[14] B. LeBaron, "Persistence of the Dow Jones index on rising volume," Univ. Wisconsin–Madison, Tech. Rep.
[15] ——, "Some relations between volatility and serial correlation in stock market returns," *J. Business*, vol. 65 pp., 198–219, 1992.
[16] R. Y. Liu and K. Singh, "Moving blocks jackknife and boostrap capture weak dependence," in *Exploring the Limits of the Boostrap*, R. LePage and L. Billard, Eds. New York: Wiley, 1992, pp. 225–248.
[17] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computa.*, vol. 4, pp. 448–472, 1992.
[18] R. Neal, *Bayesian Learning for Neural Networks*, No. 118 in Lecture Notes in Statistics. New York, Springer-Verlag, 1996.
[19] D. A. Nix and A. S. Weigend, "Learning local error bars for nonlinear regression," in *Advances in Neural Information Processing Systems 7 (NIPS*94)*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 488–496.
[20] G. Paass, "Assessing and improving neural-network predictions by the bootstrap algorithm," in *Advances in Neural Information Processing Systems 5 (NIPS*92)*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. San Mateo, CA: Morgan Kaufmann, 1993, pp. 196–203.
[21] R. Tibshirani, "A comparison of some error estimates for neural-network models," *Neural Computa.*, vol. 8, pp. 152–163, 1996.
[22] J. Timmer and A. J. Weigend, "Modeling volatility using state-space models," Leonard N. Stern School of Business, New York Univ., Tech. Rep. IS-97-11, 1997.
[23] A. S. Weigend and A. N. Srivastava, "Predicting conditional probability distributions: A connectionist approach," *Int. J. Neural Syst.*, vol. 6, pp. 109–118, 1995.
[24] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting the future: A connectionist approach," *Int. J. Neural Syst.*, vol. 1, pp. 193–209, 1990.
[25] ——, "Predicting sunspots and exchange rates with connectionist networks," in *Nonlinear Modeling and Forecasting*, M. Casdagli and S. Euband, Eds. Reading, MA: Addison-Wesley, 1992, pp. 395–432.
[26] A. S. Weigend, M. Mangeas, and A. N. Srivastava, "Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting," *Int. J. Neural Syst.*, vol. 6, pp. 373–399, 1995.
[27] A. S. Weigend, H. G. Zimmermann, and R. Neuneier, "Clearning," in *Neural Networks in Financial Engineering, Proc. NNCM'95,* London, A.-P. N. Refenes, Y. Abu-Mostafa, J. Moddy, and A. Weigend, Eds. Singapore: World, 1996, pp. 511–522.