Benchmarking Bayesian neural networks for time series forecasting

Rafael A. Calvo *
SEDAL,University of Sydney
rafa@sedal.usyd.edu.au
Marwan Jabri
SEDAL, University of Sydney
marwan@sedal.usyd.edu.au

ABSTRACT

We report a benchmarking of neural networks and regression techniques in a time series forecasting task. The estimation errors, computing costs and additional information obtained by Bayesian neural networks are compared with other neural network models and with Multivariate Adaptive Regression Splines (MARS). The Mackey Glass time series in chaotic regime was used to generate the two data sets, one with noise and one without.

1. Introduction

Novel artificial neural network (ANN) techniques are giving new development possibilities in many time series applications. Their performance must be compared with other ANN and with non neural models in order to choose the most suitable method for a particular application and data set. These performances are usually measured by the accurracy of the prediction, the computational costs and by other information they might provide.

In this paper we will consider the use of Bayesian ANN in a forecasting task, and compare its performance with other methods. As it is the case in many real world applications the variable to be predicted will have a non linear relation to other variables and might have noise corrupting the data points.

Having a set of data points we can arrange them, using an appropiate windowing, instances $(\overline{x}, y)$ which are assumed to be the result of:

$$y(\overline{x}) = f(\overline{x}) + \xi(\overline{x}),$$

where ξ is a noise term.

Training the network using standard back-propagation techniques [9],[8] or more complex variations of it [10] we get a network such that given a new input vector $\overline{x}$, we get an output $o(\overline{x})$ that can be interpreted as an estimate of the regression $f(\overline{x})$. In many applications however it is of great importance to quantify the accurracy of these estimations.

In section 1.1, we introduce Bayesian forecasting using neural networks, and in 1.2 the Markov Chain Monte Carlo (MCMC) techniques used to sample the weight space of the neural networks. In section 2 we describe the data sets used based the Mackey Glass series, the neural network architecture and other methodological issues. Section 3 describes the results and compares various back-propagation and neural network algorithms, and Multivariate Adaptive Regression Splines (MARS) to the Bayesian techniques, we show here (sec 3.1) how Bayesian learning performs better than other methods, specially in the noisy data set, the computational expense is high (sec. 3.2), but has the advantage (sec. 3.3) that it can be used to obtain the prediction intervals that describe a degree of belief of the predictions.

1.1. Bayesian Learning and forecasting

The goal of Bayesian forecasting is to obtain the predictive distribution: $P(y^{(n+1)}|D)$ where $y^{(n+1)}$ is the variable we want to predict at time $n+1$ and D is the knowledge we have expressed in a database that must at least contain the value y^n. In the general case:

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(n)}, y^{(n)})\} \quad (1)$$

where x is the "cause" of a causal (or multivariate) forecasting model. So finally $P(y^{(n+1)}|x^{(n+1)}, D)$ is the predictive distribution.

Model fitting is done in the Bayesian framework by writing the probability of having a parameter θ given the data $\overline{x}, \overline{y}$ and the model H like,

*Supported by a scholarship from Universidad Nacional de Rosario through the FOMEC project and a Norman I. Price Scholarship for Electrl. Engineering from The University of Sydney. Permanent address: Instituto de Fisica Rosario, Bvd. 27 de Febrero 210 Bis, 2000 Rosario, Argentina.

$$P(\theta|\overline{x},\overline{y},H) = \frac{P(\overline{y}|\overline{x},\theta,H)P(\theta|H)}{P(\overline{y}|\overline{x},H)} \quad (2)$$

which is equivalente to say that the:

$$posterior = \frac{likelihood \times prior}{evidence}$$

$P(\overline{y}|\overline{x},H)$ in the first inference level is just a normalizing constant but at the second level is called evidence.

The evaluation of this distribution involves marginalizing over all levels of uncertainity: Model selection (H), hyperparameters (Ω) and parameters (θ).

$$P(y^{(n+1)}|x^{(n+1)},D) =$$
$$\sum_H \int P(y^{(n+1)}|x^{(n+1)},\theta,\Omega)P(\theta,\Omega|D)d\theta d\Omega \quad (3)$$

where θ are the parameter (weights and biases) and Ω are the noise over parameters.

The evaluation of $P(y^{(n+1)}|x^{(n+1)},\theta,\Omega)$ only requires a single pass through the network.

Typically, marginalization over θ and H affects the predictive distribution (eq. 3) significantly, but integration over the hyperparameters Ω has a lesser effect. Marginalization can rarely be done analitically. The alternatives are Gaussian approximations [5], [6] and Monte Carlo Methods [7].

Complex models might produce overfitting the data, they also have an obvious extra computational cost. Bayesian inference can help tackle the problems of complexity, uncertainity and selection of probabilistic models.

1.2. Markov Chain Monte Carlo

To the Bayesian inference framework just defined we add the fact of using neural network models described by $f(\overline{x},\theta)$.

The best single valued prediciton that minimizes the squared error is the mean of the predictive distribution that we can define as the mean of the network outputs corresponding to the conditional distribution.

$$\hat{y}^{(n+1)} = \int f(x^{(n+1)},\theta)P(\theta,\Omega|D)d\theta d\Omega$$

In Monte Carlo, these integrals take the form of expectation functions

$$E(a) = \int a(\theta)Q(\theta)d\theta$$

Using a sample of values from Q (posterior) we can approximate this integral by

$$E(a) \simeq \frac{1}{N}\sum a(\theta^{(t)})$$

where $\theta^{(1)},\theta^{(2)},....,\theta^{(n)}$ are generated by a process that results in each of them having the distribution defined by Q.

We can write the posterior for the parameters and hyperparameters after "receiving" D (eq.1)as,

$$P(\theta,\Omega|D) = \frac{P(D|\theta,\Omega)P(\theta,\Omega)}{P(D)} \quad (4)$$

In MCMC we do not try to express the posterior in a direct way. The iterative method first gives a state vector $\zeta^1 = (\theta,\Omega)$ and generate a new random state vector ζ^2 from a probability distribution $q(\zeta^1,\zeta^2)$ then we obtain ζ^3 by sampling from $q(\zeta^2,\zeta^3)$, and so forth. The transition probability q is constructed in such a way that an ergodic Markov process is defined with stationary distribution equal to the desired posterior distribution.

Several methods might be used to obtain the sample from the posterior distribution in the MCMC frame work: Metropolis method, Gibbs sampling, stochastic dynamics and Hybrid Monte Carlo.

The hybrid Monte Carlo method developed by Duane [2], and used in this paper is a merge of the stochastic dynamics and Metropolis algorithms.

Basically the Metropolis algorithm generates the sequence $\theta^{(1)},\theta^{(2)},...$ mentioned above by first generating $\theta^{(t+1)}$ from $\theta^{(t)}$ by: generating a "candidate state" from a proposal distribution and then accepting or not based in its probability density relative to that of the old state with respect to the desired invariant distribution Q. If $\theta^{(t+1)}$ is accepted then becames next state in the chain. If $\theta^{(t+1)}$ is not accepted then the states the same as the old.

Creating a fictious variable p and time τ, we can use the Hamiltonian stochastic dynamics formalism and sample q (previously called θ) for a constant energy H, sampling for different H in a second stage.

Since Hamiltonian dynamics cannot be simulated exactly the leapfrog method is used.

Putting together these two, what the Hybrid Monte Carlo algorithm does is:

1. Randomly choose if the transition goes forward on backwards in the fictious time ($\lambda = \pm$).

2. The current state $(q,p) = (\hat{q}(0),\hat{p}(0))$ evolves after L leapfrog iterations $\lambda\eta$ size to $(\hat{q}(\lambda\eta L),(\hat{p}(\lambda\eta L)) = (q^*,p^*)$

3. Regard (q^*,p^*) as candidate for next state with probability

$$min(1, exp(-(H(q^*,p^*) - H(q,p))),$$

where H can be interpreted as an error function being minimized.

Summarizing, Bayesian learning takes a distribution of the parameters $P(\theta,\Omega)$, combines it with information from a training set and then integrates the posterior obtaining the desired forecast. Important features are that: the results do not overfit the data, the prediction accurancy can be improved and that the prediction intervals can be estimated.

2. Data and Methods

The Mackey Glass equation was originally developed for modelling white blood cells production. It is a time delay ordinary differential equation:

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t)$$

chosing $\tau = 30$ the equation becomes chaotic and only short term forecasts are feasible.

Integrating the equation over the range $[t, t+\delta t]$ by the trapezoidal rule yields after re-arranging terms:

$$x(t+\Delta t) = \frac{2-b\Delta t}{2+b\Delta t}x(t)+$$

$$\frac{a\Delta t}{2+b\Delta t}\left[\frac{x(t+\Delta t-\tau)}{1+x^c(t+\Delta t-\tau)} + \frac{x(t-\tau)}{1+x^c(t-\tau)}\right]$$

And constants were assigned the values a=0.2, b=0.1 and c=10 with an integration step $\Delta t = 1$. The data is sampled every 6 points.

Patterns were generated windowing 6 inputs and 1 output. 500 patterns were used for training and 500 for testing in Bayes, but 50 of the training patterns had to be used for cross-validation in the backpropagation approach. The fact that the Bayesian approach does not produce overfitting and that no data is wasted is of great importance in real world applications where data is scarce. The second set of pattern files is equal to the above but with a positive uniform noise of 20% of the original signal added to the targets.

All the backpropagation and Bayesian networks used had 6 input units, 5 *tanh* hidden units and 1 output unit. The output can only be linear in the Bayes implementation we used and so we tried linear and the more commonly used *tanh* output units in backpropation. In the case of tanh output units the data was normalized dividing by 2, so all the points are in the range (0,1).

Each run of the Bayesian learning had an initialization and a sampling phase. During initialization different stepsize factors (η) and lenghts L are tried to get a stable system. During sampling different L were tried, choosing the one with the smallest mean error in the last 200 states. η is choosed so that no more than 20% of the candidates states are rejected, but big enough so the sampling is efficient. The limitations of the method are computational: 1 sampling run takes about 15 hours on a Sun Ultra I.

We have used version 3.6 of the MARS library provided by Friedman. In MARS we have used a maximum of 2 interactions, 30 and 100 basis functions. Noticing very litle improvement on more complex regression schemes.

	$ARV * 10^{-4}$ train set	$ARV * 10^{-4}$ test set
a-stp (best)	22.8	19.5
BFGS (best)	23.2	20.9
Shann (best)	37.0	36.7
bp-lin	58.95	68.55
bp-tanh	12.01	12.51
bayes 128	7.44	8.34
bayes 2k	6.66	7.43
bayes 32k	6.40	7.56
mars 30bsfn	13.32	15.9
mars 100bsfn	3.26	7.17

Table: 1: Performances in Mackey Glass series. a-stp: Hertz rules adaptive steepest descent with weight update per pattern. shann: is the Shanno varian of conjugate gradients with inexact line search. BFGS: is the quasi-Newton method with inexact line search.

3. Results

3.1. Estimation

The first goal of this research was to compare the accurracy of Bayesian learning forecast against other neural network and regression models. We use Azoff's NN benchmarks [1], the results of an implementation of backpropagation in MUME [4], the Multivariate Adaptive Regression Splines method [3] and compare them with the results obtained using Neal's implementation of Bayesian learning [7].

Table 1 shows the result for the Mackey Glass series without noise and Table 2 for the series with 20% of noise added to the targets.

The error ARV is the average relative variance:

$$ARV = \frac{1}{N\sigma^2}\sum_{i=1}^{n}(t_i - o_i)^2$$

where the σ is the standard deviation of each set. The normalization implies that if the estimated mean of the data is used as predictor, ARV=1 is obtained.

In the case of Bayes the mean ARV for 200 Markov Chain Monte Carlo states is shown.

3.2. Computational expenses

There are very important differences in the computational cost of the methods used. MARS took always less than 3 minutes, the backpropagation simulations took about 5 hours and the Bayesian

	$ARV * 10^{-2}$ train set	ARV * 10−2 test set
bp-lin	19.20	19.17
bp-tanh	10.68	11.99
bayes 128	8.91	9.40
bayes 2k	8.90	9.39
bayes 32k	8.97	9.39
mars 30bsfn	8.32	9.56
mars 100bsfn	8.21	9.97

Table: 2: Performances in Mackey Glass series with 20% noise in the targets

```
For each pattern of the test set:
 1. Select one state of the MCMC sampling.
 2. Generate file of error-output pairs.
 3. Sort by error.
 4. Delete the 5% of pairs with biggest
    error.
 5. Save minimum and maximum output in a
    file.
```

Fig. 1: Procedure used for estimating the prediction intervals.

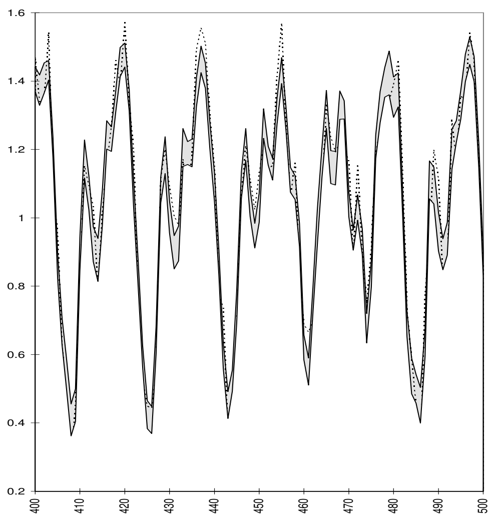

Fig. 2: Prediction intervals (grey area) and expected target values (dashed line) for the noisy Mackey Glass series.

Markov Chain Monte Carlo simulations took about 20 hours.

If the computational costs are a determining factor in the application studied these differences must be taken into account. MARS gives good estimations at a minimum cost. We believe that MCMC simulations need an important time optimization.

3.3. Prediction intervals

For many applications it is important to quantify the accurracy of the predictions produced by a model. This accurracy can be categorized in two ways: the accurracy of the prediction of the true regression and that of the prediction with respect to the observed output. We will refer to the latter through the prediction intervals. We show in Figure 1 the procedure used for estimating the prediction intervals. Figure 2 shows the 95% prediction intervals and the target values for the last 100 points of the test set, this interval is giving the region of the sampled predictive ditribution with smaller error. Note that the prediction intervals of the Bayesian learning fall short of the real value in the peak areas.

4. Conclusions

Different neural networks and regression techniques have been compared in a time series forecasting task. We have shown that the important differences in prediction accurracy, computational cost and additional information provided by each method are important factors to be considered for a specific application. These factors are also function not only of the amount of data used but also of noise.

Bayesian neural networks using Markov chain Monte Carlo simulations perform better than other NN techniques on the Mackey Glass problem, only outperformed by MARS on the clean data set. Their relative computational costs are high but they do provide prediction intervals which may be useful in certain applications. Also, in real world problems with noisy and scarce data, the fact that Bayesian neural networks do not overfit could be of practical value. Further study is being done adding new techniques to the benchmark trying them on different data sets and estimating formalized prediction intervals. .

References

[1] E.M. Azoff. *Neural Network time series forecasting of financial markets.* Wiley, Sussex, England, 1994.

[2] S. Duane, D. Kennedy, B. Pendleton., and D. Roweth. Hybrid monte carlo. *Physics Letter B*, 195:216–222, 1987.

[3] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991.

[4] M. Jabri, E. Tinker, and L. Leerink. Mume: A multi-net multi-architecture neural simulation environment. In Josef Skrzypek, editor, *Neural*

network simulation environments, pages 229–247. Kluwer, Norwell:MA, 1994.

[5] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[6] D.J.C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[7] R. Neal. *Bayesian learning for neural networks*. University of Toronto, Canada, 1995.

[8] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, 1986.

[9] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by backpropagating errors. *Nature*, 323:533–536, 1986.

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