Inferencing Bayesian Networks From Time Series Data Using Natural Selection

Andrew J. Novobilski
The University of Texas at Arlington
P.O. Box 4328
Gulf Shores, AL 36547
(334) 948-8373
andyn@novotech.com

Farhad A. Kamangar, Ph.D.
Computer Science & Engineering Dept.
University of Texas at Arlington
Arlington, TX 76019
(817) 272-3617
kamangar@cse.uta.edu

Abstract

This paper describes a new framework for using natural selection to evolve Bayesian Networks for use in forecasting time series data. It extends current research by introducing a tree based representation of a candidate Bayesian Network that addresses the problem of model identification and training through the use of natural selection. The framework constructs a modified Naïve Bayesian classifier by searching for relationships within the data that will produce a model for the underlying process generating the time series data. Experimental results are presented that compare forecasts in the presence of multiple sources of information made using the naturally selected belief network versus a random walk.

Introduction

Recent literature in the fields of Data Mining and forecasting has included articles outlining new approaches for forecasting time series data utilizing Bayesian Networks[1]..A Bayesian Network is a directed acyclic graph with nodes representing the attributes of the model and directed links representing a causal relationship between parent and child. Together, this information represents the dependence between variables and gives a concise specification of the joint probability of the model [2]. Significantly, it provides a white box approach to representing relationships that exist within the domain being modeled and can handle inferencing in the absence of complete information.

One area of particular interest is the way in which both the structure and values of network representations can be learned from the data alone. To this end, a significant amount of work has been done on learning the network structure given an identified set of attributes of the data to be modeled. However, in the case of forecasting a time series, each past value becomes a potential attribute for use in the network model describing the process that produced the time series. This increase in computational complexity, plus the ability to enhance the forecast model using data sources external to the time series being forecast, underscores the need for an autonomous method for identifying the forecast model. Examples of frameworks that address these issues in the context of using Bayesian Networks to forecast time series data include [3,4].

The framework presented here relies upon evolutionary computing to derive a valid forecasting model from the time series to be forecast by using Genetic Programming. Genetic Programming [5] is a search method based on the mechanics of natural selection [6] that has been adapted to manipulate tree based models. There is a strong potential, as evidenced in completed work such as Kennedy et. al.[7] and De Jong et. al.[8], for evolutionary programming to have a positive impact on the identification of appropriate Bayesian network models through correct encoding of the network structure and proper choices of fitness functions.

One of the most promising aspects of using a genetic programming framework for the identification of Bayesian networks is the closeness the two approaches already share. For example, Ramoni and Sebastiani [9] point out that learning both the Belief Net's parameters and structure from available data can be viewed as a search problem for identifying a suitable Bayesian network for modeling the sample data. Heckerman also discusses this by pointing out several methods for selecting candidate network representations, in addition to discussing various methods for evaluation the fitness of the learned representation with respect to the sample data.

A Bayesian Network Approach to Forecasting

The Evolutionary Bayesian Network approach utilizes a Bayesian network to model a stochastic process. The network combines conditionally dependent and independent information to produce a model of the stationary process producing the first difference in the time series being forecasted.

Consider the time series Y_t that is passed through a differencing mechanism to produce the series Z_t. The series Z_t can then be represented by the model:

$$Z_{t+1} = m(Z_t, Z_{t-1}, ..., a_t, a_{t-1}) \quad \text{[Eq. 1]}$$

where a_t, a_{t-1}, … are a set of independent, identically distributed random variables and m() is a function both the previous values of Z and the random variables a.

One of the simplest models m() is that of the Random Walk:

$$Z_{t+1} = Z_t + a_t \quad \text{[Eq. 2]}$$

The Random Walk, first discussed by Bachelier in 1900 [10], has been used as the de Facto standard of comparison for methods that forecast future values in the stock market. The predictive model for the Random Walk is:

$$\hat{Z}_{t+1} = Z_t + E[a_t] \quad \text{[Eq 3]}$$

Assuming that a_t is normally distributed with mean equal to 0, Eq 3 becomes:

$$\hat{Z}_{t+1} = Z_t \quad \text{[Eq 4]}$$

and will be used as the baseline model predictor that the Bayesian Network model predictor is compared to.

The Bayesian model representation of Eq 1 is:

$$Z_{t+1} = \delta, \max_{\min(Z)\le\delta\le\max(Z)} P(\delta \wedge Z_t \wedge Z_{t-1} \wedge \ldots) \quad \text{[Eq 5]}$$

where min(Z) and max(Z) are the minimum and maximum possible values of Z and δ is a continuous variable representing the possible values for Z at time t+1.

Although possible to work with predictors that use the continuous value δ, the framework discussed in this paper relies on the set of discrete values D, containing elements $\{d^1, d^2, \ldots\}$. The values of D are defined for the predictive model by dividing up the range (min(Z),max(Z)) into N discrete ranges R with boundaries λ,such that:

$$\min(Z) < R_1 \le \lambda_1 \quad \text{[Eq. 6]}$$

$$\lambda_{i-1} < R_i \le \lambda_i \quad \text{for i=2,…,N-1} \quad \text{[Eq. 7]}$$

$$\lambda_{N-1} < R_N < \max(Z) \quad \text{[Eq. 8]}$$

with d^i defined as:

$$d^1 = \lambda_1 \quad \text{[Eq. 9]}$$

$$d^i = \frac{\lambda_{i-1} + \lambda_i}{2} \quad \text{for i=2,…,N-1} \quad \text{[Eq. 10]}$$

$$d^N = \lambda_N = \max(Z) \quad \text{[Eq. 11]}$$

Given the values of D, the predictive model is defined as a Bayesian Classifier. The classifier is trained using a set of attributes that relate values of d^i with prior values of Z and returns the predicted value for d_t by utilizing the maximum a posteriori hypothesis to create:

$$\hat{Z}_{t+1} = \hat{d}^i, \max_{d^i \in D} P(d^i \wedge Z_t \wedge Z_{t-1} \wedge \ldots) \quad \text{[Eq. 12]}$$

Initially, the simplifying assumption is made that the prior values of Z are conditionally independent given the value of Z at time t. Based on this assumption, Eq 14 can be rewritten as the Naive Bayes Classifier:

$$\hat{Z}_{t+1} = \hat{d}^i, \max_{d^i \in D} P(d^i) \prod_{j=1}^{t} P(Z_{t-j} \mid d^i) \quad \text{[Eq. 13]}$$

The computational impact of using the Naive Bayes classifier is important. It greatly reduces the amount of storage and number of calculations required to represent the Bayesian network. This is significant in an natural selection environment where all members of the population must be created, trained, and then tested for fitness during each generation. However, the simplifying assumption of conditional independence between prior values of Z means that the MAP hypothesis will only be returned when this assumption is true. This is not the case when a process has an autoregressive component to it. Therefore, it is necessary to support the case where the value of Z_t is conditionally dependent on a subset of the values of Z at prior times.

The framework supports the possibility of autoregressive components by moving from a Naive Bayes classifier to a simple Bayes classifier, defined as:

$$\hat{Z}_{t+1} = \hat{d}^i, \max_{d^i \in D} P(d^i \mid Z_{t-l_1}, Z_{t-l_2}, \ldots) \prod_{k \in L} P(Z_{t-k}) \prod_{\substack{j=1 \\ j \notin L}}^{t} P(Z_{t-j} \mid d^i) \quad \text{[Eq. 14]}$$

where L is the subset of lags $\{l_1, l_2, l_3, \ldots\}$from K, the set all lags {0,1,…,t}.

The last improvement the framework makes to the basic classifier is the application of Occam's Razor to the number of terms used within the classifier. This subset of K, K' is then used giving:

$$\hat{Z}_{t+1} = \hat{d}^i, \max_{d_i \in D} P(d^i \mid Z_{t-l_1}, Z_{t-l_2}, \ldots) \prod_{k \in L} P(Z_{t-k}) \prod_{\substack{j \in K' \\ j \notin L}} P(Z_{t-j} \mid d^i) \quad \text{[Eq. 15]}$$

Figure 1 shows a sample classifier, in network form, for Z_{t+1} that relies on the conditionally dependent terms $\{Z_{t-2}, Z_{t-3}\}$ and the conditionally independent terms $\{Z_{t-1}, Z_{t-5}\}$.

Natural Selection

Given the desired form of the Evolved Bayesian Network model, it is necessary to specify a tree structure that will satisfy the requirements of the natural selection process. Specifically:

1. The tree structure must allow for the possible selection of any Z_{t-i} into either the conditionally independent or conditionally dependent section of the network.
2. The tree must allow for the dynamic specification of range mapping (positioning of λ's) during the quantitization process of individual states for each variable node.
3. The tree must not inhibit the genetic operators used in the natural selection process.

The tree used by the natural selection framework to evolve populations of member Bayesian Networks is best described as a set of trees within a tree structure. The root tree, shown in Figure 2, is used to describe the member variables contained within the network and the conditional relationships between them. Each node contains a single integer attribute that will be used to determine some aspect of the network based on the node's

position relative to the root. The first child node always represents Z_t, while each of the remaining children represent the remainder of the Z_{t-i}'s used in defining the predicting model. Specification of the Z_{t-i} being represented by a particular node is encoded in that node's value.

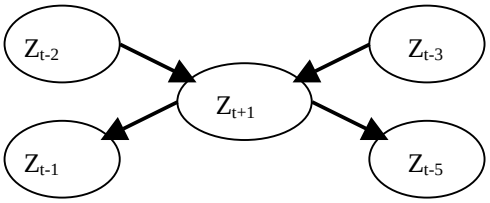

Figure 1 -A Simplified Bayesian Network for Selecting d_i.

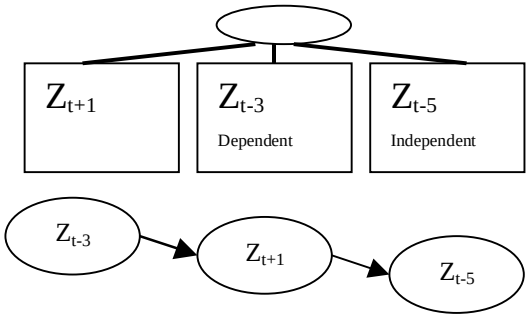

Figure 2 - A Sample Root Tree and the Bayesian Network it represents.

Each node contained within the root tree contains a subtree that encodes the λs used to partition the interval (min(Z), max(Z)) into discrete categories. During the natural selection process, high level belief networks can be modified by adding or removing subtrees to the immediate child level of the root tree. Enhancements to the low level quantitization information is made by attaching new subtrees to the children of the nodes representing variables in the network. Since the value of the node in the genetic encoding is the same in all cases, the listed constraints are met.

The Genetic Programming Framework

The Genetic Programming framework takes a series to be forecast, a set of related data series to be used in predicting the forecast value, and a set of parameters that drive the process. The main algorithm imports the data (Y_t). converts the series to be forecast into the desired difference model (Z_t) and then divides the data into training, test, and evaluation subsets. Next, the evolutionary search mechanism is invoked and relies upon Netica, a commercial off the self (COTS) product with a programmable Applications Programmer Interface, to build, train, and provide probabilities for the query node during forecasting. A series of independent runs are completed, with each run consisting of applying the reproductions, crossover, and mutation operators to produce a series of generations of potential Bayesian Networks. The best individual across all generations of all runs is kept and used for the final forecast. Once the final forecast is made, it is compared against a Random Walk as a relative performance measurement. Both the resulting forecast values and performance statistics are preserved for post processing purposes.

The evolutionary part of the algorithm derives from the use of reproduction, crossover, and mutation to generate a new population from an old one. The choice of using crossover or reproduction to add members to the new population is based on the desired percentage of new members generated using reproduction. For example, it 30% of the new population is to be generated by reproduction, then the remaining 70% will be generated by crossover. Reproduction is the selection of an individual to be carried forward as-is into the next generation. The random selection of individuals is made using a probability distribution based on the linear fitness ranking [11] of the population. Members with the best fitness values are higher ranked and therefor selected more often.

Crossover is the creation of two new members by randomly selecting two existing members (possibly the same member twice) of the current population in the same manner as reproduction. Then a single link in each of the individuals is randomly selected and broken to create an upper and lower partial tree. The upper tree of the first individual is combined with the lower tree of the second individual and the upper tree of the second individual is combined with the lower tree of the first individual two create two new members for the next generation. Mutation is defined as randomly changing the value of a node within the tree representation of an individual. This operation occurs with a very low probability and is intended to displace the individual if it becomes stuck in a local minimum.

In order for the Evolving Bayesian Network framework to utilize the given tree representation to encode the Bayesian network it was necessary to add one adaptation related to crossover and mutation. Since crossover could occur between arbitrary points, it is possible that the act of crossover or mutation could change either the number of nodes in the network or the organization of the categories that each node was broken into. Therefore, the member trees had to be scanned for validity. If a tree was found to be invalid it was kept in the population but given a very low fitness ranking. Invalid trees were added to the end of the ranked list after being sub ranked by their complexity, with the least complex being given a better rank position.

Once the tree is transformed into a network, training commences. Fitness is computed by using the individual Bayesian Network to select the most likely value of the time series for the next time period given information from previous values of the time series. Training is

accomplished making a subset of the data available to the Netica case based learning functionality after the candidate tree was defined.

Once training is completed, the individual is measured for fitness, using the fitness function, F(), defined as:

$$F(member_i) = 1 - \sqrt{MNSE(\hat{Z}, Z)} \quad \text{[Eq. 16]}$$

where MNSE, the Mean Normalized Square Error, is computed as:

$$MNSE(\hat{Z}, Z) = \frac{1}{|Z|}\sum_{i=1}^{|Z|}\left[\frac{\hat{Z}_i - Z_i}{\max(Z) - \min(Z)}\right]^2 \quad \text{[Eq. 17]}$$

Fitness is measured by using the trained network to forecast values for the subset of Z set aside for testing.

This process is then repeated for the selected number of runs, generations, and population size until the best Bayesian Network is produced by the framework. The resulting Bayesian Network is then used to forecast values for the remaining subset of Z set aside for the purpose of evaluating the forecasting method against a Random walk.

The success of the search process depends on new individuals being created from the most relatively fit in the current population. A fitness value for an entire member is considered to be shared by each member of the power set of individual sub trees it contains. These subtrees, or schemata, are then combined during the generation of a new population to create, in theory, a more robust population that will produce increasingly fit individuals. Unfortunately, this technique is not guaranteed to converge to the best fit individual and therefore requires the use of large populations randomly sampled to create the initial population such that as many schemata as possible are represented.

Experimental Results

Table 1 shows the overall results of forecasting for the GM time series given the number of records to hold back for testing, and the method used to provide the forecast. Each forecast consists of 40 one step ahead predictions using a network that was evolved using a time series with 942 values. Available historical data was delivered as a single record consisting of the 19 previous values of the variable. The HLD column indicates the number of records held back for the testing of the network. The remaining 942 – HLD records were used in the natural selection of the Bayesian Network and its supervised training. Each experimental forecast was made using a population of size 25, evolved through 15 generations, with the natural selection process repeated five times per forecast. This was sufficient to allow the process to achieve convergence when 70% of the population tested better than the random walk. In addition to the univariate forecast, represented by MAP in the METHOD column, experimental results were compiled for the multivariate method M-MAP (allowing the framework to build a network utilizing information from other securities) and for the random walk, RWALK.

HLD	METHOD	GNS	RNS	MNSE TRN	MNSE EVL	MAPE EVL
40	MAP	15	5	0.001647	0.001434	1.6317%
40	M-MAP	13	3	0.001615	0.001440	1.6576%
40	RWALK			0.001662	0.001457	1.6032%
60	MAP	12	3	0.001554	0.001612	1.6768%
60	M-MAP	15	3	0.001611	0.001376	1.5040%
60	RWALK			0.001659	0.001457	1.6032%
80	MAP	9	4	0.001425	0.001805	1.6845%
80	M-MAP	11	12	0.001475	0.001472	1.5958%
80	RWALK			0.001477	0.001457	1.6032%

Table 1 - Experiment Results for Forecasting the Future Value of GM Stock Prices.

The last three columns contain information on the forecasting error obtained when comparing the actual value of the series to the forecast value produced by the best individual produced by the evolutionary process. This information has been split into three columns; MSNE TRN, MSNE EVL, and MAPE EVL. MSNE TRN contains the Mean Square Normalized Error for the indicated number of records held back for the testing of the trained network. MSNE EVL contains the MSNE for the final set of records used to evaluate the trained network on information not seen during the selection process. The MAPE EVAL column provides the Mean Average Percent Error between the true and forecast value for the training record set and the evaluation record set of the most fit member of the evolved population. MAPE was provided as an accepted standard of error evaluation and was not used in the process of naturally selecting the Evolved Bayesian Network used to provide the forecasts. Figure 3 shows the complete dataset for the GM Stock Price time series. This series shows a generally rising trend over time. For the GM forecast, the winning Bayesian network from the natural selection process, shown in Figure 4 relied upon the previous price of GM stock, in addition to previous prices from the GE and IBM stock as well. This network was used to generate the forecast shown in figure 5. Figure 6 shows the Average Percentage error for each of the forecast values generated during the evaluation phase of the framework.

Discussion

Overall, the framework for naturally selecting Bayesian Networks outperformed the Random walk, but not by a large margin. The best forecast was delivered by an Evolved Bayesian Network with a holdback of greater than or equal to 60 records, 20 records more than the number forecast during the evaluation phase of the framework. Additionally, the best MAPE EVAL value belonged to the same forecast that contained the best MSNE EVAL value. The forecasting results for the best naturally selected Bayesian Network were within 4% for first 7 values and 5% for remaining values over the data

held back for evaluation of the selected time series. It is interesting to note that the selected forecast model holds within the noted error tolerance over an extended period of time without requiring retraining of the network.

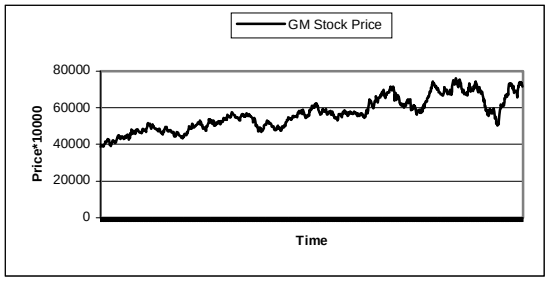

Figure 3 - The GM Stock Price Time Series.

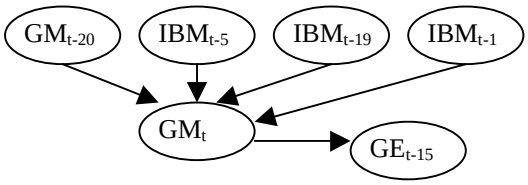

Figure 4 - The Bayesian Network Model for the Price of GM Stock.

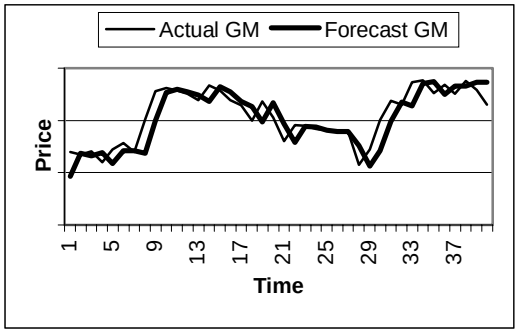

Figure 5 - The Forecasted Price of GM Stock.

Based on the initial results, it would be interesting to see if the selected nodes and their accompanying probabilities could be mapped directly to the coefficients of an ARIMA model with some degree of utility. It would also be interesting to study the effect on the applicability of using the Model Averaging method is each of the available hypothesis were significantly different.

Conclusions

The problem of identifying and training a model that can be used to forecast the value of a time series given historical data has been the focus of much research. The method discussed in this paper combines natural selection with a Bayesian Model. The resultant forecasting framework shares similar model components with the Box and Jenkin's ARIMA model and has been experimentally shown to evolve Bayesian Networks that perform better than a random walk.

The framework for naturally selecting Bayesian Networks was also able to separate the classification of data points into relevant attributes from the process of using the discovered attributed in the high level forecasting model. This resulted in allowing the operators involved with the genetic programming based search of the model space to cause changes in the quantitization levels of selected attributes to occur more frequently than the organization and addition/removal of the attributes themselves. In essence, the natural selection process would move the search to a high probability area of success, and then allow for refinement of the selected model to obtain the best solution in that locality.

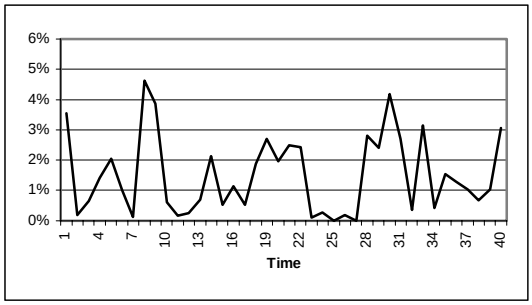

Figure 6 - The Average Percent Error of the Forecasted GM Stock Prices.

References

[1] Heckerman, D., A. Mamdani, and M. Wellman. 1995. Real World Applications of Bayesian Networks. *Communications of the ACM* 38, no. 5: 25-30.

[2] Russell, Stuart, and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs: Prentice Hall.

[3] Abamson, B. 1991. ARCO1: An Application of Belief Networks to the Oil Market. In *Proceedings of The Uncertainty in Artificial Intelligence Conference*: 1-8.

[4] Dagum, P., A. Galper, E. Horvitz, and A. Seiver. 1995. Uncertain Reasoning and Forecasting. *International Journal of Forecasting* 11, no. 1: 73-87.

[5] Koza, John. 1996. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press.

[6] Holland, John. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge: MIT Press.

[7] Kennedy, H., C. Chinniah, P. Bradbeer, and L. Morss. 1997. The Construction and Evaluation of Decision Trees In a Comparison of Evolutionary and Concept Learning Methods. In Selected Papers from the AISB International Workshop Evolutionary Computing. Springer-Verlag.

[8] De Jong, K., W. Spears, and D. Gordon. 1993. Using Genetic Algorithms for Concept Learning. San Diego, CA: Naval Research Lab Technical Report.

[9] Ramoni, M., and P. Sebastiani. 1997. Learning Bayesian Networks from Incomplete Databases. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. 401-408.

[10] Bachelier, Louis. 1964. The Random Character of Stock Market Prices. Edited by Paul Cootner. Theory of Speculation. MIT Press.

[11] Miller, B., and D. Goldberg. 1996. Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. In *Evolutionary Computation Journal* 4, no. 2: 113-131. 1996.