Mining Cross-Predicting Stochastic ARMA Time Series in SQL Server 2005

Bo Thiesson and Jesper Lind
Microsoft Research, USA

Abstract

We present a prototype that we have developed for analyzing so-called stochastic ARMA models in SQL Server 2005, Analysis Services. The class of stochastic ARMA models extends the classic ARMA time-series models by replacing (or smoothing) the deterministic relationship between target and regressors in these models with a conditional Gaussian distribution having a small controllable variance. As this variance approaches zero, a stochastic ARMA model approaches a classic ARMA model. We represent a stochastic ARMA model as a directed graphical model. In doing so, we benefit from the ability to apply standard graphical-model-inference algorithms during parameter estimation (including estimation in the presence of time series with incomplete data), model selection, and prediction. The graphical model representation also offers a visual representation that is easy to interpretate. We demonstrate how the graphical representation in this way lends itself as a conceptually easy way of extending the models to handling cross predicting time series, periodicity, and trends.

Keywords: time series, graphical models, ARMA, Bayes net, SQL Server.

1 Introduction

Graphical models have been used to represent time-series models for almost two decades (e.g., Dean and Kanazawa [1]; Cooper, Horvitz, and Heckerman [2]). The benefits of such representation include the ability to apply standard graphical-model-inference algorithms for parameter estimation (including estimation in the presence of missing data), for model selection, and for prediction. We introduce the most fundamental concepts for this class of models when needed in this paper. A gentle introduction and more details can be found in, for instance, Jensen [3].

We express the classic autoregressive moving average (ARMA) time-series model (e.g., Box, Jenkins, and Reinsel [4]) as a graphical model to achieve the above mentioned benefits. However, as demonstrated in Thiesson *et al.* [5] and in Thiesson and Meek [6] the classic ARMA model includes deterministic relationships, making it effectively impossible to estimate model parameters via standard algorithms, such as the Expectation–Maximization (EM) algorithm or gradient-based optimization algorithms, in this framework. Consequently, a variation of the ARMA model for which these estimation algorithms can be applied was introduced. The variation, called stochastic ARMA (or σARMA), replaces the deterministic component of the ARMA model with a conditional Gaussian distribution having a small variance. As this variance approaches zero, the stochastic ARMA model approaches the classic ARMA model. The stochastic version of the ARMA model not only has the desired effect of making the standard algorithms effective for parameter estimation. It also provides a controlled and principled way of smoothing the prediction model by accounting for the smoothing during the parameter estimation and model selection process as opposed to just learning a classic ARMA model and then add an additional smoothing variance to the predictions.

As demonstrated in Thiesson *et al.* [5] the graphical model representation is very intuitive and is therefore very easy to extend to more sophisticated models, including cross-predicting time series. We will in this paper concentrate more on cross prediction—the use of one time series to help predict another time series (e.g., Ghahramani [7]; Meek, Chickering, and Heckerman [8])—and we will in an intuitive graphical way show how cross predictors become a mean for handling periodicity and trends.

We have implemented the stochastic ARMA models as a plug-in for SQL Server 2005, Analysis Services. Detailed information on how to build a such plug-in algorithm is available at [9]. Our implementation is intended as a tool for mining large data bases for cross-predicting time series, and computational efficiency is therefore of great importance.

The graphical model framework can handle cross predictors in a theoretical sound way, but they introduce some challenges regarding computational efficiency during both learning and prediction. We will account for these challenges and describe the approximations we have made in order to ensure fast learning and prediction. We emphasize that these approximations only affect models with cross-predicting time series.

Finally, Thiesson *et al.* [5] has already empirically demonstrated that stochastic ARMA models benefit prediction by the controlled way of smoothing and by allowing cross-predicting time series. We will not pursue the quality of stochastic ARMA models further in this paper. Instead, experimental results will concentrate on the scalability of learning these models. We will demonstrate that particular settings for the parameterization of a model can have a dramatic effect on the computational efficiency.

2 Time-Series Models

We begin by introducing notation and nomenclature. We denote a temporal sequence of observation variables by $Y = (Y_1, Y_2, \ldots, Y_T)$, and we denote a sub-sequence consisting of the i'th through the j'th element by $Y_i^j = (Y_i, Y_{i+1}, \ldots, Y_j)$, $i < j$. Time-series data is a sequence of values for these variables denoted by $y = (y_1, y_2, \ldots, y_T)$. We suppose that these observed values are obtained at discrete, equispaced intervals of time. In this paper we will consider incomplete observation sequences in the sense that some of the observation variables may have missing observations. For notational convenience, we will represent such a sequence of observations as a complete sequence, and it will be clear from context that this sequence has missing observations.

The ARMA time-series models and the stochastic variants considered in this paper, associate a latent "white noise" variable with each observable variable. These latent variables are denoted $E = (E_1, E_2, \ldots, E_T)$.

Some of the models will also contain cross predictors. A cross predictor is an observation variable from a related time series, which is used in the predictive model for the time series under consideration. For instance, $Y' = (Y'_1, Y'_2, \ldots, Y'_{T'})$ and $Y'' = (Y''_1, Y''_2, \ldots, Y''_{T''})$ may be observation variables from related time series, where Y'_{t-12} Y'_{t-1} and Y''_{t-1} are cross-predictor variables for Y_t. If Y'_s is a cross predictor for Y_t, it is always the case that $s < t$. Let C_t denote a vector, such as $(Y'_{t-12}, Y_{t-1}, Y''_{t-1})$, of cross-predictor variables for Y_t. The set of cross-predictor vectors for all variables Y is denoted $C = (C_1, C_2, \ldots, C_T)$.

The stochastic time-series models can handle incomplete time series, where some values for the observation variables, Y, are missing. Hence, in real-world situations where the length of multiple cross-predicting time series do not match, we have two ways of insuring that Y, E, and C are all of the same length. We can introduce observation variables with missing values, or we can shorten a sequence of observation variables, as necessary. Both of these options are possible in our implementation, and in the following we will therefore assume that Y, E, and C are all of the same length.

2.1 ARMA Models

In slightly different notation than usual (see, e.g., Box, Jenkins, and Reinsel [4]) the ARMA(p, q) time series model is defined as the deterministic relation

$$Y_t = \zeta + \sum_{j=0}^{q} \beta_j E_{t-j} + \sum_{i=1}^{p} \alpha_i Y_{t-i} \tag{1}$$

where ζ denotes the intercept, $\sum_{i=1}^{p} \alpha_i Y_{t-i}$ is the autoregressive (AR) part, $\sum_{j=0}^{q} \beta_j E_{t-j}$ is the moving average (MA) part with β_0 fixed as 1, and

$E_t \sim \mathcal{N}(0, \gamma)$ is "white noise" with E_t mutually independent for all t. The construction of this model therefore involves estimation of the free parameters ζ, $(\alpha_1, \ldots, \alpha_p)$, $(\beta_1, \ldots, \beta_q)$, and γ.

For a constructed model, the one step-ahead forecast $\hat{Y}_t$ given the past can be computed as

$$\hat{Y}_t = \zeta + \sum_{j=1}^{q} \beta_j E_{t-j} + \sum_{i=1}^{p} \alpha_i Y_{t-i} \tag{2}$$

where we exploit that at any time t, the error in the ARMA model can be determined as the difference between the actual observed value and the one step-ahead forecast

$$E_t = Y_t - \hat{Y}_t$$

The variance for this forecast is γ.

An ARMA model can be represented by a directed graphical model—also called a Bayes net. A directed graphical model associates a graph with the model, where the graph is a very intuitive representation of the structural relations in the model. As an example, Figure 1 shows the graphical representation for an ARMA(2,2) model. Nodes in the graphical representation represents variables in the model and arcs represent direct relations between these variables. The representation of an ARMA model contains both stochastic and deterministic nodes, represented by respectively single- and double-circles. A node is deterministic if the value of the variable represented by that node is a deterministic function of the values for variables represented by nodes pointing to that node in the graphical representation. Otherwise, the node is stochastic. From the definition of the ARMA models, we see that the observable variables (the Y's) are represented by deterministic nodes and the error variables (the E's) are represented by stochastic nodes. The relations between variables are defined by (1) and accordingly, $Y_{t-p}, \ldots, Y_{t-1}$ and $E_{t-q}, \ldots, E_t$ all point to Y_t. We are interested in the *conditional likelihood models*, where we condition on the first $R = max(p, q)$ variables. Relations between variables for $t \leq R$ can therefore be ignored. It should be noted that if we artificially extend a time series back in time for R (unobserved) time steps, this model represents what is known in the literature as the *exact likelihood model*. There are alternative methods for dealing with the beginning of a time series (see, e.g., Box, Jenkins, and Reinsel [4]).

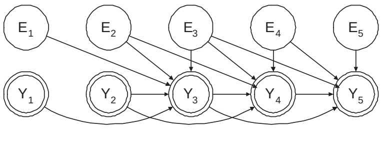

Figure 1: ARMA(2,2) model.

2.2 σARMA Models

A σARMA model is identical to an ARMA model except that the deterministic relation in (1) is replaced by a conditional Normal distribution (with variance σ, thereby the name). The graphical representation for a σARMA model is therefore likewise similar to the ARMA model, the only difference being that deterministic nodes are now stochastic.

More specifically, $Y_t | E_{t-q}^{t}, Y_{t-p}^{t-1} \sim \mathcal{N}(\mu, \sigma)$, where the functional expression for the mean μ and the variance σ are shared across the observation variables. The variance is fixed at a given (small) value to be specified by the user. The mean is related to the regressor variables as follows

$$\mu = \zeta + \sum_{j=0}^{q} \beta_j E_{t-j} + \sum_{i=1}^{p} \alpha_i Y_{t-i} \tag{3}$$

We see from this representation that ARMA is the limit of σARMA as $\sigma \to 0$. Thiesson *et al.* [5] also shows that σ becomes a minimum allowed variance for the one step-ahead forecast and in this way takes the role as a controllable smoothing parameter.

By fixing $\beta_0 = 1$ in the ARMA model, the variance γ of E_t has the semantic of being the variance for the one-step forecast. With the additional smoothing, γ does not carry the same semantic for a σARMA model. Without this semantic for γ, it seems natural to consider a variation of the σARMA model class that lets β_0 vary freely. Yet another variation will let σ vary freely.

Our implementation of the of σARMA models allows for all four variations, where the parameters β_0 and σ can be fixed or vary freely during parameter estimation and model selection. Notice, however, that by letting σ vary freely we have a more flexible model, but we lose the ability to control the smoothing.

2.3 σARMA^{xp} Models

The σARMA^{xp} class of models includes the following generalizations of the σARMA class: (1) a model may define multiple time series—with different p's and q's in different time series, and (2) a time series is allowed additional dependencies on observations from related time series, called cross-predictors (the 'xp' in the name).

Consider the part of a σARMA^{xp} model which describes a particular time series in the set of time series defined by the model. The representation of this time series is similar to an σARMA model, except that Y_t additionally depends on the vector of cross predictors C_t. Let η be a vector of regression coefficients associated with these cross predictors. In this case

$Y_t | E_{t-q}^t, Y_{t-p}^{t-1}, C_t \sim \mathcal{N}(\mu, \sigma)$ with mean

$$\mu = \zeta + \sum_{j=0}^{q} \beta_j E_{t-j} + \sum_{i=1}^{p} \alpha_i Y_{t-i} + \eta C_t \tag{4}$$

The graphical representation of an σARMA^{xp} model is shown in Figure 2. The model represents three time series. The first time series (for observations Y) corresponds to an σARMA(2,2) model with no cross-predictors, the second time series (for observations Y') corresponds to a σARMA(1,1) model, where Y_{t-1} is a cross predictor for Y_t', and the third time series (for observations Y'') corresponds to a σARMA(1,0) model with Y_{t-1}' being a cross predictor for Y_t''.

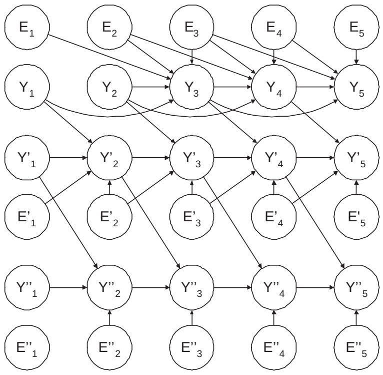

Figure 2: σARMA^{xp} for three cross predicting time series.

Notice that the cross predictors for a time series separates all observation and error variables for that time series from the remaining time series in the model. Given full observations on the cross predictors, the distributions for the individual time series in the model are therefore independent, which is an important characteristic for a σARMA^{xp} model. We will exploit this independence in approximations yielding computationally efficient learning and predictions, as described later in this paper.

Finally, it is worth mentioning that besides the stochastic nature of the σARMA^{xp} models, these models are different from vector ARMA models (see, e.g., Reinsel [10]). First of all, different time series in a σARMA model may have different numbers of AR and MA regressors, and second, cross predictors between time series are not defined by the ARMA structure, but are chosen in a selective way for each time series in the model (see Section 4).

3 ML Estimation

We will need to define a few notational conveniences before presenting the ML estimate for a σARMA^{xp} model. Consider, again, a part of the model, defining only a particular time series. Let $\alpha = (\alpha_1, \ldots, \alpha_p)$ and $\beta = (\beta_1, \ldots, \beta_q)$ denote the vectors of AR and MA regression coefficients for that time series. The parameters in this part of the model are therefore given by $\theta_s = \{\zeta, \alpha, \beta_0, \beta, \eta, \sigma, \gamma\}$, where β_0 and σ are fixed at a particular value, and may be free parameters for the variations, mentioned in Section 2.2. Notice that except for θ_s we have eased the notation a bit by suppressing the relation to the particular time series (here denoted s)—but we stress that the set of parameters and their values may be different for the different time series in the model. Similarly, we suppress the relation to the particular time series when referring to stochastic variables associated with that time series. Finally, the distribution for the entire σARMA^{xp} model is represented by the collective set of parameters for all time series in the model $\theta = \times_s \theta_s$, and we let θ^* represent the associated posterior distribution that conditions on all observations in the model.

3.1 EM algorithm

It turns out that the log-likelihood for a σARMA^{xp} model is difficult to maximize directly due to the latent error variables and otherwise potential incomplete data in the statistical model. Thiesson *et al.* [5] and Thiesson and Meek [6] demonstrate how the ML estimate in this case can be obtained via simple iterative estimation algorithms, such as the EM algorithm and gradient based optimization algorithms, respectively. Let us concentrate on the EM algorithm in the following.

Roughly speaking, the EM algorithm converts the ML estimation problem into an converging iterative sequence of parameter updates. Let $\mathbb{E}_{\theta^*}$ denote the expectation with respect to the posterior distribution defined by θ^*. Following Thiesson *et al.* [5] each iteration in the EM algorithm then updates all free parameters in the model by the following iterations.

3.1.1 σARMA^{xp}

For any individual time series in the model, The parameter for the error variance is at each iteration updated as the expected sample variance. That is,

$$\gamma \leftarrow \sum_t \mathbb{E}_{\theta^*}\left[E_t^2\right] / (T - R) \qquad (5)$$

Further, let $X_t = (E_{t-q}^{t-1}, Y_{t-p}^{t-1}, C_t)$ and let $\phi = (\beta, \alpha, \eta)$ denote the associated regression coefficients between these variable and Y_t for that particular time series. The remaining free parameters in the model, (ζ, ϕ), are updated by solving the following system of equations

$$\sum_t \boldsymbol{E}_{\theta^*}\left[X_t^\top X_t\right]\phi^\top + \sum_t \boldsymbol{E}_{\theta^*}\left[X_t^\top\right]\zeta = \sum_t \boldsymbol{E}_{\theta^*}\left[Y_t X_t^\top\right] - \sum_t \boldsymbol{E}_{\theta^*}\left[X_t^\top E_t\right]$$

$$\sum_t \boldsymbol{E}_{\theta^*}\left[X_t\right]\phi^\top + \sum_t \zeta = \sum_t \boldsymbol{E}_{\theta^*}\left[Y_t\right] - \sum_t \boldsymbol{E}_{\theta^*}\left[E_t\right]$$

where $\top$ denotes transpose.

This set of equations can be singular (or close to singular) so one should be careful and use a method robust to this situation when solving the equations. We use pseudo-inversion to extend the notion of inverse matrices to singular matrices when solving the equations (see, e.g., Golub and Van Loan [11]).

Finally, by realizing that the σARMA^{xp} model defines a Bayes net, the expected sufficient statistics involved in the above ML estimation can be efficiently computed by the inference technique, described in Lauritzen and Jensen [12] or any other method for inference in Gaussian Bayes nets.

3.1.2 σARMA^{xp} variations

In a variation of the σARMA^{xp} model we may include β_0 as one of the free parameters in the model. Therefore this time, let $X_t = (E_{t-q}^t, Y_{t-p}^{t-1}, C_t)$ and $\phi = (\beta_0, \beta, \alpha, \eta)$. The update for γ is the same as above, but the update for the parameters (ζ, ϕ) is now obtained by solving a slightly different set of equations

$$\sum_t \boldsymbol{E}_{\theta^*}\left[X_t^\top X_t\right]\phi^\top + \sum_t \boldsymbol{E}_{\theta^*}\left[X_t^\top\right]\zeta = \sum_t \boldsymbol{E}_{\theta^*}\left[Y_t X_t^\top\right]$$

$$\sum_t \boldsymbol{E}_{\theta^*}\left[X_t\right]\phi^\top + \sum_t \zeta = \sum_t \boldsymbol{E}_{\theta^*}\left[Y_t\right]$$

In yet another variation, we may (in addition) let σ vary freely in the model. The update for σ is in this case obtained by first computing the expected joint sample covariance matrix for target and regressor variables $Z_t = (Y_t, X_t)$ as

$$Cov(Z_t, Z_t) = \sum_t \boldsymbol{E}_{\theta^*}\left[Z_t^\top Z_t\right]/(T-R) - \sum_t \boldsymbol{E}_{\theta^*}\left[Z_t^\top\right]\sum_t \boldsymbol{E}_{\theta^*}\left[Z_t\right]$$

and then derive the conditional variance for the target

$$\sigma \leftarrow Cov(Y_t, Y_t) - Cov(Y_t, X_t)Cov(X_t, X_t)^{-1}Cov(X_t, Y_t)$$

3.2 Approximation

It is important to notice that the posterior distribution for the entire set of time series in the σARMA^{xp} model is involved in calculating the expected values used for the above parameter updates.

The time series are coupled through cross predicting observation variables (the Y's). Observations from one time series may therefore influence another time series, either directly as a cross predictor for the time series or indirectly by propagating through an un-observed cross predictor. For instance, if Y'_3 is not observed in Figure 2, the observation for Y_2 will affect the Y'' time series. This situation can be handled by the inference technique in Lauritzen and Jensen [12] but can become too complex and computational expensive for our purpose.

Readers familiar with graphical models may appreciate that a particular complex situation appears if a time series has two or more cross predictors from the same time series. For example, let us consider a situation where time series Y has cross predictors from time series Y' with lags 1 and 12.— Not an un-realistic situation if data is monthly and cyclical by a year. In this case the cross predictors creates long un-broken cycles in the graphical representation of the model, $Y'_{t-12} \rightarrow Y'_{t-11} \rightarrow \cdots \rightarrow Y'_{t-1} \rightarrow Y_t \leftarrow Y'_{t-12}$ (we ignore direction of arcs). By standard junction tree construction tecniques these cycles are triangulated, but leaves the graphical structure used for inference in the Lauritzen and Jensen (2001) algorithm very complex.

On the other hand, if we have complete data for the cross predictors, then the situation is different. In this case the observed cross predictors separate the time series completely from other time series in the model, in effect de-coupling the posterior distribution into independent distributions for each time series. Hence, θ^* can be replaced by θ^*_s for all expectations in the parameter update formulas above. The difference in the formulas is subtle but the effect of being able to handle each time series as an individual σARMA time series can be significant for the computational efficiency of the estimation procedure.

Computational efficiency is of great concern for our implementation. If cross predictors for a time series are not completely observed we will therefore fill-in the missing values before estimation. Notice that we still handle incomplete data for the time series under consideration. We are therefore still partly handling incomplete data in a theoretical sound way by exploiting the EM algorithm.

To fill-in missing values for cross predictors, we use a simple linear interpolation from the two observed values at each side of the missing observation. We use extrapolation from the two nearest observations, if there are no observation on one of the sides. Other fill-in procedures are possible.

3.3 Time trends

We can extend the σARMA^{xp} model to handle simple linear trends for any of the time series in the model. It is achieved by including an artificial time series for time and then allowing the remaining time series in the model to have this time series for time as a cross predictor with zero lag. Having this extra time cross predictor will add $\eta_t f(t)$ as an extra term for the mean in

(4), were $f(t)$ denotes the time step and η_t the coefficient for the trend.

The time cross predictor is always observed so any real time series under consideration is completely separated from this artificial time serie; except through the observed cross predictors. We can therefore ignore learning a model for this artificial time series for time.

Dependence on time is not restricted to linear trends. The values $f(t)$ used as the artificial observations for time can be any function of time. Figure 3 shows the graphical representation for a σARMA(1,1) model with time trend $\eta_1 t + \eta_2 t^2$, where we have represented the observation variables for the artificial time series for time by their observed values.

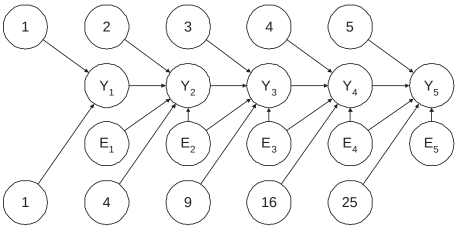

Figure 3: σARMA(1,1) model for time series with $\eta_1 t + \eta_2 t^2$ trend.

3.4 Periodicity

Periodicity is theoretically not a problem for the estimation algorithm. Periodicities can, however, introduce large (un-broken) cycles in the graphical representation for the model, which as mentioned above, creates a situation that in not computationally attractive. For example, Figure 4a shows the graphical representation for a σARMA(1,0) model with periodicities 4 and 12. We see the cycles $Y_{t-12} \rightarrow \cdots \rightarrow Y_{t-4} \leftarrow Y_{t-12}$ and $Y_{t-4} \rightarrow \cdots \rightarrow Y_t \leftarrow Y_{t-4}$.

To overcome the computational complexity we instead handle periodicities as cross predictors. For a time series with periodicity we add a copy of that time series to the σARMA^{xp} model and then replace the regressions created by periodicities by cross predictors from this added artificial time series in the model. The added time series is ignored during estimation, except as cross predictor for the original time series in question. Figure 4b shows our alternative representation for the model in Figure 4a.

If the observations in a time series with periodicities are complete, the posterior distribution used for the parameter updates during estimation are the same for both representations of the model. Both representations therefore yield the same parameter estimate in this case.

It is another matter for time series with incomplete observations. Let P_t denote the periodicity regressors in an σARMA(p,q) model at time t. For

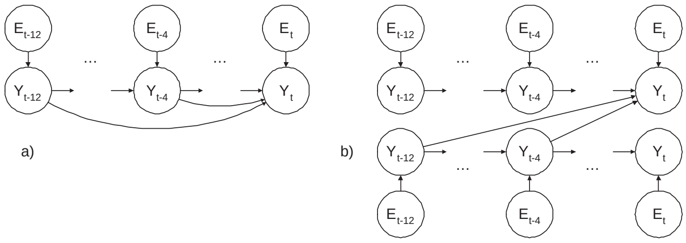

Figure 4: a) σARMA(1,1) model with periodicities 4 and 12, and b) alternative, approximate representation. Only periodicity regressors to the X_t variable are shown.

incomplete data, the alternative representation is an approximation, which factorizes the conditioning part of the conditional normal distribution defining the model for $Y_t|E^t_{t-q}, Y^{t-1}_{t-p}, P_t$ into a non-periodicity part and a periodicity part. That is, $p(E^t_{t-q}, Y^{t-1}_{t-p}, P_t) \approx p(E^t_{t-q}, Y^{t-1}_{t-p})p(P_t)$. The posterior distribution used for the parameter updates is therefore an approximation.

As for any cross predictor, we fill-in missing values for a periodicity cross predictor in order to de-couple the two time series and thereby keep the parameter estimation computational efficient. The time series under consideration is still allowed to have incomplete data, but filling in missing values for periodicity cross predictors, of course, adds to the approximate nature of the alternative implementation.

4 Model selection

As discussed in Section 3.2, all time series are de-coupled in our implementation through the observed—or filled-in— cross predictors. A σARMA model with cross predictors can therefore be learned separately for each individual time series in the σARMA^{xp} model. It is impossible to perform an exhaustive search for a such model. Hence, our implementation apply a greedy model selection strategy.

The selection criterion that we use for comparing alternative models during the search allows the use of either a BIC score or a predictive accuracy score for holdout data.

The selection strategy involves a forward-backward search when deciding which models to evaluate during the model selection. Starting from an initial σARMA(p,q) model – typically σARMA(0,0)– the forward search will first greedily add AR and MA regressors before cross predictors are added. The backward search will thereafter greedily try to remove AR and MA regressors before any of the selected cross predictors are removed.

To stay within the class of σARMA^{xp} models, all AR and MA regressors up to p and q, respectively, must be included in the model. The greedy forward and backward searches for AR and MA regressors therefore, respectively, just adds or subtracts one from either p or q at each step in the search.

For computational efficiency reasons, the greedy forward search for cross predictors is a little more complicated. We first pre-order a set of most promising cross predictors according to the selection criterion score for a model with $p = q = 0$ and a cross predictor. Each step in the greedy search then investigates cross predictors in this order and includes the first cross predictor that is found to improve the current model. The greedy search stops if none of the cross predictors in the pre-ordered list improves the current selected model. As the complexity of the model increases it becomes computationally more expensive to investigate cross predictors. For that reason, the set of pre-ordered cross predictors to investigate decreases with the complexity of the current selected model during the forward search. The backward search for cross predictors investigates the cross predictors in a standard greedy fashion by considering all remaining cross predictors at each step in the greedy search.

As mentioned in Subsections 3.3 and 3.4 the implementation handles time trends and periodicities by the means of cross-predictors.

By including artificial time series for different time trends in the class of selectable models, we can use the model selection method to select cross predictors corresponding to the appropriate time trend or combination of time trends that may be present for any of the real time series in the model.

Similarly, for each individual time series that we consider during the model selection, we can let the procedure select periodicities in the form of cross predictors from an extra artificial copy of that time series, as described in Section 3.4. As an alternative—or supplement—one could also choose to apply a method such as the Fourier analyses suggested in Vlachos *et al.* [13] in order to rank and limit the number of periodicy cross predictors to be investigated during the model search.

5 Forecasting

We now consider the problem of using a σARMA^{xp} model to forecast. For a given time series in the model, the task of forecasting is to calculate the marginal distributions for future observations given previous observations for all time series in the model. That is, the distributions $p(Y_{T+r}|\mathbf{y})$, were $r = 1, \cdots, R$, and $\mathbf{y}$ denotes the observations for all time series in the model.

Consider first the simpler situation of forecasting one step into the future. In this case we extend the model by an additional time step. For any time series in the model, the regressor variables $(E_{T-q+1}^{T}, Y_{T-p+1}^{T}, C_{T+1})$ separates the added variables (E_{T+1}, Y_{T+1}) from the remaining variables in the model. See, for example, Figure 2, where (E'_5, Y'_5) is separated from the remaining variables by (E'_4, Y'_4, Y_4). The joint posterior distribution for the prediction variable and all its regressors therefore factorizes as

$$p(E_{T-q+1}^{T+1}, Y_{T-p+1}^{T+1}, C_{T+1}|\mathbf{y}) =$$
$$p(E_{T+1}, Y_{T+1}|E_{T-q+1}^{T}, y_{T-p+1}^{T}, c_{T+1})p(E_{T-q+1}^{T}, y_{T-p+1}^{T}, c_{T+1}|\mathbf{y}) \quad (6)$$

Recall that some of the observations y_{T-p+1}^{T} and c_{T+1} may be missing in which case these are treated as stochastic variables rather than observations on the right-hand side of (6). We can now obtain the predictive distribution by simply marginalizing the posterior Normal distribution in (6) to the prediction variable Y_{T+1}.

Let us consider the distributions involved on the right-hand side of (6) in more detail. The distribution $p(E_{T+1}, Y_{T+1}|E_{T-q+1}^{T}, y_{T-p+1}^{T}, c_{T+1})$ is computed by inserting the observations y_{T-p+1}^{T} and c_{T+1} into the distribution $p(E_{T+1})p(Y_{T+1}|E_{T-q+1}^{T+1}, Y_{T-p+1}^{T}, C_{T+1})$, as obtained from the definition of the time series in Section 2.3. The distribution $p(E_{T-q+1}^{T}, y_{T-p+1}^{T}, c_{T+1}|\mathbf{y})$ can in preparation of the forecast be pre-computed by the general Bayes net inference algorithm in Lauritzen and Jensen [12]. In fact, the computation of the joint distribution in (6) corresponds to a simple (forward) inference step in this algorithm.

When forecasting more than one step into the future we add multiple time steps—with un-observed values—to the model. Multi-step forecasting can in this way be computed in a similar way as one-step forecasting by using the Bayes net inference multiple steps forward instead of just one step.

5.1 Approximation

Recall from Subsection 3.2 that the coupling of time series through un-observed cross predictors can cause the general inference scheme to become complex and therefore computational expensive. Fast forecasting is a concern for our implementation, and for that reason we consider instead the following approximation.

In spirit with the approximation used for estimation, we will during forecasting de-couple the individual time series in the model in the sense that we never consider joint distributions that involves un-observed variables from multiple time series. During estimation, this de-coupling was achieved by filling-in (hard) values for cross predictors. For forecasting, however, we will fill-in soft values in the sense that the problematic part on the right-hand side of (6) is approximated by

$$p(E_{T-q+1}^{T}, y_{T-p+1}^{T}, c_{T+1}|\mathbf{y}) \approx p(E_{T-q+1}^{T}, y_{T-p+1}^{T}|\mathbf{y}, c_{T+1}) \prod_{i} p^*(c_{T+1}^{i}), \quad (7)$$

where c_{T+1}^{i} denotes the individual cross predictor values in the vector c_{T+1}, and $p^*(c_{T+1}^{i})$ is one, if c_{T+1}^{i} is an actual observed value, and otherwise it is the approximate marginal posterior distribution for c_{T+1}^{i}, as computed for that cross predicting time series. Notice that cross predictors always refer to variables in earlier time steps than the time step for which the forecast is performed. It is therefore always possible to compute the posterior marginal distribution $p^*(c^i_{T+1})$ before it is needed in the above approximation.

An approximate multi-step forecast can be performed by recursively applying (6) with the approximation in (7).

6 Evaluation

An empirical evaluation regarding the quality of predictions for σARMA^{xp} models can be found in Thiesson *et al.* [5]. That paper recommends $\sigma = 0.01$ as a good choice for the smoothing parameter, and shows that the choice of β_0 set as either a fixed or free parameter does not affect the prediction quality. We verified—on a few sporadic examples—that also fixed versus free σ does not affect the quality of predictions. We will not pursue further experiments regarding the prediction quality here. Instead, we will provide an empirical evaluation regarding the computational consequences of these parameter choices for the learning of a σARMA^{xp} model.

For the experiments we used synthetic data generated from the σARMA^{xp} model in Figure 2, an ARMA(2,2) model, an ARMA(4,1) model, and an ARMA(1,4) model. For each of these models we generated data from three different parameterizations, yielding 12 different time series data sets; each with 1000 observations. We also created—for each of the 12 data sets—four corresponding incomplete data sets with respectively 5, 10, 20, and 40 percent of randomly missing observations.

We used the EM algorithm to estimate models with graphical structure corresponding to the generating models for the data sets. We evaluated all four combinations of β_0 and σ as fixed or free parameters during estimation, and we repeated all experiments with the smoothing parameter initialized as $\sigma = 0.001, 0.002, \cdots, 0.01, 0.02, \cdots, 0.1$.

Let us first consider what happens when σ approaches zero regarding the number of EM iterations before convergence and regarding the overall runtime. Figures 5a and 5b shows—for a representable example—the number of iterations and the runtime (on a Pentium M 1.7 GHz Tablet PC) needed to estimate a model as a function of σ and amount of missing data. The results shown in the figures are for estimating the σARMA^{xp} model with fixed β_0 and σ. Figures for all other experiments are similar, except for a few experiments where large amounts of missing data force estimation to converge at incomparable local maxima.

Thiesson *et al.* [5] argues that the EM algorithm will only work for $\sigma > 0$. Figures 5a and 5b show that the convergence rate for EM decreases with smaller values of σ, and, in fact, suggest that the algorithm comes to a complete halt when $\sigma = 0$. The figures also illustrate that the number of iterations and the runtime increases with the amount of missing data—not a surprise, but rather a known fact about the EM algorithm in general. The

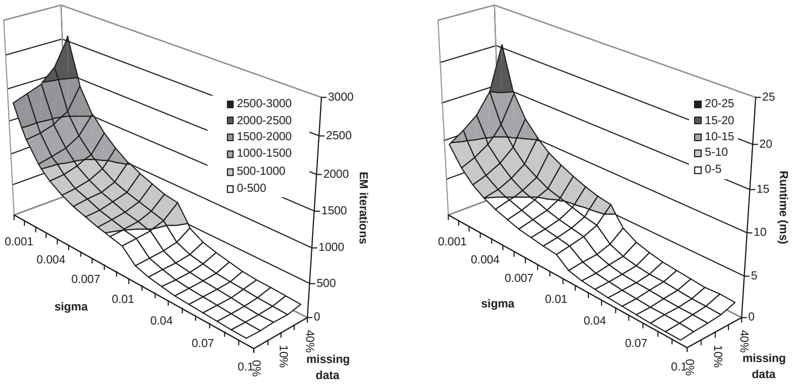

Figure 5: a) Number of EM iterations and b) runtime for model estimation as a function of σ and amount of missing data.

recommended value of $\sigma = 0.01$ seems as a computational reasonable choice. The user should, of course, base this choice of σ not only on computational efficiency concerns, but also the characteristics of the problem under investigation, because this parameter—for fixed σ models—offers a controlled way of smoothing the prediction model compared to a classic ARMA model.

Across all experiments we saw no significant systematic difference in the number of EM iterations needed to estimate a model for the four different variations of models with β_0 and σ fixed or free. We therefore recommend fixing σ in order to control the smoothing variance. Also, by fixing σ we ensure that this parameter does not incidentally move towards zero during the estimation iterations and thereby create the computational unattractive situation, mentioned above. We suspect that this scenario could happen, but we did not see it for any of the experiments that we performed.

Arguably, we saw that estimating parameters for models with fixed β_0 is slightly more expensive than for models with free β_0. However, if it is important to learn a stochastic ARMA model variation most resembling a classic ARMA model, there is no significant computational argument against fixing β_0.

7 Future work

Computational efficiency was an important concern for the implementation, described in this paper. We therefore chose to handle cross predictors and periodicities in an approximate way during both model estimation and predictions. These approximations will not affect models learned with completely observed time series data, but for un-observed cross predictors these approximations will have some effect. We would like to investigate the effect on the prediction quality, but it demands an implementation of full inference for cross-predicting time series, which we do not yet have.

Finally, we plan to investigate if any adjustments can be done to the EM estimation procedure in order to speed up estimation for σ approaching zero.

References

[1] T.L. Dean and K. Kanazawna. Probabilistic temporal reasoning. Technical report, Brown University, 1988.

[2] G.F. Cooper, E.J. Horvitz, and D.E. Heckerman. A model for temporal probabilistic reasoning. Technical report, Stanford University, 1988.

[3] F.V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Verlag, New York, 2001.

[4] G.E.P Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis*. Prentice Hall, New Jersey, 1994.

[5] B. Thiesson, D.M. Chickering, D. Heckerman, and C. Meek. ARMA time-series modeling with graphical models. In *Proc. of the Twentieth Conf. on Uncertainty in Artificial Intelligence*, pages 552–560. AUAI Press, 2004.

[6] B. Thiesson and C. Meek. Efficient gradient computation for conditional gaussian models. In *Proc. of Tenth Int. Workshop on Artificial Intelligence and Statistics*. The Society for Artificial Intelligence and Statistics, 2005.

[7] Z. Ghahramani. Learning dynamic bayesian networks. In *Adaptive Processing of Sequences and Data Structures. Lecture Notes in Artificial Intelligence*, pages 168–197. Springer-Verlag, Berlin, 1998.

[8] C. Meek, D.M. Chickering, and D. Heckerman. Autoregressive tree models for time-series analysis. In *Proc. of the Second Int. SIAM Conf. on Data Mining*, pages 229–244. SIAM, 2002.

[9] SQL Server Data Mining website, managed by the development team at Microsoft Corporation. www.SQLServerDataMining.com.

[10] G.C. Reinsel. *Elements of Multivariate Time Series Analysis*. Springer-Verlag, New York, 2003.

[11] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, London, 1996.

[12] S.L. Lauritzen and F. Jensen. Stable local computation with conditional gaussian distributions. *Statistics and Computing*, 11:191–203, 2001.

[13] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proc. of Int. Conf. on Management of Data SIGMOD 2004*. ACM, 2004.