VAR modeling, regularization and the SparseTSCGM package

Stijn de Vos

03 - 11 - 2017

Contents

We start with a small review of vector-autoregressive models. Afterwards, regularization is discussed and the SparseTSCGM package is demonstrated.

Introduction

Suppose we have four items called 'Positive Affect' (x_1), 'Anhedonia' (x_2), 'Irritability' (x_3) and 'Weightloss' (x_4). Measuring these items multiple times gives a time-series dataset where each observation consists of four elements. Written mathematically, the observation at time t consists of four numbers $(x_{1t}, x_{2t}, x_{3t}, x_{4t})$. As a shorthand notation, people often write $X_t = (x_{1t}, x_{2t}, x_{3t}, x_{4t})$. X_t is called a (four-dimensional) *vector*.

This is why it's called **vector**-autoregressive (VAR) modeling; we regress a vector X_t on past 'versions' of itself, instead of a single variable. The simplest VAR model only regresses X_t on its previous measurement X_{t-1}. This VAR model is said to have a **lag** of 1. In formula form, this model looks as follows:

$$\begin{aligned}
x_{1t} &= a_{11}x_{1(t-1)} + a_{12}x_{2(t-1)} + a_{13}x_{3(t-1)} + a_{14}x_{4(t-1)} + \epsilon_{1t} \\
x_{2t} &= a_{21}x_{1(t-1)} + a_{22}x_{2(t-1)} + a_{23}x_{3(t-1)} + a_{24}x_{4(t-1)} + \epsilon_{2t} \\
x_{3t} &= a_{31}x_{1(t-1)} + a_{32}x_{2(t-1)} + a_{33}x_{3(t-1)} + a_{34}x_{4(t-1)} + \epsilon_{3t} \\
x_{4t} &= a_{41}x_{1(t-1)} + a_{42}x_{2(t-1)} + a_{43}x_{3(t-1)} + a_{44}x_{4(t-1)} + \epsilon_{4t}
\end{aligned}$$

The numbers a_{ij} are the regression coefficients of the VAR model. The terms ϵ_{it} represent the (contemporaneous) noise at time t. VAR models usually assume that ϵ_{it} is normally distributed with mean 0 and some standard deviation: $\epsilon_{it} \sim \mathcal{N}(0, \sigma_i)$.

This all looks very nasty and complicated. Luckily, using vector notation, we can abbreviate the model description quite a bit. Using the vector notation we can write

$$\overbrace{\begin{bmatrix} x_{1t} \\ x_{2t} \\ x_{3t} \\ x_{4t} \end{bmatrix}}^{X_t} = \overbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} x_{1(t-1)} \\ x_{2(t-1)} \\ x_{3(t-1)} \\ x_{4(t-1)} \end{bmatrix}}^{X_{t-1}} + \overbrace{\begin{bmatrix} \epsilon_{1t} \\ \epsilon_{2t} \\ \epsilon_{3t} \\ \epsilon_{4t} \end{bmatrix}}^{\epsilon_t}$$

Or, even more convenient:

$$X_t = AX_{t-1} + \epsilon_t$$

You will have to take my word for it that this represents the same system of equations as written earlier. If you don't, or feel particularly bored, you can try looking up the Wikipedia page on matrix operations and confirm it for yourself!

The vector ϵ_t is comprised of four random variables, each with a normal distribution. Another way of saying this is that ϵ_t is distributed according to a *multivariate* normal distribution with mean vector $(0, 0, 0, 0)$ and covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} \end{bmatrix}$$

Note that when we include more lags in the model, we have multiple regression coefficient matrices. For example, a lag-3 model looks like this:

$$X_t = A_1X_{t-1} + A_2X_{t-2} + A_3X_{t-3} + \epsilon_t$$

For the sake of easy exposition, we stick to the lag-1 model. The most important important sets of parameters of this model are

- A, the matrix of regression coefficients, representing the associations between specific items over time;
- Σ, the covariance matrix, representing variability of contemporaneous noise at each measurement.

When we fit a VAR model, we are trying to find values for the elements of these matrices.

What is regularization?

Before discussing regularization, we briefly reconsider the likelihood function and maximum likelihood estimation. Suppose we have a dataset of observations of independent variables x_1, x_2 and a response variable y. A simple linear regression model looks like this:

$$y = ax_1 + bx_2 + \epsilon.$$

Maximum Likelihood Estimation (MLE) is a typical approach to estimating the model parameters a, b. Likelihood is a function of the model parameters and is defined as $L(a, b|X) = P(X|a, b)$, which means that the *likelihood of model parameters a and b* is defined as the probability of observing the data X, given some values for a and b. MLE finds estimations for the model parameters by calculating for which a and b the likelihood function is maximized.

Now suppose that we have a regression problem where there are more variables in the model than there are observations. This is a problem, since there is not enough information available to estimate all the parameters in the model. In this case, it is hard or impossible to find the best value for a and b based on MLE, i.e. it is not possible to find values for a and b such that the likelihood function $L(a, b|X)$ is maximized. This is one of the prime examples where regularization saves the day. Instead of estimating the model parameters by maximizing the likelihood function, we add an extra term to the likelihood function:

$$L_{\text{pen}}(a, b|X) = L(a, b|X) + \lambda \text{Pen}(a, b).$$

Here, *Pen* is short for **penalty**. This is a mathematical function that gets larger if the model parameters get bigger. Another way of thinking about regularization in this context is that we add extra information that allows us to still estimate model parameters, namely, the information that the model parameters should not be 'too big'. What 'too big' means depends on the penalty function that is used. In any case, the model estimation procedure now continues as before; the estimates for parameters a and b are those numbers for which the penalized likelihood function is maximized.

The number λ is a parameter that expresses how importart or stringent the penalty function is. If λ is large, then a small increase in penalty is amplified and the resulting estimators for model parameters a, b are smaller. An immediate question is, then: how do we pick λ? It is a model parameter in the sense that we have to find a 'best' value for it. On the other hand, we are not really interested in a particular value for λ, we just want to pick it optimally so we can estimate a and b. For this reason, a and b are sometimes in the literature referred to as *structural parameters* and λ is referred to as a *nuisance parameter*. Another term is 'hyperparameter'. Some common methods for finding a good value for λ are information-theoretical approaches such as AIC, BIC (as we'll see in the SparseTSCGM package) or cross-validation.

Examples of common regularization variants

- The **LASSO** (least absolute shrinkage and selection operator) operator: one of the most common regularization variants. It is named the LASSO because it performs shrinkage (another word for regularization) and it does so by considering the absolute values of model parameters. The penalty function here is defined as (sticking to our simple regression example)

$$\text{Pen}(a, b) = |a| + |b|$$

  In some technical literature you may see the notation $\text{Pen}(a, b) = ||\beta||_1$ (where β represents all model parameters except λ) which is also known as the L_1 penalty.
- **Ridge regression**: here, the function is

$$\text{Pen}(a, b) = \sqrt{a^2 + b^2}$$

  which is also known as $\text{Pen}(a, b) = ||\beta||_2$, the L_2 penalty.
- **Elastic-net regularization**: this regularization method uses a linear combination of the previous two penalty functions :

$$\text{Pen}(\beta) = \lambda_1||\beta||_1 + \lambda_2||\beta||_2$$

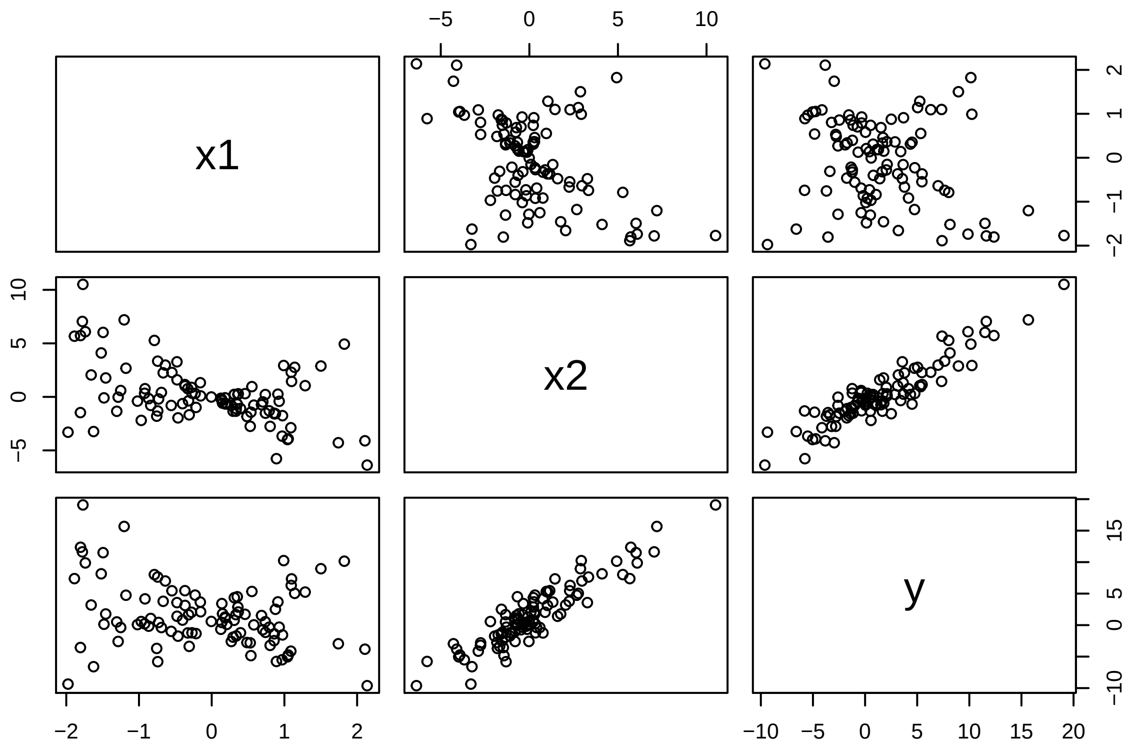
  and is thus a generalization of the previous regularization methods. If $\lambda_1 = 0$ then elastic-net regularization is just ridge regression, if $\lambda_2 = 0$ then we recover the LASSO penalty function. The R package *glmnet* implements elastic-net regularization, although the penalty is rewritten so that the method only has to consider a single regularization parameter instead of two.

Simple versus regularized regression

As an illustration, here is vanilla linear regression versus regularized regression applied to a toy example:

```r
library(glmnet)

# Not fake news, but fake data:
x1 <- rnorm(100)
x2 <- rnorm(100, mean = -1, sd = 3)*x1
y <- 1 + x1 + 2*x2 + rnorm(100, sd = 2)
data <- data.frame(x1 = x1, x2 = x2, y = y)
pairs(data)
```



```r
# simple linear regression without regularization:
simple_lm <- lm(y ~ x1 + x2)
coefficients(simple_lm)
```

```
## (Intercept)          x1          x2
##    1.214680    1.039879    1.831656
```

```r
# regression using elastic-net regularization:
cv <- cv.glmnet(x = cbind(x1, x2), y = y)

# This is the optimal value for lambda according to cross-validation:
cv$lambda.min
```

```
## [1] 0.01727381
```

```r
# fitting a penalized regression model using the optimal value for lambda:
penalized_lm_opt <- glmnet(x = cbind(x1, x2), y, lambda = cv$lambda.min)
coefficients(penalized_lm_opt)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                   s0
## (Intercept) 1.214570
## x1          1.006630
## x2          1.819863
```

We can see that in this case, the changes in estimated model parameters are minimal.

The SparseTSCGM package

The SparseTSCGM package (Abegaz and Wit 2013) combines VAR models with regularization. The main features of the package are the functions `sparse.tscgm()`, which fits the actual regularized VAR model, and `sim.data()` which simulates data according to a given network structure. There is also a `print()`, `plot()` and `summary()` function for modeling results.

This package allows one to regularize a VAR model in two different aspects. Firstly, it regularizes the matrix of regression coefficients (A in the earlier example, Γ in Abegaz et al.). Secondly, it regularizes the variance-covariance matrix of the contemporaneous errors. There is a small difference between this package and the VAR example above; instead of estimating the variance-covariance matrix Σ, it estimates its *inverse*, written $\Theta = \Sigma^{-1}$ (also known as the *precision matrix*). The reason for this is not directly related to this package. It turns out that, from a network perspective, Θ encodes the *conditional independence structure* of the variables in the model. For a technical exposition, see e.g. Lauritzen (1996). It doesn't really matter though; one can always invert the precision matrix to obtain the variance-covariance matrix, or transform it to a partial correlation matrix.

Usage

The function `sparse.tscgm()` accepts the following arguments:

- `data`: the data to which the VAR model is fitted.
- `lam1`: the hyperparameter for the regularization of the precision matrix of the contemporaneous noise. The default value is `NULL`, which means that the program generates a sequence of possible values and tries to find the optimal value according to the `optimality` argument.
- `lam2`: the hyperparameter for the regularization of the regression coefficient matrix. The default is `NULL`, with the same behaviour as `lam1`.
- `nlambda`: determines the length of the generated sequence of possible hyperparameter values if `lam1` or `lam2` is `NULL`,
- `model`: valid choices are `ar1` (the default setting) for a lag-1 model or `ar2` for a lag-2 model. Higher lags are not supported at the moment.
- `penalty`: selects the penalty function that is used for the precision and regression coefficient matrix. Possible choices are `lasso` for the LASSO penalty function and `scad` (Smoothly Clipped Absolute Deviation, the default setting), another penalty function.
- `optimality`: determines by which information-theoretical measure the hyperparameters `lam1` and `lam2` should be selected. Possible choices are `aic`, `bic`, `bic_ext` (extended Bayesian Information Criterium), `bic_mod` (Modified Bayesian Information Criterium), `gic` (Generalized Information Criteria). It can also be `NULL`, in this case the program expects two values for `lam1` and `lam2`.
- `control`: accepts an optional list of settings for the technical, behind-the-scenes estimation procedure (not discussed here).

An obvious question is why one would choose one penalty function over the other. Unfortunately, it is pretty hard to give a one-size-fits-all answer to this question. Each penalty function has its technical advantages and downsides. Luckily, the result usually does not differ too much for different penalty functions (but you should always confirm this is the case).

The following example comes from the help pages (type `?sparse.tscgm` in the R console to find it).

Hyperparameters and network sparsity

Possible alternative

GraphicalVAR

Conclusion

References

Abegaz, Fentaw, and Ernst Wit. 2013. "Sparse Time Series Chain Graphical Models for Reconstructing Genetic Networks." *Biostatistics* 14 (3). Oxford University Press: 586–99.

Lauritzen, Steffen L. 1996. *Graphical Models*. Vol. 17. Clarendon Press.