Vector Autoregressions (VARs)

Matthew Cocci

1. Intuition

Let's build up to the intuition of VARs by starting of its relatives, the zero-mean AR(1):

$$y_t = \varphi y_{t-1} + \varepsilon_t$$

The AR(1) tries to predict *this* period's value of some variable, y, given the *last* value of that variable. But we don't need to stop with just last period's lags. We can include as many as we want:

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t$$

This gives us the AR(p) model, with p lags. But again, we don't need to stop there. Suppose that we include the lagged values of *other* variables that might help predict y, like say x:

$$y_t = a_1 y_{t-1} + \cdots + a_p y_{t-p} + b_1 x_{t-1} + \cdots + b_p x_{t-p} + \varepsilon_t \tag{1}$$

And presumably, if x helps predict y, why not use y to predict x? So let's write

$$x_t = c_1 x_{t-1} + \cdots + c_p x_{t-p} + d_1 y_{t-1} + \cdots + d_p y_{t-p} + \eta_t \tag{2}$$

As you can see, we can simplify this a lot. In fact, we might toss Equations 1 and 2 into matrices and vectors to simplify notation. This will be particularly useful when we have lots and lots of variables, parameters, and lags. So we might have

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} c_1 & d_1 & \cdots & c_p & d_p \\ b_1 & a_1 & \cdots & b_p & a_p \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \vdots \\ x_{t-p} \\ y_{t-p} \end{pmatrix} + \begin{pmatrix} \varepsilon_t \\ \eta_t \end{pmatrix} \tag{3}$$

And viola. We have a bona fide *vector autoregression*, or VAR.

We arrived at it through successive generalizations of *very* simple intuitive ideas. So if you know time series, and if you know linear algebra, you're home. You know this stuff already. And had you done this 40 years ago and pushed the technique and consequences to their natural implications, you would've gotten a Nobel Prize.

2. Definition and Notation

Now, we'll generalize to non-zero mean process, and we'll fix our notation, which was a bit sloppy above. So let y_t denote an $n \times 1$ vector of observables we want to predict.

Now define the VAR(p) model as follows:

$$y_t = \phi_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + u_t \qquad u_t \sim \mathrm{N}_n(0, \Sigma) \tag{4}$$

where ϕ_0 is $n \times 1$ and where Σ and ϕ_i $(i = 1, \ldots, p)$ are $n \times n$. So clearly, y_t can be quite a complicated linear function of it's previous lags and it's components.

Next, let's write Equation 4 in more compact, matrix notation. Specifically, concatenate the ϕ_i column vectors horizontally, and stack the lags of y into one big column vector:

$$y_t = \underbrace{\begin{pmatrix} \phi_0 & \phi_1 & \cdots & \phi_p \end{pmatrix}}_{\Phi} \underbrace{\begin{pmatrix} 1 \\ y_{t-1} \\ \vdots \\ y_{t-p} \end{pmatrix}}_{x_t} + u_t \qquad u_t \sim \mathrm{N}_n(0, \Sigma)$$

$$\Leftrightarrow \qquad y_t = \Phi x_t + u_t \tag{5}$$

where Φ is $n \times (np + 1)$ and x_t is $(np + 1) \times 1$.

3. OLS Estimator

Now that we have the model, let's find the OLS for the parameters in Φ. To do so, we'll take one element (or component) of the vector y_t at a time, so one row of y_t and Φ at a time. So let's minimize the sum of squared errors for row/component i:

$$\min_{\Phi^{(i)}} \sum_{i=1}^{T} \left(y_t^{(i)} - \Phi^{(i)} x_t \right)^2 \tag{6}$$

Now this estimator looks just like standard multivariate OLS regression, resulting in the "the normal equation":

$$0 = \frac{d}{d\Phi^{(i)}} \left\{ \sum_{i=1}^{T} \left(y_t^{(i)} - \Phi^{(i)} x_t \right)^2 \right\} = -2 \sum_{i=1}^{T} x_t' \left(y_t^{(i)} - \Phi^{(i)} x_t \right)$$

$$\Rightarrow \quad \hat{\Phi}^{(i)} = \frac{\sum_{t=1}^{T} x_t' y_t^{(i)}}{\sum_{t=1}^{T} x_t' x_t} = \left[(X'X)^{-1} X' Y^{(i)} \right]'$$

where X is $T \times (np + 1)$ and $Y^{(i)}$ is $T \times 1$ where

$$X = \begin{pmatrix} x_1' \\ \vdots \\ x_T' \end{pmatrix} \qquad Y^{(i)} = \begin{pmatrix} y_1^{(i)} \\ \vdots \\ y_T^{(i)} \end{pmatrix} \tag{7}$$

So now we have the estimator for each *row* of Φ, which allows us to write

$$\hat{\Phi} = \begin{pmatrix} \hat{\Phi}^{(1)} \\ \vdots \\ \hat{\Phi}^{(n)} \end{pmatrix} = \left[(X'X)^{-1} X'Y \right]' \tag{8}$$

where X is as above and Y is $T \times n$, which is a matix with y_t' stacked on top of each other for $t = 1, \ldots, T$. So it's the $Y^{(i)}$ defined above placed next to each other in a row.

This notation we've developed also gives us a convenient way to define the *sum of squared OLS residuals* matrix:

$$\hat{S} = (Y - X\hat{\Phi})'(Y - X\hat{\Phi}) \tag{9}$$

Eventually, when we start doing the Bayesian computations, we'll use this in the multivariate analog of a result we recall from univariate mathematical statistics, both of which are written below:

$$\begin{aligned} \sum_{i=1}^{m} (y_i - \mu)^2 &= \sum_{i=1}^{m} (y_i - \hat{\mu})^2 + m(\hat{\mu} - \mu)^2 \\ (Y - X\Phi)'(Y - X\Phi) &= \hat{S} + (\Phi - \hat{\Phi})' X'X (\Phi - \hat{\Phi})' \end{aligned} \tag{10}$$

where anything with " $\hat{}$ " denotes the sample estimate/analog.

4. Likelihood Function

Now that we have an easy representation of the our VAR(p) model, we move to the likelihood function. By our definition, y_t, conditional on x_t (the lags $y_{t-1}, \ldots, y_{t-p}$), happens to be normally distributed:

$$\begin{aligned} & p(y_t \mid x_t, \Phi, \Sigma) \sim \mathrm{N}_n(\Phi x_t,\ \Sigma) \\ \Rightarrow \quad & p(y_t \mid x_t, \Phi, \Sigma) \propto |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} \left(y_t - \Phi x_t\right)' \Sigma^{-1} (y_t - \Phi x_t) \right\} \end{aligned} \tag{11}$$

Now we use the trick from the trace section in the appendix to rewrite the previous line

$$\Rightarrow \quad p(y_t \mid x_t, \Phi, \Sigma) \propto |\Sigma|^{-1/2} \exp\left\{ -\frac{1}{2} \mathrm{tr}\left[\Sigma^{-1} (y_t - \Phi x_t) \left(y_t - \Phi x_t\right)' \right] \right\} \tag{12}$$

5. Joint Density Function

Now, we construct the *joint density function* for the entire series, $Y_{1:T}$, where

$$Y_{t_0:t_1} = (y_{t_0}, \ldots, y_{t_1})$$

Therefore, conditional on a presample $y_{-p+1}, \ldots, y_0$, the jdf is written

$$p(Y_{1:T} \mid Y_{-p+1:0}, \Phi, \Sigma) = \prod_{t=1}^{T} p(y_t \mid Y_{-p+1:t-1}, \Phi, \Sigma) \tag{13}$$

$$= \prod_{t=1}^{T} p(y_t \mid Y_{t-p:t-1}, \Phi, \Sigma) \tag{14}$$

$$\propto \prod_{t=1}^{T} |\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\text{tr}\left[\Sigma^{-1}(y_t - \Phi x_t)\,(y_t - \Phi x_t)'\right]\right\}$$

where we could jump from Equation 13 to 14 since the density of y_t depends only on the previous p lags, not the entire history up until t. And given that the trace of a sum of two matrices is the sum of the traces, we can simplify the jdf

$$p(Y_{1:T} \mid Y_{-p+1:0}, \Phi, \Sigma) \propto |\Sigma|^{-\frac{T}{2}} \exp\left\{-\frac{1}{2}\text{tr}\left[\Sigma^{-1}\sum_{t=1}^{T}(y_t - \Phi x_t)\,(y_t - \Phi x_t)'\right]\right\}$$

$$\propto |\Sigma|^{-\frac{T}{2}} \exp\left\{-\frac{1}{2}\text{tr}\left[\Sigma^{-1}(Y - X\Phi)'(Y - X\Phi)\right]\right\}$$

where X and Y are defined as above in Equation 7 and the subsequent conclusion to the section.

A. Trace

Definition: If A is an $n \times n$ matrix, then

$$\text{tr}[A] = \sum_{i=1}^{n} a_{ii} \tag{15}$$

which is the sum of diagnoal elements.

Trace Fact: If X is $m \times n$ and Y is $n \times m$, then

$$\text{tr}[XY] = \text{tr}[YX] \tag{16}$$

This isn't dificult to prove, just tedious.

Useful Trick: Suppose that a is an $n \times 1$ vector and B is a symmetric positive definite $n \times n$ matrix. Then

$$a'Ba \text{ is a scalar}$$

Then, since the trace of a scalar is just eqal to that scalar, we can rewrite

$$\begin{aligned} a'Ba &= tr\left[a'Ba\right] \\ &= tr\left[a'(Ba)\right] \end{aligned}$$

Now if we use Equation 16, taking $X = a'$ and $Y = Ba$, we can carry on from the simplification above to write

$$\begin{aligned} a'Ba &= tr\left[a'(Ba)\right] = tr\left[(Ba)a'\right] \\ &= tr\left[Baa'\right] \end{aligned}$$

B. Kronecker Product and Vec Operator

B.1. Definitions

Suppose we have two matrices, A which is $m \times n$ and B which is $p \times q$. Then the *Kronecker Product* of A and B is

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

which implies that the new matrix is $(mp) \times (nq)$.

Next, the *vec operator* takes any matrix A that is $m \times n$ and stacks to columns on top of each other (left to right) to form a column vector of length mn. Supposing that a_i are column vectors to simplify notation:

$$\text{if } A = \begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix} \qquad a_i \in \mathbb{R}^{n \times 1}$$

$$\text{then vec } A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$$

B.2. Properties

Property 1 Let A be $m \times n$, B be $p \times q$, C be $n \times r$, and D be $q \times s$. Then

$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{17}$$

Property 2 $(A \otimes B)' = (A' \otimes B')$.

Property 3 $(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$.

Property 4 $\text{tr}[A'BCD'] = \text{vec}(A)'(D \otimes B)\text{vec}(C)$.

How about some proofs?

B.3. Proofs of Property

Of all the properties, this one strikes me the most of "Now what the hell?" There is zero intuition for this that I can see, so let's try to derive it.

$$[A'BCD'] = \text{vec}(A)'(D \otimes B)\text{vec}(C) \tag{18}$$

Proof. We know that the matrix that results from $A'BCD'$ must be square to apply the trace operator to it. So let's call that the beast inside the trace operator E and suppose that it is $q \times q$. Now what does that imply about the size of our other matrices?

Well to allow the matrix multiplication to be carried out (i.e. we can only multiply matrix X and matrix Y if the columns of X equal the rows of Y), and to have E be $q \times q$, this forces

$$A \text{ is } m \times q \qquad B \text{ is } m \times n \qquad C \text{ is } n \times p \qquad D \text{ is } q \times p$$

where m, n, p, and q are all left unspecified.

Okay then, let's get a move on

$$\text{tr}[A'BCD'] = \text{tr}[E] = \sum_{i=1}^{q} e_{ii}$$

Now suppose define $X = A'B$ and $Y = CD'$, implying that X is $q \times n$ and Y is $n \times q$. We can rewrite the diagonal components of E in terms of the rows of X, the $x_{i\cdot}$, and the colums of Y, $y_{\cdot i}$.

$$\text{tr}[E] = \sum_{i=1}^{q} e_{ii} = \sum_{i=1}^{q} x_{i\cdot} y_{\cdot i}$$

Now let's not forget where the rows and columns of X and Y come from. The ith row of X is the product of the ith row of A' (or the ith column of A) with each subsequent column of B. Similarly, each column of Y is the dot product of each successive row of C with the ith column of D' (or the ith row of D). Mathematically,

$$\begin{aligned}
\text{tr}[E] &= \sum_{i=1}^{q} e_{ii} = \sum_{i=1}^{q} x_{i\cdot} y_{\cdot i} \\
&= \sum_{i=1}^{q} a'_{\cdot i} \begin{pmatrix} b_{\cdot 1} & \cdots & b_{\cdot n} \end{pmatrix} \begin{pmatrix} c_{1\cdot} \\ \vdots \\ c_{n\cdot} \end{pmatrix} d'_{i\cdot}
\end{aligned}$$

where $a'_{\cdot i}$ is the ith column of A, transposed, and $d'i\cdot$ is the ith row of D, transposed.

Alright, where did that get us? Well, let's look at that last expression. If we group the last three components (with b's and c's and d's in them) into some abstract term z_i, we notice that we multiply some z_i by $a'_{\cdot i}$, add to the sum, increment i, multiply the next z_i by the next $a'_{\cdot i}$, add to the sum, increment i, and so on. Mathematically:

$$\begin{aligned}\operatorname{tr}[E] &= \sum_{i=1}^{q} a'_{\cdot i} \begin{pmatrix} b_{\cdot 1} & \cdots & b_{\cdot n} \end{pmatrix} \begin{pmatrix} c_{1\cdot} \\ \vdots \\ c_{n\cdot} \end{pmatrix} d'_{i\cdot} \\ &= \sum_{i=1}^{q} a'_{\cdot i} z_i \end{aligned} \tag{19}$$

where z_i is of size $m \times 1$.

But wouldn't it be nice if we could write that as a dot product—a product of scalars. We could use simple matrix notation to and just write

$$\begin{aligned}\operatorname{tr}[E] &= \sum_{i=1}^{q} a'_{\cdot i} z_i \\ &= \begin{pmatrix} a'_{\cdot 1} & \cdots & a'_{\cdot q} \end{pmatrix} \begin{pmatrix} z_1 \\ \vdots \\ z_q \end{pmatrix} = \operatorname{vec}(A)' \begin{pmatrix} z_1 \\ \vdots \\ z_q \end{pmatrix} \end{aligned} \tag{20}$$

So clearly you noticed what that guy on the left is! It's the transpose of the columns of A lined up alongside to each other. In other words, this is $\operatorname{vec}(A)'$!! Whew, so now we at least have *something* from Equation 18. Now let's try to pin down that z_i terms that helped us get here.

Equation 20 prompts us to think a bit differently. Namely, imagine the big column vector of z's in Equation 20. Since each z_i term is, itself, a vector of size $m \times 1$, let's go to a more granular level. Namely, if z^*_{ij} represents the j component of z_i, then we can write

$$z_i = \begin{pmatrix} z^*_{i1} \\ \vdots \\ z^*_{im} \end{pmatrix} = \begin{pmatrix} b_{\cdot 1} & \cdots & b_{\cdot n} \end{pmatrix} \begin{pmatrix} c_{1\cdot} \\ \vdots \\ c_{n\cdot} \end{pmatrix} d'_{i\cdot}$$

□