The ARMA framework

William Mann

EMORY
GOIZUETA BUSINESS SCHOOL

Outline of next two weeks

We will discuss several types of time-series processes:

- AR(p): *Autoregressive* process of order p.
- MA(q): *Moving-average* process of order q.
- ARMA(p,q): Combination of $AR(p)$ and $MA(q)$.

These can be combined to approximate any stationary process.

This will us an alternative way to build forecasts, compared to the exponential-smoothing approach from before, as well as a sound framework for any other analysis we might want to pursue.

AR and MA processes, and the Wold theorem

AR(p) process

Last week, we saw the idea of an AR(1) process:

$$y_t \quad = \quad \mu \quad + \quad \phi y_{t-1} \quad + \quad u_t$$

Every date t, the new value of y_t is affected by today's shock, but also puts some weight ϕ on yesterday's y_{t-1}.

We also saw the more general version, an AR(p) process:

$$y_t \;=\; \mu \;+\; \phi_1 y_{t-1} \;+\; \phi_2 y_{t-2} \;+\; \dots \;+\; \phi_p y_{t-p} \;+\; u_t$$

Recall that we can estimate the ϕ numbers by regression.
AR(p) is one of the building blocks of the ARIMA framework.

MA(q) process

The other major building block of time series models is the moving-average or MA process. An MA(1) looks like this:

$$y_t \quad = \quad \mu \quad + \quad u_t \quad + \quad \theta u_{t-1}$$

Like an AR process, there is persistence built into this process. Differently from AR, the persistence is through the shocks u_t.

In an MA(1) process, each shock affects today, and to a lesser extent tomorrow, then disappears from the process and no longer matters.

By contrast, in an AR process, each shock u_t feeds into y_t, so it has a long-lived and slowly-decaying effect on future y.

Again, there can be more than one lag. A general MA(q) process is:

$$y_t \ = \ \mu \ + \ u_t \ + \ \theta_1 u_{t-1} \ + \ \theta_2 u_{t-2} + \ \ldots \ + \ \theta_q u_{t-q}$$

Wold representation theorem

AR and MA processes are useful building blocks, due to the following:

Theorem: *Any covariance-stationary process can be written as a sum of AR(p) and MA(q) processes, plus a non-random constant.*

That is, regardless of the true nature behind any process,
we can always *model* it accurately with something that looks like:

$$y_t \quad = \quad \mu + \underbrace{\phi_1 y_{t-1} + \cdots + \phi_p y_{t-p}}_{\text{AR(p) terms}} + \underbrace{u_t + \theta_1 u_{t-1} + \cdots + \theta_q u_{t-q}}_{\text{MA(q) terms}}$$

A process that combines AR(p) and MA(q) is called ARMA(p,q).

The theorem above is slightly paraphrased. The true theorem only says every stationary process can be written as an MA. But there is also a "duality" between AR and MA, meaning that each one can be converted into the other. So the choice between them is really just for convenience.

ACF and PACF patterns of AR and MA processes

Autocovariances, autocorrelations, partial autocorrelations

For a stationary process, $Cov(y_t, y_s)$ depends only on $t - s$.

- One way to say this is that $Cov(y_t, y_s) = \gamma_{s-t}$ for some numbers $\gamma_0, \gamma_1, \gamma_2, \ldots$. The γ are called the **autocovariances** of y.
- The **autocorrelations** of y are $\tau_{s-t} = \gamma_{s-t}/\gamma_0$.
  Each τ_s is also the coefficient from regressing y_t on y_{t-s}.
- The **partial autocorrelations** β_{s-t} of y are the coefficients from regressing y_t on y_{t-s} while controlling for the in-between y. There is a formula for them too, but it's more complicted.

Next slide: These numbers help us understand stationary processes.

ACF and PACF patterns in AR and MA processes

One way to understand a stationary process is through its *autocorrelation function* (ACF) or *partial autocorrelation fn* (PACF).

These are plots with lags of time on the horizontal axis, and autocorrelations or partial autocorrelations on the vertical.

These figures help us judge the AR and MA patterns in a process, because we know some theoretical benchmarks. Examples:

- **AR(1)**: $ACF(h) = \phi^h$: a *decaying* pattern;
  $PACF(h) = \phi$ if $h = 1$, 0 otherwise: a *cutoff* pattern.
- **MA(1)**: $ACF(h) = \theta$ if $h = 1$, 0 otherwise: a cutoff pattern;
  $PACF(h) = \frac{\theta^h(-1)^{h+1}}{1+\theta^2+\cdots+\theta^{2h}}$: an *alternating* pattern.

See examples in Python notebook posted online.

Estimating ARMA processes

Estimating an AR(p) process

Suppose you have a series of data from a stationary process, and you think the process is described well by an AR(p):

$$y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + u_t$$

How would you go about estimating the numbers ϕ_1, ϕ_2, ... ϕ_p?

We've seen the answer before: Just run a regression.
See examples on Canvas.

Estimating an MA(q) process

Suppose you have a series of data from a stationary process, and you think the process is described well by an MA(q):

$$y_t \ = \ \mu \ + \ u_t \ + \ \theta_1 u_{t-1} \ + \ \theta_2 u_{t-2} + \ \ldots \ + \ \theta_q u_{t-q}$$

How would you go about estimating the numbers θ_1, θ_2, ... θ_p?

Unfortunately, you **cannot** run any simple regression for this. You can't actually observe the "true" shocks u_t that appear above, and you can't estimate them until you have an estimate of θ.

We use a different estimation method called **maximum likelihood**.

Maximum likelihood estimation

Maximum likelihood is an important approach in many settings.
(We will see it again with volatility models in a few weeks.)

The basic idea is this:

- Make an initial guess of all the model parameters: μ, σ^2, and any AR and MA terms $\phi_1, \phi_2, ..., \theta_1, \theta_2, ...$.
- Calculate the **likelihood** (probability) that you would observe the data you did, if the parameters were equal to your guess.
- Try different guesses until that likelihood is as high as possible.

The parameter values that generate the highest likelihood of observing the true data are the **maximum likelihood estimate**.

The likelihood function

- To implement MLE, you need to know the probability described in the prior slide, which is called the **likelihood function**.
- For any standard ARMA process, the function is well-known. You just have a computer find parameters to maximize it. More often, you actually maximize the *log* of the likelihood.
- Example: Here's the log likelihood function for an AR(1).

$$\mathscr{L} = -\frac{T}{2}\ln(\sigma^2) - \frac{T}{2}\ln(2\phi) - \frac{1}{2}\ln\left(\frac{\sigma^2}{1-\phi^2}\right) - \frac{1-\phi^2}{2\sigma^2}\hat{u}_1^2 - \frac{1}{2\sigma^2}\sum_{t=2}^{T}\hat{u}_t^2$$

After estimation: Measures of model quality

Information criteria

An *information criterion* compares models with each other:

- First estimate the model, and measure its *fit* in the data.
  - For a linear regression, fit is measured by R^2.
  - When applying MLE, we use the likelihood function, evaluated at our best parameter estimates. Higher value = better fit.
  - Most software reports the *log* of the likelihood function, which is typically negative, so a *less-negative* number is a better fit.
- Build in some penalty to avoid *overfitting*.
  - The penalty should increase with the number of model terms.
  - In linear regression, adjusted R^2 is an example of this.

The two standard information criteria are AIC and BIC.
See next two slides and examples in Python notebook.

Akaike information criterion (AIC)

The AIC measure of a model is given by

$$AIC \quad = \quad 2 \times k \quad - \quad 2 \times \mathscr{L}$$

- k is the number of parameters that were estimated.
  - For an ARMA model, this is the number of AR and MA terms, plus 1 for σ^2, plus 1 if there was a constant term μ.
  - Example: ARMA(1,1) with a constant term would have $k = 4$.
- $\mathscr{L}$ is the maximum *log* likelihood that we can achieve.
  - This is part of the output from your MLE process.
- You want to find the **smallest** possible AIC.
  - Intuition: Higher $\mathscr{L}$ is good. Higher k is bad.

Bayesian information criterion (BIC, SIC, SBC, or SBIC)

The BIC measure of a model is given by

$$BIC \quad = \quad \ln(N) \times k \quad - \quad 2 \times \mathscr{L}$$

- N is the number of observations in the dataset, everything else is as in the prior slide.
- Intuition: Just a different way of penalizing k.
- There are deep statistical arguments for why the AIC and BIC look the way they do, but this is beyond the scope of class.
- Both are popular and should yield similar conclusions.

How to use information criteria

It's common to search over models to find the smallest AIC or BIC.

- Combine common sense and judgment with this process.
    - Many times, two different models will yield very similar AIC and BIC numbers. It may be that one model scores marginally better, but the other is more economically reasonable.
- AIC and BIC only give *relative* comparisons.
  There is no sense of a "good" or "bad" value to judge by.
    - For *overall* assessment of a model, you should instead check whether there are any remaining features of the data that seem important to capture. See next section.

After estimation: Check for remaining time-series patterns

Checking for time-series patterns

We are done with model selection, once we are satisfied that we have captured all the important time-series features of the data.

To judge this, we examine the *residuals* from the fitted models.

- Visually, look at ACF and PACF graphs of the residuals.
- Formally, we can apply statistical tests.
  The most common choice is the Ljung-Box test.

Ljung-Box test

The Ljung-Box test statistic is defined as

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k}$$

where $\hat{\rho}_k$ is the sample autocorrelation up to lag k, and h is some specified number of lags we want to include.

The null hypothesis is that the data are independently distributed. Under this hypothesis, Q follows a standard χ^2 distribution.

Hence, compare Q with critical values of this distribution.

The test will tell us if there is *some* remaining time series feature left in the data (up to lag h), but will not tell us exactly what it is.

Common practice

- The Ljung-Box test statistic and p-values are a standard part of the output from ARMA estimation (usually with $h = 1$).
- A common approach is to try different models until L-B test fails to reject the null that the residuals have no persistence.
- Again, keep in mind the role of common sense and judgment. If you already have a sensible model, it may not be worth adding extra complications just to lower the L-B statistic marginally.
- **Critique of common practice:** When there are AR terms in the model, it is actually *incorrect* to apply the L-B test to the residuals of the model. The correct test is a modified version called the *Breusch-Godfrey* test, which leads to lower p-values. But you will rarely (if ever) see this in practice.

ARMA model selection: The Box-Jenkins approach

The Box-Jenkins approach

Combining all our pieces so far, we can map out an ARMA-based framework for modeling and forecasting a stationary data series:

- Examine the series, especially its ACF and PACF graphs, to determine a few choices of p and q that we should consider.
- For each pair of (p, q), estimate the parameter values by MLE.
- Compare each set of results based on AIC or BIC. Check the ACF and PACF of the residuals, and the L-B test, for any remaining time-series patterns that should be modeled.
- Choose a model that scores well and is economically reasonable.

If the data are not stationary, you first transform as necessary so that they are, and then follow this procedure. See discussion later.

After model selection and estimation:
Building forecasts and prediction intervals

One-step forecast and prediction interval for MA(1)

Let's first think about a simple MA(1) process $y_t = \mu + u_t + \theta u_{t-1}$.
Suppose the data ends at T. Let's forecast $y_{T+1} = \mu + u_{T+1} + \theta u_T$.

- We first fit the model to get estimates of μ, θ, and also all of the u values (using the sample residuals from the model).
    - All of these are just estimates, meaning there is still uncertainty in them – but we will ignore that and act as though they are the true values. This is not quite ideal, but it is standard.
- As of time T, we forecast y_{T+1} with its conditional expectation, $\mathbb{E}_T[y_{T+1}] = \mu + \theta u_T$ (using the fact that $\mathbb{E}[u_{T+1}] = 0$).
- $Var_T(y_{T+1}) = \sigma^2$, since u_{T+1} is the only thing unknown as of t. A 95% prediction interval is our forecast $\pm 2 \times \sqrt{\sigma^2}$.

Two-step forecast and prediction interval for MA(1)

Now let's forecast $y_{T+2} = \mu + u_{T+2} + \theta u_{T+1}$ as of time T.

The forecast is $\mathbb{E}_T[y_{T+2}] = \mu$ since we don't know u_{T+2} or u_{T+1}.

Also because we don't know either of these errors, the prediction interval is wider: $Var_T(y_{T+2}) = Var(u_{T+2} + \theta u_{T+1}) = \sigma^2 \times (1 + \theta^2)$.

A 95% confidence interval is our forecast $\pm\sqrt{\sigma^2 \times (1 + \theta^2)}$.

In fact this expected value and variance are the long-run values, so they will apply to all dates $T + 2$ onward. This reflects the fact that shocks only affect an MA(1) process for two dates before dying out.

Forecasting an AR(1) process

Now consider an AR(1) process, $y_t = \mu + \phi y_{t-1} + u_t$.

At time T we forecast $y_{T+1} = \mu + \phi y_T + u_t$ as $\mathbb{E}_T[y_{T+1}] = \mu + \phi y_T$.

Farther out, $\mathbb{E}_T[y_{T+2}] = \mathbb{E}_T[\mu + \phi y_{T+1} + u_T] = \mu + \phi \mathbb{E}_T[y_{T+1}]$.
And we just calculated $\mathbb{E}_T[y_{T+1}] = \mu + \phi y_T$.
We can just substitute that in, which gives us

$$\mathbb{E}_T[y_{T+2}] = \mu(1+\phi) + \phi^2 y_T$$

We can repeat this process to forecast as far out as we want.

Not surprisingly, the forecasts for an AR(1) will start close to y_T, and decay slowly to the long-run average of $\frac{\mu}{1-\phi}$.

Prediction interval for an AR(1) process

Recall that any ARMA process can be written as a pure MA.
The AR(1) process can be written as an MA with ϕ^t at each lag.

$$\begin{aligned} y_t &= \mu + \phi y_{t-1} + u_t \\ &= \mu + \phi^2 y_{t-2} + \phi u_{t-1} + u_t \\ &= \mu + \phi^3 y_{t-3} + \phi^2 u_{t-2} + \phi u_{t-1} + u_t \quad = \quad \ldots \end{aligned}$$

We can get the variance of the forecast for any future date, keeping in mind that only values of u from time T onward are unknown:

- $Var_T(y_{T+1}) = Var_T(u_T + \phi u_{T-1} + \phi^2 u_{T-2} \ldots) = \sigma^2$
- $Var_T(y_{T+2}) = Var_T(u_{T+1} + \phi u_T + \phi^2 u_{T-1} \ldots) = \sigma^2(1 + \phi^2)$
- $Var_T(y_{T+2}) = Var_T(u_{T+2} + \phi u_{T+1} + \phi^2 u_T \ldots) = \sigma^2(1 + \phi^2 + \phi^4)$

...and so forth. The variance grows towards a limit of $\frac{\sigma^2}{1-\phi^2}$.

Forecasts and prediction intervals in general

The procedure is the same for any ARMA model:

- Build forecasts by substituting in values of y and u from the end of the dataset into the AR and MA terms of the model.
    - After reaching the last y, substitute forecasts of y.
    - After reaching the last u, substitute zeros.
- For a prediction interval at horizon k:
    - Convert to MA form, square the first k MA terms, add them up plus 1, and multiply by σ^2. This is the forecast variance.
    - Take the square root of the variance to get the standard error. A 95% CI is the forecast $\pm$ two SEs.
- Of course this should really be done with software, not by hand.

Practical notes

- All the above formulas for prediction intervals assume that the errors in the model are independent. They are not quite correct if there are remaining time-series patterns in the residuals.
    - This is one reason to check for such patterns.
    - However, the *forecasts* themselves are not affected by this issue, as they only assume the shocks are zero on average.
- *All* the formulas ignore the fact that the model parameters themselves are estimated with uncertainty.
    - This is a limitation, but it is standard.
    - One result is that prediction intervals are always too narrow.

Handling trends and seasonality in the ARMA framework

Comparing ARMA with exponential smoothing methods

The ARMA framework assumes you have a stationary process.

- This means that ARMA is best seen as an alternative to *simple* exponential smoothing. It will fail to capture stochastic trends, or seasonality, for the same reason that SES failed to do so.

Recall how we extended SES to address these issues:

- To handle trends, we used "double" exponential smoothing.
- To handle trends *and* seasonality, we introduced "triple" exponential smoothing (also called the Holt-Winters method).

How do we do the same thing in the ARMA framework?

- To handle trends, we extend to "ARIMA" models. Basically, modeling *differenced* data by ARMA.
- To handle trends *and* seasonality, we extend to "SARIMA," which features "seasonal" AR, MA, and differencing terms.

Trending data: ARIMA models

A strong trend is a source of nonstationary behavior.

In the ARMA framework, trends are typically modeled as a unit root:

- Difference the data, and check that this difference is stationary.
- Then analyze the differenced data with the ARMA procedure.

A process whose difference is ARMA(p, q) is called ARIMA(p, d, q).
ARIMA stands for **A**uto**R**egressive **I**ntegrated **M**oving **A**verage.

- The p and q parameters have the same meaning as before.
- We add d in between them to denote the number of times that the data must be differenced in order for it to be stationary.
- In almost every real-world scenario, $d = 1$.

Seasonal data: SARMA and SARIMA models

- Seasonality leads to regular spikes in the ACF and PACF. These cannot be modeled with standard AR or MA processes.
- Instead, we add another dimension to the model: A seasonal ARIMA model (SARIMA) has the usual ARIMA terms, plus *seasonal* versions of these terms labeled $(P, D, Q)_s$.
- Altogether this is written as SARIMA$(p, d, q) \times (P, D, Q)_s$.
- SARIMA is the most general category of ARMA model, and produces forecasts comparable to Holt-Winters.
- See examples on Canvas of SARIMA estimation and forecasting.