DS-GA 3001.001 Special Topics in Data Science: Modeling Time Series
Handout: Summary ARIMA models

1 Preface

The general philosophy of this class of models is that we believe that data comes from a (usually zero/constant mean) joint gaussian distribution; in this case the goal of statistical modeling is to provide a parametric description for the *autocovariance function*, which since we only model stationary processes should only depend on the lag. We achieve this mathematical objective by taking white noise and applying some linear transformation to 'color' it. All models in this class behave this way; they differ in the exact form of the linear transform and the number of parameters involved. The goal of the ARIMA tool set is to give us a mathematical understanding of what kind of covariance structure can individual models capture (model/parameters->ACF). Once we've worked that out, there are general purpose solutions to make predictions (since all of the elements of the sequence are jointly gaussian this will be a linear predictor, with weights specified by the ACF, see slides for expression) . There are also generic ways of learning the parameters from observations (maximum likelihood).

There's admittedly not a lot of intuition to be had about this class of models, it's mostly down to plain math. In general, there's no clear parallel between such the generative process and how we might think real data is actually generated (though AR models arguably make some intuitive sense). Still, this class of models remains popular in practice, at least in some circles, and we need to figure out how far such simple model gets us before we can think about more interpretable/structured generative models.

2 Model specification

The starting point for all ARIMA models is the simplest possible model of a time series, *white noise*. This model assumes a factorized normal distribution for individual elements w_i:[1]

$$\mathrm{P}(\mathbf{w}) = \prod_i \mathrm{P}(w_i) \tag{1}$$

with zero mean, and variance σ^2, $w_i \sim \mathcal{N}(0, \sigma^2)$.

The most obvious way of linearly filtering white noise is using a *moving average* MA(q), defined as:

$$x_i = w_i + \sum_{j=1}^{q} \theta_j w_{i-j} \tag{2}$$

with parameters θ_j. This only captures short term dependencies up to lag q.

A less obvious, but more intuitive alternative, is to use autoregressive models, based on the idea that the current value x_i can be *linearly* [2] explained by the recent history $x_{i-1}, x_{i-2}, \ldots x_{i-p}$. Formally, an autoregressive process of order p AR(p) is defined as:

$$x_i = w_i + \sum_{j=1}^{p} \phi_j x_{i-j} \tag{3}$$

with parameters ϕ_j. Expanding this recursion, it is easy to see that x_i is also a linear combination of white noise, with the important distinction that the number of terms is no longer fixed, but increases

[1] Since white noise has no dependencies over time, it seems a rather useless model of time series it itself; still it makes for a mathematically tractable starting point for more complex models. It also denotes the point where there is no additional structure to be explained in residuals of model fits.

[2] The AR(p) graphical model makes fora natural way of describing many real-world data generating processes, but the linearity assumption might be too restrictive in some applications.

with i (see the AR(1) example in the lecture, example 3.1 from tsa4 for general case). In principle the ACF of such processes can span a broad time lag range with fewer parameters.

AR and MA can be further combined resulting in the general class ARMA(p, q):

$$x_i - \sum_{j=1}^{p} \phi_j x_{i-j} = \sum_{l=0}^{q} \theta_l w_{i-l} \tag{4}$$

where we have moved the AR component on the left side (the separation is done by random variable type, x:left, w:right, since we'll want to use the polynomial form later) and defined $\theta_0 = 1$ for shorthand. These model class can in principle capture the ACF of any stationary process, given enough degrees of freedom.

All the above models are special instances of the umbrella linear model, defined as

$$x_i = \sum_{j=-\infty}^{\infty} \psi_j w_{i-j} \tag{5}$$

with parameters ψ_j constrained such that the series is finite, $\sum_j |\psi_j| < \infty$. See Example 3.4 in book or slides for converting an AR model into this representation.

Lastly, all the above models implicitly have zero mean (linear combination of white noise). Integrated models, ARIMA(p, d, q) further assume that observations have a time-dependent non-zero mean; this trend can be removed after a reparametrization where we no longer model the original data, but the stationary component, obtained after taking d finite differences. This is to say that the d-th order difference of the data, $\nabla^k x_i$, is an ARMA(p, q) process, where the difference operator is defined as $\nabla x_i = x_i - x_{i-1}$. This operator can be used to remove nonlinear trends (though one would need to apply it repeatedly).

3 Auto-covariance

If we remember the general formula for the covariance of linear combinations:

$$\text{cov}\left(\sum_i a_i X_i; \sum_i b_j Y_j\right) = \sum_{ij} a_i b_j \text{cov}(X_i, Y_j) \tag{6}$$

and we use the properties of white noise, we can derive closed form expressions for the auto-covariance of any linear process as:

$$R_h = \sigma^2 \sum_k \psi_k \psi_{k+h}. \tag{7}$$

[all of the terms that don't have exactly lag h will go away because the noise is independent, leaving us to only accumulate all the pairs at lag h; for any of the simple models we have looked at most of ψ_ks will be zero anyways, leaving us with very simple expressions.]

4 Backward operator and polynomial representation.

The class of ARIMA models can be understood in a general way using its polynomial representation. The first step to achieve this is to define the *backward* operator B:

$$Bx_i = x_{i-1} \tag{8}$$

Some useful consequences of this are:

$$B^k x_i = x_{i-k} \tag{9}$$

$$B^{-1} x_i = x_{i+1} \tag{10}$$

$$\nabla^d x_i = (1 - B)^d \tag{11}$$

This abstract operator allows us to rewrite ARMA processed in terms of polynomials $P(\cdot)$ and $Q(\cdot)$, corresponding to the AR, and MA component respectively:

$$P(B)x_i = Q(B)w_i, \text{ where} \tag{12}$$

$$P(B) = 1 - \sum_{j=1}^{p} \phi_j B^j \tag{13}$$

$$Q(B) = 1 + \sum_{k=1}^{p} \theta_k B^k \tag{14}$$

The roots of polynomials P and Q completely specify the properties of the underlying process. They determine whether a process is *causal* or *invertible* (see def. on tsa4 book, pg85). [3] Common roots allows us easily detect parameter redundancy (common roots should be first cancelled out!). The roots are also needed to work out the functional form of the ACF for any arbitrary ARIMA model, when identifying the standard linear form and using the R_h formula (see above) is hard. The details of how exactly we go from roots to the ACF is provided in the book/slides, but the general idea is that for AR(p) each root will contribute one term in the ACF, with an exponential decay, with multi-roots get treated slightly differently. The core mathematical machinery that makes this general solution possible is *difference equations* (and the trick why it works is that we can write recursion equations for $\gamma(h)$ in the form of such an equation). See book/slides for concrete examples.

For the mathematically inclined: there is no actual proof for the general case, instead the book uses some simple examples to draw parallels between the original and the polynomial representation, how the roots show up in the form of the ACF etc. Admittedly this will prove deeply unsatisfying for some, but it is also the standard way these models are taught. The goal of this one lecture here is to give you some rough notion of how these kind of models work, before we get into more structured probabilistic models which are the main topic of the class.

[3] We know why we care about causality: because we try to predict the future from past observations and because we think this is how the world works, but why might we care about a model being invertible? It is mostly for mathematical reasons: if causality means that we can move the AR effects on the right side (as we did for AR(1) explicitly using recursion, or implicitly using the inversion of polynomial P(B)), invertibility means that we can move the MA part to the left side, by inverting Q(B). N.B. doing so will usually result to going from a finite sum of terms to a series.