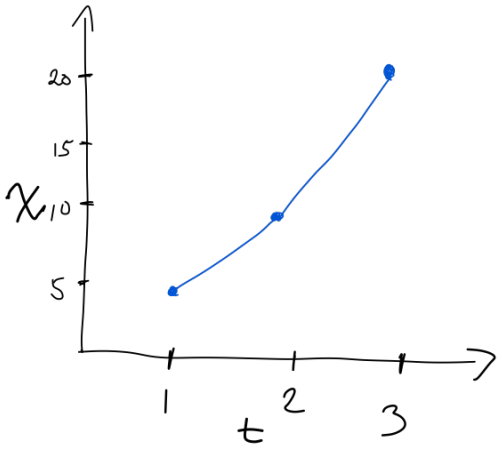
In our lab session today, we went over simple cases of Autoregressive models, simplex projection (Sugihara 1990) and S-Map (Sugihara 1994).

Autoregressive models assume that the value at a point t in time is determined by the value(s) before it (t-1, t-2, etc).

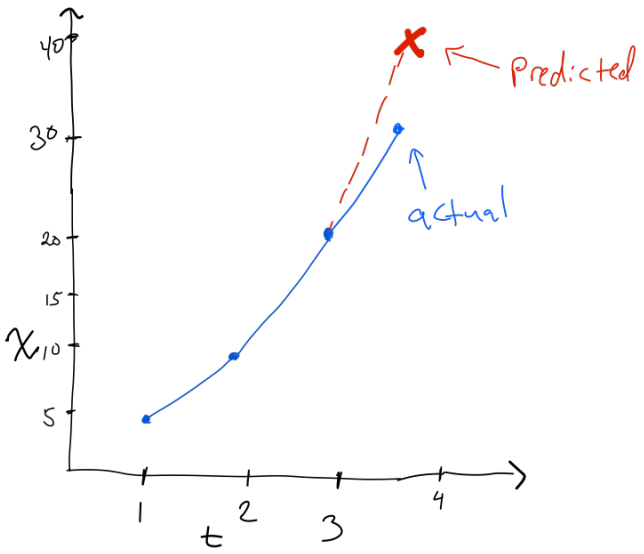
For example, if we had the following time Series:



We can see that the value at t1 is 5, t2 it's 10, and t3 it's 20. From this amount of data, we might assume that this system can be explained in the following form:

$$X_{t+1} = \alpha X_t \quad , \quad \alpha = 2$$

Here, alpha represents our coefficient (or AR parameter) that describes our dynamics; our dynamics being simply double the last value. Given this, what might you predict the value at t4 to be? That's right! Our prediction based on our data thus far would lead us to guess the next point is 40. But what if we get this:



Here we see the actual value at t4 is 30, not 40. So our thought that every value is double the previous value isn't actually correct. This essentially give us 3 equations:

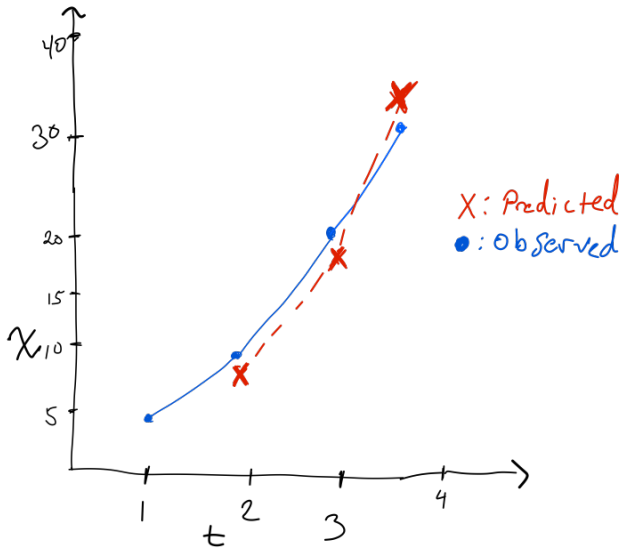
$$x_{t+1} = \alpha x_t$$

1) $10 = \alpha 5$

2) $20 = \alpha 10$

3) $30 = \alpha 20$

So alpha =2 works for the first two equations, but not really for the third. We can think of our error for alpha=2 to be 10, because our sum of errors is 10 (because our first two predictions have no error, but our third prediction is off by 10). What if instead we said alpha =1.8? We would get 9 for equation 1, 18 for equation 2, and 36 for equation 3, shown below. Can you solve for the total error now? That's right! It's 9. So alpha=1.8 did better than alpha=2. But is 1.8 the best solution? There are ways to analytically solve for the best solution, but we use a tool called Singular Value Decomposition (SVD). Don't worry too much about how it works, just know that it finds the best set of parameters when given a set of equations.



But what if we wanted to make our function a bit more complicated? What if we assumed every given point at time t was influenced by the point before it, and the point before that one, like this:

$$x_t = \alpha x_{t-1} + \beta x_{t-2}$$

Now we have two parameters: alpha and beta. But what are alpha and beta for our given time series? Well we can still use SVD to get the best fit solution. But if given only 4 points (like we currently have), we can only make 2 equations in that form. Can you write them down? With 2 equations and 2 variables, you can actually exactly solve for solutions (if they exist), and don't necessarily need a "best fit". Ideally time series are much longer so we have many more equations than variables, and our SVD solution to those variables is a better representation of true overall dynamics.

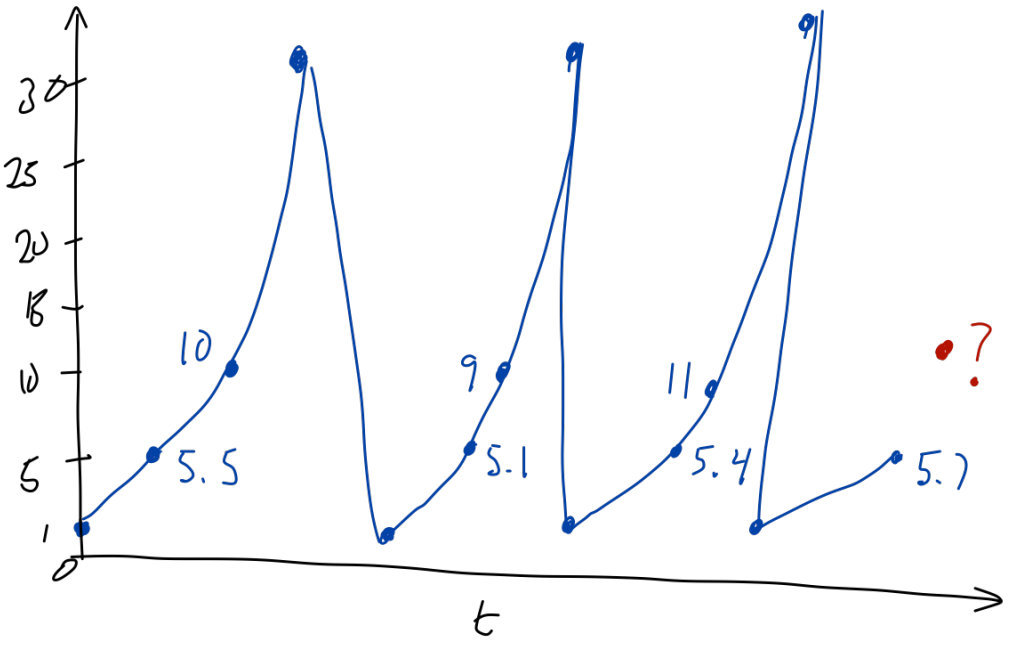
Non-linear models

OK that's all for now on Autoregressive models. Let's talk about some non-linear models. First, what's the difference between a linear model (like AR) and a non-linear one? Well notice in the AR model above, we only found one alpha to satisfy the whole system. We found alpha=1.8 to be our best fit for all of them. But what if alpha could change? What if alpha was 2 when xt was equal to 5 and 10, but equal to 1.5 when xt was 20? It would have perfect predictions! The fact that we allow our alpha to change over time now makes it non-linear.

In other words, our relationships between variables are not constant; they can change over time. Another word for this is state-dependent: given a certain "state" (or values) of variables (xt, xt-1...), the dynamics (alpha, beta...) behave in a certain way.

How can we incorporate this into our model? There are two main methods we use: simplex and S-map. Let's start with simplex.

Simplex looks for points in time in our time series with a similar state as the current state. It has nothing to do with their proximity in time; the closest state could be anywhere on the time series. Take, for example:



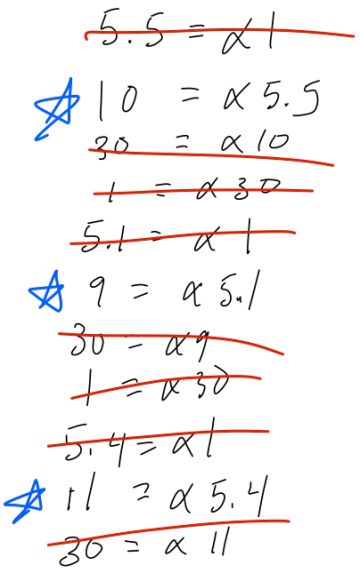
Here, we are trying to predict what the red point will be. We can kind of guess it will be around 10, but what is our guess exactly? Well we look for the most similar states to our current state. In 1 dimension, our state is simply 5.7. In higher dimensions (embedding with more lags) our state will also include previous points, so in 2D, our state would be the 2 dimensional point [5.7, 0]. But let's stick with 1D. Our most similar states (or nearest neighbors) are 5.4 and 5.5. Notice how are nearest neighbor (5.5) is actually the furthest in time. Now our prediction is simply a weighted average of what these neighbors did, so about 10.5, but a little lower (like 10.45) because it is weighted more closely to the nearest neighbor (5.5 -> 10).

Note I chose 2 neighbors. We tend to choose E+1 neighbors in our average, where E is our embedding dimension (1 in this case).

Now S-map is a bit more like our AR model. Let's write some equations for this 1D example, just like above:

$$
\begin{aligned}
5.5 &= \alpha\, 1 \\
10 &= \alpha\, 5.5 \\
30 &= \alpha\, 10 \\
1 &= \alpha\, 30 \\
5.1 &= \alpha\, 1 \\
9 &= \alpha\, 5.1 \\
30 &= \alpha\, 9 \\
1 &= \alpha\, 30 \\
5.4 &= \alpha\, 1 \\
11 &= \alpha\, 5.4 \\
30 &= \alpha\, 11
\end{aligned}
$$

We see that sometimes alpha is about 5 in the first equation, then 2 in the next, then 3, then goes way down to 1/30. Thus, our AR approximation using SVD will be some average of these values, something like 2 ish. We can see that most of the time, this is a crappy prediction for this system! But what if we only used similar states for our fit. So if our current value is 5.7 and we want to predict what will come next, instead of doing SVD on all these equations, we only do it on similar ones, like this:



Now we run SVD on the states that are similar to 5.7, and run SVD on those only. We get an alpha of about 2, which is what we wanted! So now we can say when our value is about 5, our alpha should be about 2. Each unique state gets its own calculated alpha (and beta, gamma, etc. depending on how many dimensions you choose to embed in).

There is a parameter, theta, that determines how you choose which equations are “similar enough”. We will go over this theta parameter next session.