Notes on Trends, Smoothing, and Detrending

36-740/620

27 August 2026 (accompanying lecture 2)

Contents

1 Setting the Scene: Random Processes

Our data will generally be a realization of a **random process** or **stochastic process**, i.e., a collection of random variables, with one random variable for each point in time and space. The random variable we observe at spatial coordinate r and time t will be, generically, $X(r,t)$. If we just care about spatial data, we'll just have $X(r)$, and purely temporal data (like the Kyoto data) will be $X(t)$. In general, the random variables at different points will all be statistically dependent on each other, meaning that

$$\mathbb{P}\left(X(r,t) = x_1 \wedge X(q,s) = x_2\right) \neq \mathbb{P}\left(X(r,t) = x_1\right)\mathbb{P}\left(X(q,s) = x_2\right) \ .$$

Mathematically, it's sometimes convenient to think of the whole random process as a *random function*, with the domain of the function being coordinates (r,t) and the range being possible values of $X(r,t)$. The "dependence" means that when we randomly pick the function, we have to pick the *whole* function, we don't get to make an independent random choice of the value of $X(r,t)$ for each separate r,t.

Because the $X(r,t)$ are random variables, it makes sense to think about their central tendency, and the expected value or mean is usually the most convenient measure of central tendency:

$$\mu(r,t) = \mathbb{E}\left[X(r,t)\right]$$

This is what mean by the **trend** of the process. Notice that this is a *deterministic* function, though usually an unknown one we'd like to learn.

Mathematically, we can always decompose[1] the deterministic random variables into the trend plus random or stochastic **fluctuations**, which always have expectation zero:

$$X(r,t) = \mu(r,t) + \epsilon(r,t) \tag{1}$$
$$\mathbb{E}\left[\epsilon(r,t)\right] = 0 \tag{2}$$

We can't assert anything more about ϵ without further assumptions. In particular, we can't make the usual assumptions that are made in regression courses, that ϵ is uncorrelated, that it has constant variance, that it has the same distribution at each point, that it's Gaussian, etc. But we can, without any loss of generality, say $\mathbb{E}\left[\epsilon(r,t)\right] = 0$ everywhere.

[1] People sometimes call this a "signal plus noise" representation of X. But we can use this even when it's very hard to think of the trend μ as a "signal" (from who?).

2 Learning the Trend

We would like to learn the trend function μ.

2.1 Many realizations

If we have *many* independent realizations of the process, we could just average them. Say we can re-run some experiment many times, getting independent realizations $X^{(1)}, X^{(2)}, \ldots X^{(m)}$. Then

$$\frac{1}{m}\sum_{i=1}^{m} X^{(i)}(r,t) \to \mu(r,t)$$

by the ordinary law of large numbers. As my mention of "experiment" suggests, this idea *can* be applied in the experimental sciences. In neuroscience, for instance, we might repeatedly measure what happens to a particular neuron (or group of neurons, or brain region) after we apply some stimulus to an experimental animal[2]. This is, however, not available in non-experimental contexts.

2.2 Single realization

More typically, we have only *one* realization of the X process — only one history of cherry-blossoms in Kyoto, only one map of shale deposits under Pennsylvania, etc. We will only be able to learn about μ under assumptions, usually assumptions about μ itself[3].

2.2.1 Trends from theory

In some situations, we might have a well-confirmed scientific theory which tells us *exactly* what the trend should be, say that $\mu(r,t) = f(r,t)$ for some completely-specified function f. There are very few situations where this applies — *perhaps* in parts of astronomy and physics.

More common, even in physics and astronomy, is the situation where theory pins down a specific functional *form* for the trend, but there are some unknown coefficients in the formula. That is, the theory says that $\mu(r,t)$ has to be of the form $f(r,t;\theta)$, where f is some mathematical expression where everything is known except for one or more parameters θ. One would then need to estimate θ from the data X to obtain an estimate of the trend. (I'll say a little bit about *how* to do this estimate below, and we'll come back to it later in the course.) The dependence of the observations on each other makes the statistical inference more complicated, which we'll come back to later in the course, but in principle this is just about estimating parameters.

In order for fitting a theoretical model to the data to be a good idea, you need to have reasons to think your theory *does* give you a good idea of the trend. There are domains where this is true — physics, chemistry,

[2] I once helped analyze an experiment where a mechanical vibrator repeatedly gave very precise tweaks to a rat's whiskers. (Rats devote a *lot* of their brain to processing sensations from their whiskers, which help them navigate in the dark.) The electrical responses in the rat's brain were recorded by electrodes inserted through the skull. The machine could repeat the exact same stimulus many times, and on each run the experiment recorded the pattern of electrical activity over time at specific points in the brain. This was similar from one repetition of the experiment to the next (that was the signal or trend) but not identical (that was the noise or fluctuations). Ultimately the goal of the project was to build a de-coder, which could read brain activity and say infer what stimulus was being presented, basically by comparing it to different trends obtained from different stimuli. The ultimate goal of *that* was prosthetic brain-computer interfaces for people with damage to nerves, sensory organs, etc., but as you can imagine there are a lot of steps in between.

[3] The difficulty is that pretty much any data is *logically* compatible with pretty much any trend. For instance, one possibility is that $\epsilon(r,t) = 0$ everywhere, so $X(r,t) = \mu(r,t)$ and what we saw in the data is in fact the only possibility, and if we could re-do the experiment we'd see the same thing over and over. This conflicts with our intuition that a lot of what we see in our data is more or less accidental and even just sheer error, but it *could*, logically, be possible. Or at the other extreme it's always possible that $\mu(r,t)$ is constant in r and t, and all the variation we see in the data is entirely due to the fluctuations ϵ. Now, when we learn or make inferences, we always *rule out* some possibilities, but here we can't rule out possible trend curves using the data alone. So the ruling-out must come from some other source, which is the assumptions we make. If, for instance, $X(t)$ looks pretty much like an exponential growth curve, and I nonetheless want to insist that $\mu(t) =$ constant, I am committing myself to an $\epsilon(t)$ with some very odd-sounding properties (like very strong correlations over very long time intervals). If you are unwilling to believe in noise like that — if you assume that it can't be right — then you can rule out my hypothesis of a constant μ. (How might we check this, without circularity?)

some parts of biology (especially the more "population"-y fields: genetics, evolution, ecology, epidemiology) — where we have very well-supported theories which are detailed enough to pin down specific functional forms. (Some economists think this is true in economics as well, but I think they're fooling themselves.) Beyond that, though, scientific theories are usually much looser – they might tell us that one variable tends to go up with another, but just don't have the detail to give a functional form. This might not be a permanent condition — it used to be true even of physics! — but for now, in lots of scientific domains, let alone in business or policy, we just don't have credible theories which tell us about trends.

2.2.1.1 Curve-fitting using ordinary least squares

I said above that if we've got a theory which says that the trend function $\mu(r,t)$ should have a specific form, $f(r,t;\theta)$, we can estimate the unknown parameters θ by fitting the model to the data. This is usually called "curve fitting", though when we're dealing with spatial data it'd more properly be "surface fitting", etc. The oldest[4] and in many ways most natural way to do this is ordinary least squares (OLS). You're already familiar with OLS from linear regression, but here's how it goes for general curve-fitting.

We have n data points, with coordinates $(r_1,t_1),(r_2,t_2),\ldots(r_n,t_n)$, and at the i^{th} data point we observed $X(r_i,t_i)$. If we make a particular guess about the parameters, say θ, we predict $f(r_i,t_i;\theta)$ for what we'd observe at that data point. This will give us a certain mean squared error (MSE) over the data points:

$$MSE(\theta) = \frac{1}{n}\sum_{i=1}^{n}(X(r_i,t_i) - f(r_i,t_i;\theta))^2$$

The least-squares estimate (LSE) is the value of the parameters which minimizes the MSE:

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{n}\sum_{i=1}^{n}(X(r_i,t_i) - f(r_i,t_i;\theta))^2$$

Taking the derivative and setting it equal to zero gives

$$\sum_{i=1}^{n}(X(r_i,t_i) - f(r_i,t_i;\theta))\nabla_\theta f(r_i,t_i;\theta) = 0$$

which is called the **estimating equation** (or **normal equation**) for the problem. If $f(r,t;\theta)$ is linear in θ this is a linear system of equations which we can solve the same way you did in your linear models course. If $f(r,t;\theta)$ is non-linear in θ, we need to use numerical methods for solving non-linear equations (like Newton's method; see the optional appendix below).

Now, *why* should we do this? Well, suppose the model is right, in the sense that $\mu(r,t) = f(r,t;\theta_0)$ for some

[4]The history here is interesting. From very ancient times, astronomers have had models which predicted where every planet should appear in the sky at every point in time. (Now-a-days, we think of these parameters as describing the orbits of the Earth and the other planets around the Sun, but before Copernicus they had much more complicated models for the orbits of the Sun and the planets around the Earth.) If the model for, say, Mars had d parameters, the ancient practice was to pick d times when the location of Mars was known, so there was one equation per unknown, solve for the parameters, and then calculate where Mars should be at all other times. But with the accumulation of data, the astronomers usually had *more* observations than parameters, i.e., more equations than unknowns. Moreover it was clear that the same parameter values couldn't fit *all* the observations exactly, and that different choices of which data points to use would give different parameters and different predictions. For, literally, thousands of years, the practice was to try to find the "best" d observations, and use those to estimate the parameters. When other scientists, mostly physicists, also started coming up with quantitative models, they estimated parameters the same way. It wasn't until the 1700s that people started working on how to use *all* the observations, which they thought of as finding the best-fitting solution when there were more equations (observations) than unknowns (parameters). The method of least squares, as we know it, emerged around 1800 in this context, and it took about another hundred years for people to truly realize that you could use it when ϵ is a genuine fluctuation and not just an error of measurement. (On the technical history of statistics here, see Stigler (1986) and especially Farebrother (1999); on the conceptual shifts that were needed to go from "this is a way of dealing with errors of observation" to "this is a way of modeling how the world works", see Porter (1986) and Hacking (1990).)

particular θ_0. What will happen to the *expected* MSE? Use linearity of expectation:

$$\begin{aligned}
\mathbb{E}\left[MSE(\theta_0)\right] &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(X(r_i,t_i) - f(r_i,t_i;\theta_0))^2\right] && (3)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(\mu(r_i,t_i) + \epsilon(r_i,t_i) - f(r_i,t_i;\theta_0))^2\right] && (4)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(f(r_i,t_i;\theta_0) + \epsilon(r_i,t_i) - f(r_i,t_i;\theta_0))^2\right] && (5)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(\epsilon(r_i,t_i))^2\right] && (6)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(\epsilon(r_i,t_i))^2\right] && (7)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathrm{Var}\left[\epsilon(r_i,t_i)\right] && (8)
\end{aligned}$$

since $\mathbb{E}\left[\epsilon(r,t)\right] = 0$ everywhere. On the other hand, look at what happens if we use a *different* parameter value $\theta \neq \theta_0$; things start off almost the same but then take a turn:

$$\begin{aligned}
\mathbb{E}\left[MSE(\theta)\right] &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(X(r_i,t_i) - f(r_i,t_i;\theta))^2\right] && (9)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(\mu(r_i,t_i) + \epsilon(r_i,t_i) - f(r_i,t_i;\theta))^2\right] && (10)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(f(r_i,t_i;\theta_0) + \epsilon(r_i,t_i) - f(r_i,t_i;\theta))^2\right] && (11)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[((f(r_i,t_i;\theta_0) - f(r_i,t_i;\theta) + \epsilon(r_i,t_i))^2\right] && (12)\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[(f(r_i,t_i;\theta_0) - f(r_i,t_i;\theta))^2 + (f(r_i,t_i;\theta_0) - f(r_i,t_i;\theta))\epsilon(r_i,t_i) + \epsilon(r_i,t_i)^2\right] && (13)\\
&= \frac{1}{n}\sum_{i=1}^{n}(f(r_i,t_i;\theta_0) - f(r_i,t_i;\theta))^2 + \mathrm{Var}\left[\epsilon(r_i,t_i)\right] && (14)
\end{aligned}$$

since $(f(r_i,t_i;\theta_0) - f(r_i,t_i;\theta))$ is non-random and $\mathbb{E}\left[\epsilon(r_i,t_i)\right] = 0$. So

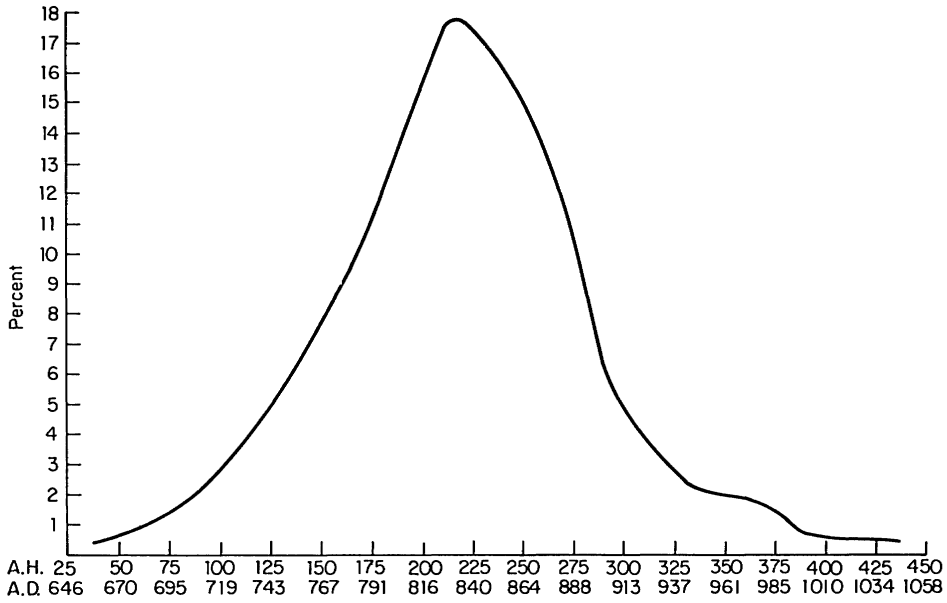
$$\mathbb{E}\left[MSE(\theta)\right] - \mathbb{E}\left[MSE(\theta_0)\right] = \frac{1}{n}\sum_{i=1}^{n}(f(r_i,t_i;\theta_0) - f(r_i,t_i;\theta))^2 > 0$$

In a word: the true parameter value has a smaller *expected* MSE than any other parameter value. If we think that the MSE from larger and larger data sets should converge on the expected MSE, then the least-squares estimates should converge on the true parameter values. When and why the large-sample MSE should converge is something we'll come back to later in the course.

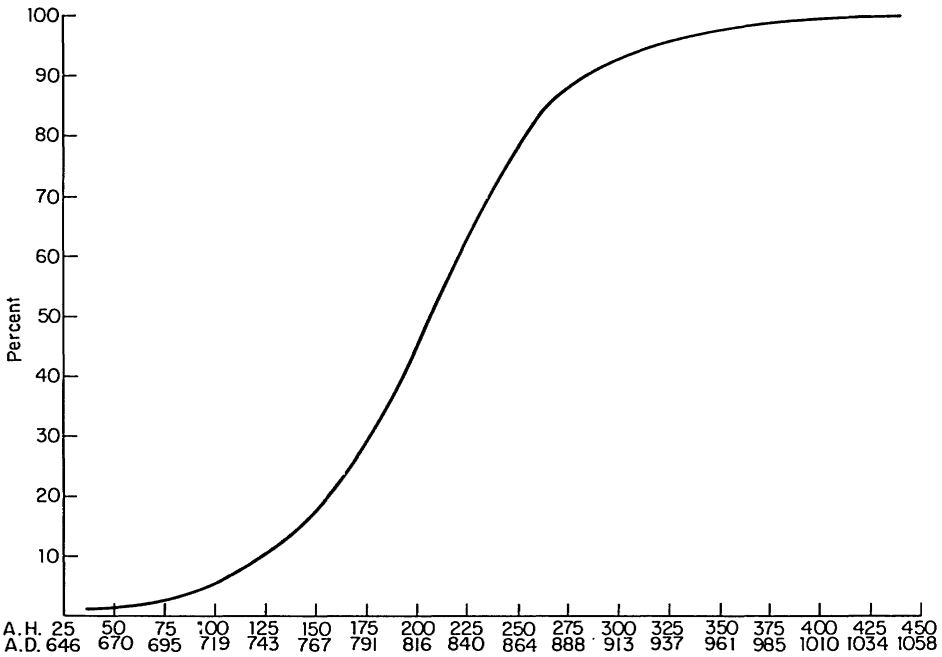
2.2.1.2 A small historical example

Here's a nice example of curve-fitting to get a trend, from Bulliet (1979). Using historical sources[5], Bulliet was able to work out when various people in medieval Iran converted to Islam, and so (by addition) what fraction of all the conversions happened between particular years:

[5]Here "medieval Iran" means, roughly, modern Iran and Afghanistan. Specifically, Bulliet used a collection of biographical dictionaries written in various Iranian cities about prominent local men (and a few prominent women). Each biography includes



By taking cumulative sums, this also told Bulliet what fraction of conversions had happened up to any given year:



Graph 3. Cumulative *S*-curve of Iranian conversion.

At this point, Bulliet borrowed a model from a subject in sociology called the "diffusion of innovations",

the full name of its subject in the old Arabic style which would translate as "A, the son of B, the son of C, the son of D, ... the son of W". By starting with the dates of the biography entry, and working backwards to the most recent ancestor with a non-Muslim name, and reckoning so many years per generation, Bulliet could estimate when each family had converted to Islam. (For Iran, it's easy to tell the difference between Muslim and non-Muslim names, because the non-Muslim names were in Persian, not Arabic. Also, while some Biblical names are also common Muslim names [Musa/Moses, Dawoud/David, Harun/Aaron, Isa/Jesus, etc.], the pre-Muslim Iranians were mostly not Christian or Jewish, and so Biblical names were rare among them. Bulliet also applied his method to Egypt and Spain, but results were less clear there, for precisely that reason — those were Christianized countries before the Muslim conquests.)

which is about how new ideas or practices spread by word of mouth[6]. Imagine there are two types of people: there are those who have already adopted the new idea, or those who haven't. Say that there are N people total, and the number who have adopted by time t is $A(t)$. When an adopter meets a non-adopter, they will pass on the idea with some probability c. Everybody meets everywhere body else more or less at random. The probability that an adopter meets a non-adopter is then $\frac{A(t)}{N}\frac{N-A(t)}{N} = a(t)(1-a(t))$, abbreviating $A(t)/N = a(t)$. So the average number of new adoptions between time t and $t+dt$ will be $Nca(t)(1-a(t))dt$, and we get a differential equation for the rate of change in $a(t)$,

$$\frac{da}{dt} = ca(1-a)$$

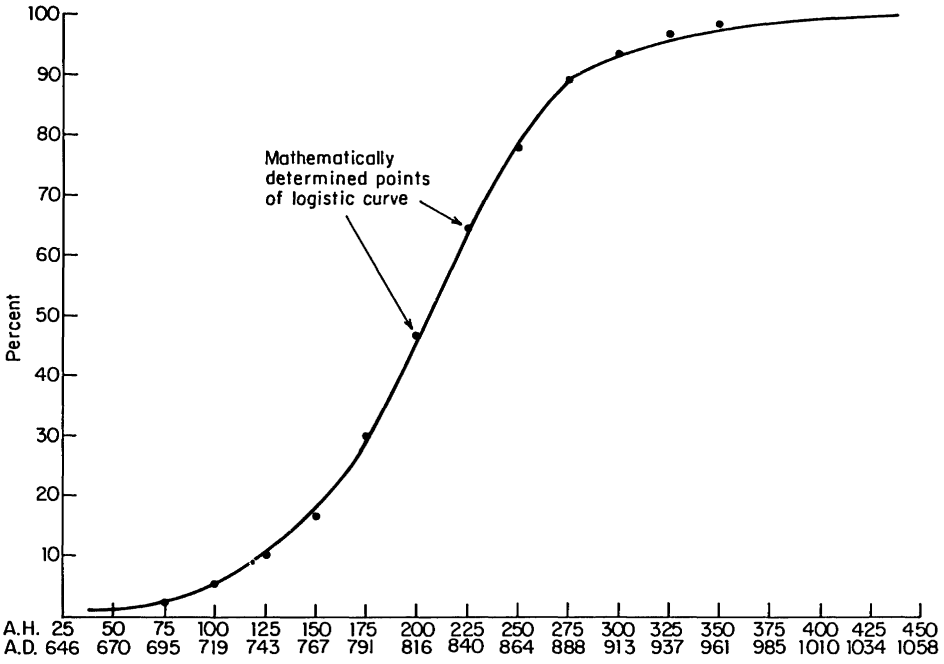
The solution (as you can check by differentiating) is

$$a(t) = a(0)\frac{a(0)e^{ct}}{1+a(0)(e^{ct}-1)}$$

Assuming $a(0) \ll 1$, and indeed $a(0) \approx 0$ (there are very few initial adoptees), at small times this is going to look like exponential growth. At large times, however, this is going to saturate at 1, because almost everyone has already adopted.

Notice that there's *one* unknown parameter, c, so that's the only knob we can adjust to fit the curve. (If we didn't know $a(0)$ we could also treat that as a parameter, but usually we do.)

This is a really simple model, but it has proven itself very, very good at fitting data about the adoption of all kinds of innovations; early successes included modeling the adoption of new crops, of color (as opposed to black-and-white) televisions, etc. (Rogers 2003). Here's what happened when Bulliet fit[7] this model to his data from medieval Iran:



Graph 4. Comparison of *S*-shaped conversion curve with logistic curve.

[6]The sociologists, in turn, had borrowed it from epidemiology; it's a simplified form of the usual kind of model of the spread of contagious diseases. (People have been comparing the spread of ideas they *don't* like to contagious diseases for at least 2000 years. What was original was to compare the spread of *all* ideas to the spread of diseases, and to know enough about how diseases spread to make the comparison mathematical and not just rhetorical.

[7]To be really precise, Bulliet (i) wrote the model in a different but equivalent form, in our notation $a(t) = \frac{1}{1+e^{(\theta_1-\theta_2 t)}}$ (Bulliet 1979, 160n8), and (ii) fit the curve by assuming it went through two of his data points, solving for θ_1 and θ_2, and then plotting the implied values of the curve at other t's (Bulliet 1979, 29). It would be interesting to re-do the estimation using least squares and *all* data points.

(This is an exceptionally good fit and you shouldn't feel bad if your data don't match a theoretical model this well.)

2.2.2 Trends from data

Instead of assuming a whole theory about How the World Works, we might, instead, make a different sort of assumption, which is just that μ doesn't change very rapidly — that it is a **smooth** function. The basic idea we need is that $\mu(t-h) \approx \mu(t) \approx \mu(t+h)$, at least for sufficiently small h, at least at most t. (You can write out the corresponding idea for spatio-temporal trends, with some more notation.) Each $X(t)$ is a noisy observation of $\mu(t)$, but if $\mu(t-h)$ and $\mu(t+h)$ are close to $\mu(t)$, then $X(t-h)$ and $X(t+h)$ will be noisy observations of things close to $\mu(t)$, giving us more information.

To make this concrete, and see how it can work, let's think of a very crude estimate:

$$\widehat{\mu}(t_i) = \frac{1}{3}\sum_{k=-1}^{k=+1} X(t_{i+k})$$

Each X is trend plus fluctuation, so

$$\begin{aligned}
\widehat{\mu}(t_i) &= \frac{1}{3}\sum_{k=-1}^{k=+1} \mu(t_{i+k}) + \frac{1}{3}\sum_{k=-1}^{k=+1} \epsilon(t_{i+k}) && (15)\\
&= \mu(t_i) + \frac{1}{3}\sum_{k=-1}^{k=+1} (\mu(t_{i+k}) - \mu(t_i)) + \frac{1}{3}\sum_{k=-1}^{k=+1} \epsilon(t_{i+k}) && (16)\\
&= \text{truth} + \text{bias} + \text{noise} && (17)
\end{aligned}$$

The first term is exactly what we want. The third term is noise, and the average of many noise terms is *generally* smaller than the individual noises. (If the ϵ were uncorrelated with equal variance σ^2, this would have variance $\sigma^2/3$.) The middle term is **bias**, systematic error. The more rapidly μ changes, and the further apart our observations are located, big this bias gets. Conversely, if μ changes slowly and our observations are closely spaced, the bias will be small.

At this point, we *could* go into considerable detail about exactly what the bias is, under various precise mathematical characterizations of "smooth function". (For instance, we could use Taylor approximation[8] to relate the bias to the first and second derivatives of μ.) But it'll be more productive to think about the more general *type* of estimator which includes the simple averaging of near-by values. There is, after all, no reason to just use the immediate neighbors, nor to use equal weights.

[8] As you remember from calculus, the Taylor expansion of a function f around a point t_0 is $f(t) = f(t_0) + ((t-t_0)f'(t_0) + \frac{1}{2}(t-t_0)^2 f''(t_0) + \ldots$, where f', f'', etc., are the first, second, etc., derivatives. In a Taylor approximation, we truncate the series at a certain order. The error of a k^{th} order approximation is $O((t-t_0)^{k+1})$.

3 Linear smoothers

We have data points $x(t_1), \ldots x(t_n)$; for conciseness I will sometimes abbreviate these as $x_1, \ldots x_n$.

The general form of a **linear smoother** is

$$\widehat{\mu}(t) = \sum_{j=1}^{n} w(t, t_j) x_j$$

That is, the predictions or fitted values, the estimates of the trend μ, are always linear combinations of the observed values of x. Notice that the smoother is linear in x, not in t — the resulting curve can be *very* nonlinear in t.

Those of you taking this course have (probably!) seen linear smoothers in previous statistics or machine learning courses, but it's worth reviewing some of their relevant properties, and examples, to have them front-of-mind.

3.1 Examples of linear smoothers

Most of the smoothing, regression and estimation techniques used in practice are linear smoothers.[^linearsmoothers]

- *Global constant*: The trivial case, $w = 1/n$ no matter what.
- *k Nearest neighbors* (kNN): $w(t, t_j) = 1/k$ if t_j is one of the k points closest to t, and otherwise $w = 0$.
- *Moving average* (MA): $w(t, t_j) = 0$ outside some distance from t, and constant within it.
  - If data points are always equally spaced in time (or space, etc.), moving averages and nearest neighbors are equivalent, but they can differ for irregularly-spaced data.
  - You can also do one-sided moving averages, usually for the past.
- *Weighted moving averages*: $w(t, t_j) = f(|t - t_j|/\tau)$, for some fixed function f. A common choice is the exponential function, giving **exponentially-weighted moving averages**. (One-sided exponentially-weighted moving averages are often used in practice for forecasting.)
- *Kernels*: Like moving averages, but with weights normalized, so we're always doing a proper average. That is, one picks a probability-density function K, and some scale, or **bandwidth**, h, and

$$w(t, t_j) = \frac{K(|t - t_j|/h)}{\sum_{j'=1}^{n} K(|t - t_{j'}|/h)}$$

- *Linear and polynomial regression*: It's not obvious, but these too are linear smoothers, though the weights are weird. (See the optional section on linear regression below.)
- *Splines*: These are smooth, piecewise polynomials, and so they, too, are linear smoothers. (We'll go over the exact formulas below.)

3.2 Properties of linear smoothers

The fact that lots of statistical methods can be seen as linear smoothers is useful because it gives us a unified way of understanding how these methods work, and in particular of when they will do a good job at recovering trends. Before establishing these properties, I need to introduce one last lump of notation.

At each t_i where we took an observation, we have a fitted value $\widehat{\mu}(t_i)$, or for short $\widehat{\mu}_i$. Group these together into a vector, or $n \times 1$ matrix,

$$\widehat{\mu} = \begin{bmatrix} \widehat{\mu}(t_1) \\ \widehat{\mu}(t_2) \\ \vdots \\ \widehat{\mu}(t_n) \end{bmatrix}$$

Similarly, group all the observations of x into a vector $\mathbf{x}$. Then the fitted vectors are the observations *times* a matrix:

$$\widehat{\mu} = \mathbf{w}\mathbf{x}$$

where $w_{ij} = w(t_i, t_j)$.

3.2.1 Expectation of the fitted values; bias

The first thing this lets us get at is the expected value of the fitted values:

$$\begin{aligned}
\mathbb{E}\left[\widehat{\mu}\right] &= \mathbb{E}\left[\mathbf{w}\mathbf{X}\right] && (18)\\
&= \mathbf{w}\mathbb{E}\left[\mathbf{X}\right] && (19)\\
&= \mathbf{w}\mathbb{E}\left[\mu + \epsilon\right] && (20)\\
&= \mathbf{w}\mathbb{E}\left[\mu\right] + \mathbf{w}\mathbb{E}\left[\epsilon\right] && (21)\\
&= \mathbf{w}\mu && (22)
\end{aligned}$$

because μ is a constant vector, and the noise ϵ always has expectation 0.

The **bias** of any estimator is the difference between its expected value and the truth. The estimator is **unbiased** when that difference is zero. So we have a simple equation for when a smoother gives us an unbiased estimate of the trend:

$$\text{unbiased trend estimates} \Leftrightarrow \mu = \mathbf{w}\mu \tag{23}$$

This says that we get an unbiased estimate if, and only if, μ is an eigenvector of $\mathbf{w}$ with eigenvalue 1. (See below for some reminders of the necessary linear algebra.) Otherwise, we get some bias in our estimate.

3.2.2 Shrinkage

Suppose (what is generally true) that $\mathbf{w}$ has n linearly independent eigenvectors, say $\mathbf{v}_1, \ldots \mathbf{v}_n$, with corresponding eigenvalues $\lambda_1, \ldots \lambda_n$. These eigenvectors form a basis for space of vectors, so for an arbitrary vector $\mathbf{x}$,

$$\mathbf{x} = \sum_{i=1}^{n} c_i \mathbf{v}_i$$

for some coordinates $c_1, \ldots c_n$. These coordinates say how much $\mathbf{x}$ projects on to the eigenvectors, i.e., how similar it is to each of the patterns those eigenvectors represent.

Now what happens when we multiply by $\mathbf{w}$, i.e., when we apply our favorite linear smoother to $\mathbf{x}$?

$$\mathbf{w}\mathbf{x} = \sum_{i=1}^{n} \mathbf{w} c_i \mathbf{v}_i = \sum_{i=1}^{n} c_i \lambda_i \mathbf{v}_i$$

The components of $\mathbf{x}$ which match eigenvectors $\mathbf{v}_i$ with large eigenvalues λ_i get enhance, and those which match eigenvectors with small eigenvalues get **shrunk**.

Most of the time, when we're dealing with smoothers, the largest eigenvalue is exactly 1. The eigenvectors with eigenvalue 1 define an **invariant** subspace, one which is left alone by the smoother. Because all the other eigenvalues are smaller, smoothing *shrinks* the part of the data which doesn't conform to the invariant subspace. If the eigenvalues which are less than 1 are in fact exactly 0, every component of the data which doesn't fall in the invariant subspace is projected away, and the vector of fitted values $\widehat{\mu}$ has to fall in the span of the leading eigenvectors. If the smaller eigenvalues aren't quite 0, the shrinkage isn't quite so extreme, and $\widehat{\mu}$ doesn't have to be in the invariant subspace, but it is always closer to that space than the data was.

1. If all the weights are non-negative, $w_{ij} \geq 0$, and the weights for each fitted value sum to 1, $\sum_j w_{ij} = 1$ for all i, then the largest eigenvalue of $\mathbf{w}$ is exactly 1, and all the other eigenvalues are ≤ 1 in magnitude. So when the linear smoother really does proper averages to get each fitted value, the vector of fitted values is shrunk towards the invariant subspace.
2. For linear regression, every eigenvalue of $\mathbf{w}$ is either 0 or 1. Hence vector of fitted values is always in the invariant subspace.

3.2.2.1 A little example

Let's go back to the idea of averaging each observation with the one ahead and behind it. We need to decide what to do with the first and last observations; I'll set them to weights of 1/2

```
n <- 10
w <- matrix(0, nrow=10, ncol=10)
diag(w) <- 1/3
for (i in 2:(n-1)) {
    w[i,i+1] <- 1/3
    w[i,i-1] <- 1/3
}
w[1,1] <- 1/2
w[1,2] <- 1/2
w[n,n-1] <- 1/2
w[n,n] <- 1/2
```

(You can check that the `w` this produces has the right entries.)

Here are the eigenvalues:

```
eigen(w)$values
```

```
##  [1]  1.00000000  0.96261129  0.85490143  0.68968376  0.48651845 -0.31012390
##  [7]  0.26920019 -0.23729622 -0.11137134  0.06254301
```

Notice, as promised, that 1 is an eigenvalue, and all the others are smaller in magnitude. The corresponding eigenvector is

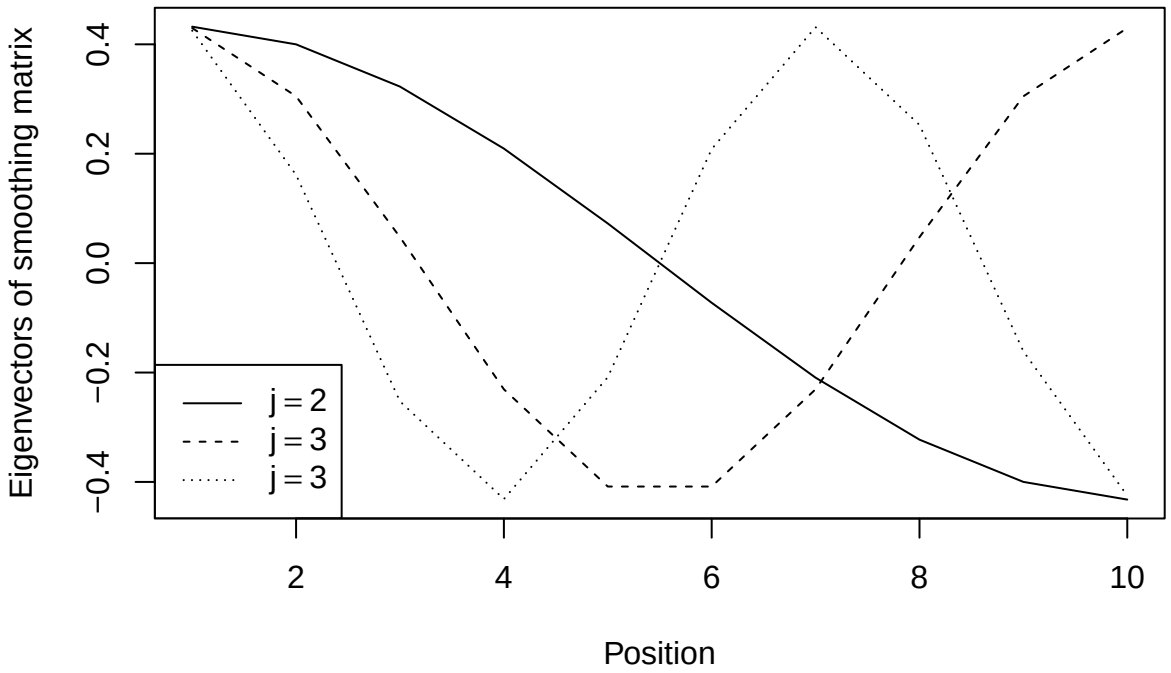
```
eigen(w)$vectors[,1]
```

```
##  [1] 0.3162278 0.3162278 0.3162278 0.3162278 0.3162278 0.3162278 0.3162278
##  [8] 0.3162278 0.3162278 0.3162278
```

i.e., *constant*. This should make sense — if every entry in $\mathbf{x}$ was the same, averaging wouldn't change anything.

The next few eigenvectors are better grasped by visualization than by looking at the numbers.

```
plot(1:10, eigen(w)$vectors[,2], lty="solid", type="l",
     xlab="Position", ylab="Eigenvectors of smoothing matrix")
lines(1:10, eigen(w)$vectors[,3], lty="dashed")
lines(1:10, eigen(w)$vectors[,4], lty="dotted")
legend("bottomleft", legend=c(expression(j==2), expression(j==3),
                              expression(j==3)),
       lty=c("solid","dashed","dotted"))
```



These look somewhat like sine waves, because they *are* somewhat like sine waves[9]. Because the eigenvalues which go with these waves are pretty close to 1, the components of the original data which match these sine waves will be shrunk just a little, while the other components of the data will be shrunk a lot. So these (plus the constant) are the sorts of patterns which this moving average will preserve; everything else it tends to ignore.

3.2.3 Variance of the fitted values

Because the data contains fluctuations, which are random, it makes sense to worry about the variance and covariance of the fitted values. These are easy:

$$\begin{aligned} \mathrm{Var}\left[\widehat{\mu}\right] &= \mathrm{Var}\left[\mathbf{w}\mathbf{X}\right] && (24)\\ &= \mathbf{w}\mathrm{Var}\left[\mathbf{X}\right]\mathbf{w}^T && (25)\\ &= \mathbf{w}\mathrm{Var}\left[\mu+\epsilon\right]\mathbf{w}^T && (26)\\ &= \mathbf{w}\mathrm{Var}\left[\epsilon\right]\mathbf{w}^T && (27) \end{aligned}$$

Of course, we don't necessarily know what $\mathrm{Var}\left[\epsilon\right]$ is. In other classes, we often assume that all of the noise terms have equal variance σ^2, and are uncorrelated with each other. That means $\mathrm{Var}\left[\epsilon\right] = \sigma^2\mathbf{I}$, and we get the special case

$$\mathrm{Var}\left[\widehat{\mu}\right] = \sigma^2\mathbf{w}\mathbf{w}^T$$

3.2.3.1 Covariance of Fitted Values; the Yule-Slutsky Effect

[9] Covering the connection between moving averages and sine waves in detail involves a mathematical subject called "Fourier analysis". This is generally important in applied math, and extremely useful for spatio-temporal data analysis, but not part of the usual statistics curriculum. We *may* cover it in the later part of the course as an advanced topic. For now, you can take the notion that the eigenvectors are nearly sine waves on trust. — If you are unwilling to take it on trust, think about how you'd approximate a 2nd derivative with finite differences: $\frac{\frac{x(t+h)-x(t)}{h} - \frac{(x(t)-x(t-h)}{h}}{h} = \frac{x(t+h)-2x(t)+x(t-h)}{h^2}$. So replacing $x(t)$ with the moving average is approximately the same as replacing $x(t)$ with $x(t)+\kappa x''(t)$ for some scaling factor κ. And you can verify for yourself that every sine or cosine function is an eigenfunction of this differential operator, i.e., that $\left(1+\kappa\frac{d^2}{dt^2}\right)\sin\omega t = \lambda\sin\omega t$. (Can you find λ?) So it should make some sense that the eigenvectors of the smoothing matrix are close to the eigenfunctions of that differential operator.

Let's look at that last equation again. Assuming $\mathrm{Var}[\epsilon] = \sigma^2 \mathbf{I}$ means that there's no correlation among the fluctuations, $\mathrm{Cov}[\epsilon(t), \epsilon(s)] = 0$ if $t \neq s$. This will be the case if, and only if, there's no correlation among the observations, $\mathrm{Cov}[X(t), X(s)] = 0$ if $t \neq s$. (Why?) But now look at what's going on with the fitted values:

$$\mathrm{Var}[\widehat{\mu}] = \sigma^2 \mathbf{w}\mathbf{w}^T$$

The covariances between the fitted values come from the off-diagonal entries of $\mathbf{w}\mathbf{w}^T$. If those off-diagonal entries aren't 0, there will be correlations among the fitted values. This in turn will generally happen when $\mathbf{w}$ isn't diagonal. Moral: *Smoothing creates correlations.*

The fact that smoothing creates correlations even when they're not present in the data we start with was discovered independently by two statisticians in the 1920s, G. Udny Yule in Britain and E. E. Slutsky in the USSR. Today it's called the "Yule-Slutsky effect", and it's one of the basic things to look out for when doing spatio-temporal data analyses.

(Interestingly, Yule and Slutsky had very different interpretations of their discovery. Yule was interested in finding correlations between variables as a first step towards causal inference, so he thought of correlations induced by smoothing as an artifact and a nuisance. He thought about how to discount for them, or even remove them. Slutsky, on the other hand, thought of averaging noise over time as an *explanation* of real correlations in the world. He specifically thought it explained "business cycles", the way capitalist economies alternate between booms and crashes. To his mind, the economy is constantly subject to random, uncorrelated shocks (a bad or good harvest, finding a new oil patch, technological inventions, etc.), but it takes some time for the economy to respond to them, for them to work their way through the system, as it were. So the state of the economy at any particular time is an average over all its recent shocks. That this gives fluctuations of a certain variance and rough frequency was, to him, a feature, not a bug.[10])

3.2.4 More about shrinkage; degrees of freedom

As the data randomly vary, the fitted values are going to randomly vary in response. One way to get a sense of how sensitive the fitted values are to the data is to look at the sum of the covariances,

$$\sum_{i=1}^{n} \mathrm{Cov}[\widehat{\mu}_i, X_i] = \sum_{i=1}^{n} \mathrm{Cov}\left[\sum_{j=1}^{n} w_{ij} X_j, X_i\right] \tag{28}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \mathrm{Cov}[X_i, X_j] \tag{29}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \mathrm{Cov}[\epsilon_i, \epsilon_j] \tag{30}$$

Again, in the special case where $\mathrm{Cov}[\epsilon_i, \epsilon_j] = 0$ if $i \neq j$, and $= \sigma^2$ if $i = j$, this simplifies to

$$\sum_{i=1}^{n} \mathrm{Cov}[\widehat{\mu}_i, X_i] = \sigma^2 \,\mathrm{tr}\, \mathbf{w}$$

Remembering that the trace of a matrix equals the sum of its eigenvalues, we use this to *define* the **effective number of degrees of freedom**:

$$\mathrm{DoF}\, \mathbf{w} \equiv \mathrm{tr}\, \mathbf{w}$$

If $\mathbf{w}$ is an idempotent projection matrix, this is exactly equal to the number of dimensions we're projecting down to. If $\mathbf{w}$ isn't a projection matrix, we pick up the dimension of the invariant space (= the number of $\lambda = 1$ eigenvalues), plus a bit more.

[10] The USSR in the 1920s being what it was, Slutsky had to do some fast talking to try to reconcile this with Marxism. (In particular, Marx had given his own theory of business cycles (Marx 1936, ch. 25), and while it wasn't very quantitative, it certainly didn't *sound* like "random stuff happens but it takes a while to be felt". Moreover, Marx thought crashes should be getting more and more destructive as capitalism advanced (Marx 1936, ch. 32), while Slutsky's model said their distribution should be stationary.) Slutsky was lucky to be allowed to escape into pure probability theory (Klein 1997, 276–79); many of his colleagues were not and went to the gulag or were otherwise purged.

3.3 Splines: Direct control over smoothness

This is a good place to introduce (or re-introduce) **splines**, more formally **smoothing splines**[11]. These are smooth curves which we find by balancing coming close to the data points, against not having too much wiggliness or curvature. More specifically, we pick a[12] $\lambda > 0$, and then solve the optimization problem

$$\widehat{\mu} = \operatorname*{argmin}_m \frac{1}{n}\sum_{i=1}^{n}(x_i - m(t_i))^2 + \lambda \int (m''(t))^2 dt$$

The first term we're optimizing over is the mean squared error. The second term is the average curvature of the function. It would be zero for any linear function (whose second derivative is 0 everywhere), and it grows the more the function bends and twists around. One way to read this is that we are **penalizing** curvature. In fact, you can think of λ as a price — we're willing to trade one unit more of curvature for at least λ units of reduced MSE. (Conversely, we're willing to increase the MSE by one unit, if in doing so we reduce the curvature by at least $1/\lambda$ units.) As $\lambda \to 0$, we get incredibly wiggly functions which go through the data points exactly. As $\lambda \to \infty$, we approach the best-fitting straight line, i.e., linear regression.

The optimization here is over all possible functions with second derivatives. You might well ask how we could possibly do this; the space of functions seems hopelessly vast to search over. (It's certainly infinite-dimensional.) The answer is that the solution is *always* a piecewise cubic polynomial, with pieces at the data points t_i, and that the solution is also continuous, with continuous first and second derivatives. (See the optional section below, if you care.) All of this means that we only have n unknown coefficients to find, rather than having to search over all possible functions. (Again, see the optional section below.)

One upshot of being a piecewise polynomial is that the spline is a linear smoother, with weights which do a kind of local averaging. Another is that there is a direct correspondence between λ and the effective degrees of freedom, where increasing λ lowers the degrees of freedom and vice versa. (The exact relationship depends on the values of t_i — again, see the optional section.)

3.3.1 Multi-dimensional splines

All of the above is for one-dimensional smoothing. If we're dealing with data over two dimensions, so it's really $x(r,t)$ rather than just $x(t)$, we can define the analogous problem:

$$\operatorname*{argmin}_m \frac{1}{n}\sum_{i=1}^{n}(x_i - m(r_i,t_i))^2 + \lambda \int \left[\left(\frac{\partial^2 m}{\partial r^2}\right)^2 + 2\left(\frac{\partial^2 m}{\partial r \partial t}\right)^2 + \left(\frac{\partial^2 m}{\partial t^2}\right)^2\right] dr dt$$

(Three or more coordinates is similar, but with even more notation.) These are called **thin-plate** splines.

An alternative approach is to use what are called a "tensor-product splines". If you dive into the math (see the optional section), you find that one-dimensional splines are linear combinations of "basis functions" $B_1(t), \ldots B_n(t)$ — the exact shape of the basis functions depends on the t_i but not on the x_i. In two dimensions, we can find the basis functions for the two coordinates, say B_i for r and C_j for t, and take products, so

$$m(r,t) = \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}B_i(r)C_j(t)$$

- Thin-plate splines have the nice interpretation of trading curvature against ME, just like in 1D. Tensor-product splines are harder to interpret.
  - On the other hand, this interpretation itself makes more sense when the coordinates have similar scales and meaning — if r and t are both map coordinates, total curvature is a lot more meaningful than if r is space and t is time.
- Tensor-product splines are usually easier to calculate than thin-plate splines.

[11] People have defined many variants on splines. When I use the word without an adjective, I mean smoothing splines, as defined by the optimization problem immediately following this footnote.

[12] I apologize for the fact that this is not an eigenvalue. But everyone uses λ for this control setting, and everyone uses λ for eigenvalues, and I will trust you to put them in separate name-spaces in your mind.

3.3.2 Splines in R

The simplest way to work with splines in R is to use the `smooth.spline` command. This takes two vectors, `x` (corresponding to our t variable) and `y` (corresponding to our x variable), and returns a slightly complicated object. The `x` and `y` components contain the `x` values, and the *fitted* `y` values, so it can be used for plotting. (See examples in the slides for lecture 1 and lecture 4.) You can also calculate predictions at *new* values, not corresponding to the data: if `ss` is your spline object, `predict(ss, x)` will return a list, where the `$x` component are the values in the input vector `x` in order, and the `$y` component are the predicted values of the spline at those coordinates.

For more sophisticated uses, especially multi-dimensional splines, you need to use a package. I recommend the `mgcv` package (Wood 2004,@Wood-on-GAMs), which is primarily designed for smooth, additive models, and as such includes a lot of different smoothers. A command like

```
library(mgcv)
my.fit <- gam(x ~ s(t), data=df)
```

estimates a regression model where `x` (drawn from data frame `df`) is a smooth spline function of `t` (drawn from the same source), and stores the result in the object `my.fit`. On the other hand,

```
gam(x ~ s(r,t), data=df)
```

models `x` as a thin-plate spline over the two variables `r` and `t`, while

```
gam(x ~ te(r, t), data=df)
```

uses a tensor-product spline. If you want to model `x` as the *sum* of two smooth functions, that would be

```
gam(x ~ s(r) + s(t), data=df)
```

There are also a large number of more advanced packages exclusively for dealing with splines — too many for me to list here.

3.3.3 Hodrick-Prescott filtering

In econometrics, it is very common to use what is called a "Hodrick-Prescott filter" to extract the trend of a time series. (The canonical reference is Hodrick and Prescott (1997), but this circulated as a working paper for many years before being formally published.) As shown by Paige and Trindade (2010), this is *exactly* the same as applying a smoothing spline to the data. The only difference is that Hodrick and Prescott supplied an ingenious, but quite fallacious, argument for why λ should always be exactly 1600.

4 Choosing How, or How Much, to Smooth — Cross-validation

Pretty much any smoothing method we might use has some adjustable control settings[13] — the size of the moving-average window, say, or the "bandwidth" of a kernel — which we need to set. How should we tune these control settings?

One of the simplest, but also most practical, ways of picking control settings is to **make the smoother predict well on data it didn't see**. The uses to which we're putting our smoother — interpolating to fill in gaps in the data, extrapolating beyond the range of the data, filtering to remove noise from the data — are all, in the end, about predicting unseen values. The most natural way to make sure the smoother can predict unseen values is to *have it predict unseen values*.

This brings us to the idea of **cross-validation**. There are many variants, but the key notion is this:

- Take the data set, and divide the data points into a **training set** and a **testing set**.
- Fit each smoother (or other model) to the training set.
- Evaluate each smoother on the testing set, say by taking the squared error.
- Average the scores for each model over different choices of training and testing sets.

This gives a score for each smoother, which says how well it was able to predict data which it *didn't* get to see during the fitting or training process. This is exactly what we want.

4.1 Leave-one-out Cross-validation

The most straight-forward form of this is **leave-one-out cross-validation** (LOOCV).[14] We do the following for each smoother we're considering:

- For each of the n data points:
  - Fit each smoother using every data point *except* data point i; call this function $\widehat{\mu}^{(-i)}$;
  - Calculate the prediction at data point i, $\widehat{\mu}^{(-i)}(t_i)$;
  - Calculate squared[15] prediction error $(x_i - \widehat{\mu}^{(-i)}(t_i))^2$.
- Average the squared prediction errors for all models over all n data points, to get $n^{-1}\sum_{i=1}^{n}(x_i - \widehat{\mu}^{(-i)}(t_i))^2$.

This makes it sound as though, with LOOCV, we need to estimate each possible smoother n times, which would be slow for a large data set. There is in fact a trick for linear smoothers which lets us fit the smoother just once. What happens when we hold back data point i, and then make a prediction at t_i? The observed response at i can't contribute to the prediction, but otherwise the linear smoother should work as before, so

$$\widehat{\mu}^{(-i)}(t_i) = \frac{(\mathbf{wx})_i - w_{ii}x_i}{1 - w_{ii}}$$

The numerator just removes the contribution to $\widehat{\mu}(t_i)$ that came from x_i, and the denominator just re-normalizes the weights in the smoother. Now a little algebra says that

$$x_i - \widehat{\mu}^{(-i)}(t_i) = \frac{x_i - \widehat{\mu}(x_i)}{1 - w_{ii}}$$

The quantity on the left of that equation is what we want to square and average to get the leave-one-out CV score, but everything on the right can be calculated from the fit we did to the whole data. The leave-one-out CV score is therefore

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \widehat{\mu}(t_i)}{1 - w_{ii}}\right)^2 \tag{31}$$

Again, this formula requires *both* leave-one-out-CV *and* a linear smoother.

[13] Some people call these "hyperparameters", rather than "control settings". I think this is a misleading name, because they're not in any way parameters of the process we're studying, they're just aspects of *our* method.

[14] Some of the text in this section is modified from Shalizi (n.d.).

[15] The same idea works if you'd rather use the absolute error, a "robust" loss function, etc.

If we had independent observations, you might think that LOOCV would give us n independent, unbiased estimates of how well a smoother predicts new data, and so an unbiased, low-variance estimate of model performance. The problem is that even if the *data points* are independent, $\widehat{\mu}^{(-i)}$ and $\widehat{\mu}^{(-j)}$ are based on $n-2$ shared data points, so the leave-one-out errors for each data point are pretty correlated with each other. (Can you work out the covariance between $x_i - \widehat{\mu}^{(-i)}(t_i)$ and $x_j - \widehat{\mu}^{(-j)}(t_j)$?) One consequence of this is that LOOCV tends to over-fit — to pick models with more degrees of freedom than are strictly necessary.

We sometimes therefore prefer to use *many* data points in each testing set, so the predictions are less correlated with each other. The commonest approach is **k-fold CV**, where we divide all the data points into k equal groups, or "folds", and each fold gets to be the testing set in turn. (Typically $k = 5$ or 10.) This is much less prone to over-fitting than leave-one-out.

4.2 Special issues on cross-validation for dependent data

We will return to this topic towards the end of the course.

These forms of cross-validation work very, very well for independent data, to the point where they're pretty much the standard for any situation involving statistical modeling of independent data. But what about dependent data? If each data point tends to be like its neighbors, you might well worry that a smoother which *just* copies the neighbors will have an unfair advantage in cross-validation. At the very least, this would seem to lead to more noise in our estimate of how well each smoother predicts.

A natural way out of this is to do leave-one-out cross-validation, but to leave out a buffer of size (say) h around each point (Burman, Chow, and Nolan 1994). This buffer is not part of the training set, but we also don't use it in evaluating the prediction of the smoother. (We *do* use values in the buffer when we *calculate* the prediction at the held-out point, however.) If remote observations are nearly independent of each other, and the buffer is big enough, then the testing set is nearly independent of the training set, and everything is well-behaved. You can combine this idea with k-fold cross-validation, where you omit a buffer around each block (Racine 2000).

The obvious question is "how big a buffer?", to which the obvious answer is "big enough to make the training and test points nearly independent". Since we usually don't know that, we typically rely on rules of thumb. Burman, Chow, and Nolan (1994) suggest using a constant fraction of n, say $h = 0.5n$. Racine (2000) suggests that this rule is prone to over-fitting, and that a better one would be $h = n^{0.25}$. I don't know of a really great, automatic resolution of this issue.

In practice, it can often be easiest to just live with the extra imprecision and over-fitting that come from ignoring dependence, but we'll come back to this later when we look in more detail at time-series forecasting.

4.3 Smoothing ratios and proportions

Before moving on from smoothers, I should mention one last special consideration. When our data consists of a ratio or proportion, say $X(t) = Y(t)/Z(t)$, we *could* smooth it the same way we'd smooth any other numerical data. However, if the denominator is small, any noise in it can have a massive effect on the ratio. It can make a lot more sense to *separately* smooth Y and Z, and then take the ratio of the smoothed values. Kafadar (1996), from whom I learned this point, explores how best to do this with splines and related smoothers.

5 Detrended values and residuals

After we have chosen how to smooth the data, and gotten an estimate of the trend, we also have an estimate of the fluctuations. Since we can always write

$$X(r,t) = \mu(r,t) + \epsilon(r,t)$$

where μ is a deterministic (but unknown) function and ϵ is a stochastic process, the natural estimate of the latter is

$$\widehat{\epsilon}(r,t) \equiv X(r,t) - \widehat{\mu}(r,t)$$

These natural estimates of the noise are the **residuals** of the smoother.

When we use a linear smoother, we can find the residuals in matrix form:

$$\begin{aligned} \widehat{\epsilon} &= \mathbf{x} - \widehat{\mu} && (32)\\ &= \mathbf{x} - \mathbf{w}\mathbf{x} && (33)\\ &= (\mathbf{I} - \mathbf{w})\mathbf{x} && (34) \end{aligned}$$

Thus the residuals, like the fitted values, also linear in $\mathbf{x}$. We can therefore work out properties of the residuals much as we did properties of the fitted values.

The expected value of the residual vector is

$$\begin{aligned} \mathbb{E}\left[\widehat{\epsilon}\right] &= \mathbb{E}\left[(\mathbf{I} - \mathbf{w})\mathbf{X}\right] && (35)\\ &= (\mathbf{I} - \mathbf{w})\mu && (36) \end{aligned}$$

This will be zero — i.e., match the true expectation of the noise — if and only if μ is an eigenvalue of $\mathbf{I} - \mathbf{w}$ with eigenvalue 0.

— It should not be hard to convince yourself that if $\mathbf{v}$ is an eigenvector of $\mathbf{w}$ with eigenvalue λ, then $\mathbf{v}$ is also an eigenvector of $\mathbf{I} - \mathbf{w}$, with eigenvalue $1 - \lambda$. It should also not be hard to check that if $\mathbf{w}$ is symmetric and/or idempotent, then so is $\mathbf{I} - \mathbf{w}$.

The variance of the residuals is

$$\mathrm{Var}\left[\widehat{\epsilon}\right] = (\mathbf{I} - \mathbf{w})\mathrm{Var}\left[\epsilon\right](\mathbf{I} - \mathbf{w})^T$$

Under the special assumption that the true noise has constant variance and is uncorrelated, $\mathrm{Var}\left[\epsilon\right] = \sigma^2\mathbf{I}$, and this simplifies to

$$\sigma^2(\mathbf{I} - \mathbf{w})(\mathbf{I} - \mathbf{w})^T$$

Notice that in general this is *not* $\sigma^2\mathbf{I}$. That is, the residuals generally have somewhat different variances, and correlation structure, than the true noise. This, too, is a manifestation of the Yule-Slutsky effect.

6 Appendix: Some mathematical details for splines

This section is optional, but perhaps interesting. The text is largely ripped off from Shalizi (n.d.).

6.1 Solutions to the spline problem are piecewise-cubic polynomials

The spline problem asks us to find the function which minimize the sum of the MSE and a certain integral. Even the MSE can be brought inside the integral, using Dirac delta functions[16]:

$$\mathcal{L} = \int \left[\lambda(m''(t))^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - m(t_i))^2 \delta(t - t_i) \right] dt \tag{37}$$

In what follows, without loss of generality, assume that the t_i are ordered, so $t_1 \leq t_2 \leq \ldots t_i \leq t_{i+1} \leq \ldots t_n$. With some loss of generality but a great gain in simplicity, assume none of the t_i are equal, so we can make those inequalities strict.

The subject which deals with maximizing or minimizing integrals of functions is the **calculus of variations**, and one of its basic tricks is to write the integrand as an expression involving x, the function, and its derivatives:

$$\mathcal{L} = \int L(t, m, m', m'')dt \tag{38}$$

where, in our case,

$$L = \lambda(m''(t))^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - m(t_i))^2 \delta(t - t_i) \tag{39}$$

This sets us up to use a general theorem of the calculus of variations, to the effect that any function $\hat{m}$ which minimizes L must also solve L's **Euler-Lagrange equation**:

$$\frac{\partial L}{\partial m} - \frac{d}{dt}\frac{\partial L}{\partial m'} + \frac{d^2}{dt^2}\frac{\partial L}{\partial m''}\bigg|_{m=\hat{m}} = 0$$

In our case, the Euler-Lagrange equation reads

$$-\frac{2}{n}\sum_{i=1}^{n}(x_i - \hat{m}(t_i))\delta(t - t_i) + 2\lambda\frac{d^2}{dt^2}\hat{m}''(t) = 0$$

Remembering that $\hat{m}''(t) = d^2\hat{m}/dt^2$,

$$\frac{d^4}{dt^4}\hat{m}(t) = \frac{1}{n\lambda}\sum_{i=1}^{n}(x_i - \hat{m}(t_i))\delta(t - t_i)$$

The right-hand side is zero at any point t other than one of the t_i, so the fourth derivative has to be zero in between the t_i. This in turn means that the function must be piecewise cubic. Now fix an t_i, and pick any two points which bracket it, but are both greater than t_{i-1} and less than t_{i+1}; call them l and u. Integrate our Euler-Lagrange equation from l to u:

$$\int_l^u \frac{d^4}{dt^4}\hat{m}(t)dt = \int_l^u \frac{1}{n\lambda}\sum_{i=1}^{n}(x_i - \hat{m}(t_i))\delta(t - t_i) \tag{40}$$

$$\hat{m}'''(u) - \hat{m}'''(l) = \frac{x_i - \hat{m}(t_i)}{n\lambda} \tag{41}$$

[16]The Dirac delta function (named after the physicist P. A. M. Dirac) is defined by how it integrates: for any function $f(t)$, $\int f(t)\delta(t)dt = f(0)$. You can think of this as the limit of a sequence of Gaussian pdfs, all centered at 0, with variances shrinking towards zero. The limit would be something like a function δ which was 0 except at the origin, but had an infinitely skinny, infinitely tall spike at the origin, with the total area under the curve being exactly 1. There isn't really a *function* like that, but it's definitely true that $\lim_{\sigma\to 0}\int f(t)\frac{1}{\sigma}phi(t/\sigma)dt = f(0)$, so $\delta(t)$ makes sense *in integrals*. One can in fact build a whole theory of "generalized functions" which make sense in integrals in this way, and which include all the actual functions as special cases of generalized functions. If this interests you, Kolmogorov and Fomin (1975) is still one of the best ways in to the subject. (Dirac, being a physicist, didn't let something like δ not really being a function stop him from calculating.)

That is, the third derivative makes a jump when we move across t_i, though (since the fourth derivative is zero), it doesn't matter which pair of points above and below t_i we compare third derivatives at. Integrating the equation again,

$$\hat{m}''(u) - \hat{m}''(l) = (u-l)\frac{x_i - \hat{m}(t_i)}{n\lambda} \tag{42}$$

Letting u and l approach x_i from either side, so $u - l \to 0$, we see that $\hat{m}''$ makes no jump at t_i. Repeating this trick twice more, we conclude the same about $\hat{m}'$ and $\hat{m}$ itself. In other words, $\hat{m}$ must be continuous, with continuous first and second derivatives, and a third derivative that is constant on each (t_i, t_{i+1}) interval. Since the fourth derivative is zero on those intervals (and undefined at the t_i), the function must be a piecewise cubic, with the piece boundaries at the t_i, and continuity (up to the second derivative) across pieces.

To see that the optimal function must be linear below t_1 and above t_n, suppose that it wasn't. Clearly, though, we could reduce the curvature as much as we want in those regions, without altering the value of the function at the boundary, or even its first derivative there. This would yield a better function, i.e., one with a lower value of $\mathcal{L}$, since the MSE would be unchanged and the average curvature would be smaller. Taking this to the limit, then, the function must be linear outside the observed data range.

6.2 Finding the spline coefficients

Splines, I said, are piecewise cubic polynomials. To see how to fit them, let's think about how to fit a global cubic polynomial. We would define four **basis functions**,

$$B_1(x) = 1 \tag{43}$$
$$B_2(x) = t \tag{44}$$
$$B_3(x) = t^2 \tag{45}$$
$$B_4(x) = t^3 \tag{46}$$

and chose to only consider regression functions that are linear combinations of the basis functions,

$$m(t) = \sum_{j=1}^{4} \beta_j B_j(t) \tag{47}$$

Such regression functions would be linear in the *transformed* variables $B_1(t), \ldots B_4(t)$, even though it is nonlinear in t.

To estimate the coefficients of the cubic polynomial, we would apply each basis function to each data point t_i and gather the results in an $n \times 4$ matrix $\mathbf{b}$,

$$b_{ij} = B_j(t_i) \tag{48}$$

Then we would do ordinary least squares using the $\mathbf{b}$ matrix in place of the usual data matrix $\mathbf{t}$:

$$\hat{\beta} = (\mathbf{b}^T\mathbf{b})^{-1}\mathbf{b}^T\mathbf{x} \tag{49}$$

Since splines are piecewise cubics, things proceed similarly, but we need to be a little more careful in defining the basis functions. Recall that we have n values of t, $t_1, t_2, \ldots t_n$, and we can assume they're in increasing order. These n **knots** define $n+1$ pieces or segments: $n-1$ of them between the knots, one from $-\infty$ to t_1, and one from t_n to $+\infty$. A third-order polynomial on each segment would seem to need a constant, linear, quadratic and cubic term per segment. So the segment running from t_i to t_{i+1} would need the basis functions

$$\mathbf{1}_{(t_i,t_{i+1})}(t),\ (t-t_i)\mathbf{1}_{(t_i,t_{i+1})}(t),\ (t-t_i)^2\mathbf{1}_{(t_i,t_{i+1})}(t),\ (t-t_i)^3\mathbf{1}_{(t_i,t_{i+1})}(t) \tag{50}$$

where as usual the indicator function $\mathbf{1}_{(t_i,t_{i+1})}(t)$ is 1 if $t \in (t_i, t_{i+1})$ and 0 otherwise. This makes it seem like we need $4(n+1) = 4n+4$ basis functions.

However, we know from linear algebra that the number of basis vectors we need is equal to the number of dimensions of the vector space. The number of adjustable coefficients for an arbitrary piecewise cubic with

$n+1$ segments is indeed $4n+4$, but splines are constrained to be smooth. The spline must be continuous, which means that at each t_i, the value of the cubic from the left, defined on (t_{i-1}, t_i), must match the value of the cubic from the right, defined on (t_i, t_{i+1}). This gives us one constraint per data point, reducing the number of adjustable coefficients to at most $3n+4$. Since the first and second derivatives are also continuous, we are down to just $n+4$ coefficients. Finally, we know that the spline function is linear outside the range of the data, i.e., on $(-\infty, t_1)$ and on (t_n, ∞), lowering the number of coefficients to n. There are no more constraints, so we end up needing only n basis functions. And in fact, from linear algebra, any set of n piecewise cubic functions which are linearly independent can be used as a basis. One common choice is

$$
\begin{aligned}
B_1(t) &= 1 && (51)\\
B_2(t) &= t && (52)\\
B_{i+2}(x) &= \frac{(t-t_i)^3_+ - (t-t_n)^3_+}{t_n - t_i} - \frac{(t-t_{n-1})^3_+ - (t-t_n)^3_+}{t_n - t_{n-1}} && (53)
\end{aligned}
$$

where $(a)_+ = a$ if $a > 0$, and $= 0$ otherwise. This rather unintuitive-looking basis has the nice property that the second and third derivatives of each B_j are zero outside the interval (t_1, t_n).

Now that we have our basis functions, we can once again write the spline as a weighted sum of them,

$$
m(t) = \sum_{j=1}^{m} \beta_j B_j(t) \tag{54}
$$

and put together the matrix $\mathbf{b}$ where $b_{ij} = B_j(t_i)$. We can write the spline objective function in terms of the basis functions,

$$
n\mathcal{L} = (\mathbf{x} - \mathbf{b}\beta)^T(\mathbf{x} - \mathbf{b}\beta) + n\lambda\beta^T\Omega\beta \tag{55}
$$

where the matrix Ω encodes information about the curvature of the basis functions:

$$
\Omega_{jk} = \int B_j''(t) B_k''(t) dt \tag{56}
$$

Notice that only the quadratic and cubic basis functions will make non-zero contributions to Ω. With the choice of basis above, the second derivatives are non-zero on, at most, the interval (t_1, t_n), so each of the integrals in Ω is going to be finite. Ω is something we (or, realistically, R) can calculate *once*, no matter what λ is. Now we can find the smoothing spline by differentiating with respect to β:

$$
\begin{aligned}
0 &= -2\mathbf{b}^T\mathbf{x} + 2\mathbf{b}^T\mathbf{b}\hat{\beta} + 2n\lambda\Omega\hat{\beta} && (57)\\
\mathbf{b}^T\mathbf{x} &= \left(\mathbf{b}^T\mathbf{b} + n\lambda\Omega\right)\hat{\beta} && (58)\\
\hat{\beta} &= \left(\mathbf{b}^T\mathbf{b} + n\lambda\Omega\right)^{-1}\mathbf{b}^T\mathbf{x} && (59)
\end{aligned}
$$

Notice, incidentally, that we can now show splines are linear smoothers:

$$
\begin{aligned}
\widehat{\mu} &= \mathbf{b}\hat{\beta} && (60)\\
&= \mathbf{b}\left(\mathbf{b}^T\mathbf{b} + n\lambda\Omega\right)^{-1}\mathbf{b}^T\mathbf{x} && (61)
\end{aligned}
$$

Once again, if this were ordinary linear regression, the OLS estimate of the coefficients would be $(\mathbf{t}^T\mathbf{t})^{-1}\mathbf{t}^T\mathbf{x}$. In comparison to that, we've made two changes. First, we've substituted the basis-function matrix $\mathbf{b}$ for the original matrix of predictor variables, $\mathbf{t}$ — a change we'd have made already for a polynomial regression. Second, the "denominator" is not $\mathbf{t}^T\mathbf{t}$, or even $\mathbf{b}^T\mathbf{b}$, but $\mathbf{b}^T\mathbf{b} + n\lambda\Omega$. Since $\mathbf{t}^T\mathbf{t}$ is n times the covariance matrix of the independent variables, we are taking the covariance matrix of the spline basis functions and adding some extra covariance — how much depends on the shapes of the functions (through Ω) and how much smoothing we want to do (through λ). The larger we make λ, the less the actual data matters to the fit.

In addition to explaining how splines can be fit quickly (do some matrix arithmetic), this illustrates two important tricks. One, which we won't explore further here, is to turn a nonlinear regression problem into one

which is linear in another set of basis functions. This is like using not just one transformation of the input variables, but a whole library of them, and letting the data decide which transformations are important. There remains the issue of selecting the basis functions, which can be quite tricky. In addition to the spline basis, most choices are various sorts of waves — sine and cosine waves of different frequencies, various wave-forms of limited spatial extent ("wavelets"), etc. The ideal is to chose a function basis where only a few non-zero coefficients would need to be estimated, but this requires some understanding of the data...

The other trick is that of stabilizing an unstable estimation problem by adding a penalty term. This reduces variance at the cost of introducing some bias.

6.3 Effective degrees of freedom

Since the effective degrees of freedom of a linear smoother is $\mathrm{tr}\,\mathbf{w}$, we've just shown that the a spline has

$$\mathrm{tr}\left(\mathbf{b}\left(\mathbf{b}^T\mathbf{b} + n\lambda\Omega\right)^{-1}\mathbf{b}^T\right)$$

degrees of freedom. Holding the data (and so $\mathbf{b}$ and Ω) fixed, as λ goes up, this will go down, and vice versa.

7 Appendix: Linear regression as a linear smoother

This section can be skipped, if you're willing to take the fact that linear regression is really, or also, a linear smoother on trust. We will be coming back to some of this stuff later, when we look at linear predictors more generally.

7.1 Simple linear regression as a linear smoother

Think of the simple linear regression model:

$$m(t) = \beta_0 + \beta_1 t$$

We usually estimate this by minimizing the mean squared error (MSE) — what's called the **ordinary least sqaures** or OLS estimate:

$$(\hat{\beta}_0, \hat{\beta}_1) = \underset{b_0, b_1}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (x_i - (b_0 + b_1 t_i))^2 \tag{62}$$

Take the derivatives with respect to b_0 and b_1, and set them to zero.

$$\frac{1}{n} \sum_{i=1}^{n} 2(x_i - (\hat{\beta}_0 + \hat{\beta}_1 t_i))(-1) = 0 \tag{63}$$

$$\frac{1}{n} \sum_{i=1}^{n} 2(x_i - (\hat{\beta}_0 + \hat{\beta}_1 t_i))(-t_i) = 0 \tag{64}$$

We can drop the over-all factor of -2 in what follows.

First, solve for $\hat{\beta}_0$ in terms of $\hat{\beta}_1$:

$$\overline{x} - \hat{\beta}_0 - \hat{\beta}_1 \overline{t} = 0 \tag{65}$$

$$\overline{x} - \hat{\beta}_1 \overline{t} = \hat{\beta}_0 \tag{66}$$

Now solve for $\hat{\beta}_1$:

$$\overline{xt} - \hat{\beta}_0 \overline{t} - \hat{\beta}_1 \overline{t^2} = 0 \tag{67}$$

$$\overline{xt} - \overline{x}\overline{t} + \hat{\beta}_1 (\overline{t})^2 - \hat{\beta}_1 \overline{t^2} = 0 \tag{68}$$

$$\frac{\overline{xt} - \overline{x}\overline{t}}{\overline{t^2} - \overline{t}^2} = \hat{\beta}_1 \tag{69}$$

It's convenient to abbreviate the denominator as s_t^2.

Putting this together,

$$\widehat{\mu}(t) = \overline{x} + \frac{t - \overline{t}}{s_t^2} (\overline{xt} - \overline{x}\overline{t})$$

This is linear in x, so we know it can be put in the linear-smoother form. In fact,

$$\widehat{\mu}(t) = \sum_{j=1}^{n} \left(\frac{1}{n} + \frac{(t - \overline{t})(t_j - \overline{t})}{n s_t^2} \right) x_j$$

This particular form of weights ensures both that the estimated curve is a straight line, and that it comes as close to the data as possible, on average, among all straight lines.

Beyond that, however, the weights are really weird. The first part is just the sample mean, but the variable part is $\propto (t - \overline{t})(t_j - \overline{t})$. This means we give more weight to data points which are far from the center of the data, no matter what. We also ramp up all the weights as the point t where we're making a prediction moves away from the center of the data.

7.2 Linear regression with multiple predictors

Suppose we want to make our prediction a linear combination of multiple variables:

$$m(t, z, \ldots u) = \beta_0 + \beta_1 t + \beta_2 z + \ldots \beta_p u$$

We fit this by least squares again. It simplifies the math immensely if we define two matrices,

$$\mathbf{t} = \begin{bmatrix} 1 & t_1 & z_1 & \ldots & u_1 \\ 1 & t_2 & z_2 & \ldots & u_2 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & t_n & z_n & \ldots & u_n \end{bmatrix}$$

and

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix}$$

Then the fitted values are

$$\mathbf{m} = \mathbf{tb}$$

and the mean-squared error is

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - (b_0 + b_1 t_i + b_2 z_i + \ldots + b_p u_i))^2 = \frac{1}{n}(\mathbf{x} - \mathbf{tb})^T(\mathbf{x} - \mathbf{tb})$$

Taking the derivative and setting it to zero at $\widehat{\beta}$ gives

$$\frac{2}{n}\mathbf{t}^T(\mathbf{x} - \mathbf{t}\widehat{\beta}) = 0$$

Solving,

$$\widehat{\beta} = (\mathbf{t}^T\mathbf{t})^{-1}\mathbf{t}^T\mathbf{x}$$

The fitted valued come from

$$\widehat{\mu} = \mathbf{t}(\mathbf{t}^T\mathbf{t})^{-1}\mathbf{t}^T\mathbf{x}$$

so the weight matrix in this case is

$$\mathbf{w} = \mathbf{t}(\mathbf{t}^T\mathbf{t})^{-1}\mathbf{t}^T$$

You can check that this matrix is symmetric, and that it is idempotent. You can also check that when we just deal with one predictor, these matrix formulas are equivalent to the ones for simple linear regression above.

These formula handles *all* of the coefficients at once. You can check, by explicitly taking the partial derivative just with respect to b_0 and setting that to 0, that it is still the case that

$$\hat{\beta}_0 = \overline{y} - (\hat{\beta}_1\overline{t} + \hat{\beta}_2\overline{z} + \ldots + \hat{\beta}_p\overline{u})$$

This tells us that the individual weights in $\mathbf{w}$ still have the same form — they're a constant, plus terms that give more weight to points far from the center of the data.

7.3 Transformations of one predictor; polynomial regression

Nothing in the logic for the previous section requires that the extra predictors be new variables. If we want to do polynomial regression,

$$m(t) = \beta_0 + \beta_1 t + \beta_2 t^2 + \ldots \beta_p t^p$$

then everything carries through exactly as before. The matrix $\mathbf{t}$ would just have columns containing the constant, t_i, t_i^2 and so on.

What's important is that we're just taking an additive combination of the functions of t on the right-hand side of the model formula, and that all of the functions on the RHS are linearly independent of each other. (If there was linear dependence, then the matrix $\mathbf{t}^T\mathbf{t}$ wouldn't be invertible.)

8 Appendix: Some linear algebra reminders: eigen-stuff

This section can be skipped if you are comfortable working with the eigenvalues and eigenvectors of matrices. Even if you are, it won't hurt to at least skim it.

A (non-zero) vector or $n \times 1$ matrix $\mathbf{v}$ is an **eigenvector** of the $n \times n$ matrix $\mathbf{w}$, with **eigenvalue** λ, when

$$\mathbf{w}\mathbf{v} = \lambda\mathbf{v}$$

We can re-write this as

$$(\mathbf{w} - \lambda\mathbf{I})\mathbf{v} = 0$$

and demanding $\mathbf{v} \neq 0$ tells us that the λ which make this true have to obey the **characteristic polynomial** or **eigenpolynomial** of the matrix

$$\det(\mathbf{w} - \lambda\mathbf{I}) = 0$$

In R, the command to find the eigenvalues and eigenvectors of a matrix is `eigen`:

```
theta <- runif(n=2)
# Why "mark"?
mark <- matrix(c(theta[1], 1-theta[1], 1-theta[2], theta[2]),
               byrow=TRUE, nrow=2, ncol=2)
mark
```

```
##           [,1]      [,2]
## [1,] 0.8689482 0.1310518
## [2,] 0.2075234 0.7924766
```

```
eigen(mark)
```

```
## eigen() decomposition
## $values
## [1] 1.0000000 0.6614248
##
## $vectors
##           [,1]       [,2]
## [1,] 0.7071068 -0.5339475
## [2,] 0.7071068  0.8455176
```

This returns a list with two components, one of them a vector of eigenvalues, the other a matrix whose columns are the eigenvectors.

8.1 General properties of eigenvalues and eigenvectors

There are a number of salient properties of eigenvalues and eigenvectors.

1. There are at most n distinct eigenvalues of $\mathbf{w}$, which might be complex[17]. When there are fewer than n distinct eigenvalues, we nonetheless write the eigenvalues as $\lambda_1, \lambda_2, \ldots \lambda_n$, keeping track of how often each one is a root of the characteristic polynomial.
2. The determinant of a matrix is the product of its eigenvalues, "counted with multiplicity":

   $$\det \mathbf{w} = \prod_{j=1}^{n} \lambda_j$$

   The trace of a matrix is the sum of its eigenvalues:

   $$\mathrm{tr}\, \mathbf{w} = \sum_{j=1}^{n} \lambda_j$$

[17] $\det(\mathbf{w} - \lambda\mathbf{I}) = 0$ is called the characteristic *polynomial* because working out the determinant gives us an n^{th} order polynomial in λ. This has at most n distinct roots, some of which might be complex; the same λ can appear multiple times as a root.

3. For any matrix $\mathbf{w}$, and any power $k > 0$, $\mathbf{v}$ is an eigenvector of $\mathbf{w}$ with eigenvalue λ if and only if $\mathbf{v}$ is an eigenvector of $\mathbf{w}^k$ with eigenvalue λ^k. If $\mathbf{w}$ is invertible, then $\mathbf{v}$ is an eigenvector of $\mathbf{w}^{-1}$ with eigenvalue $1/\lambda$.

4. If $\mathbf{v}$ is an eigenvector with eigenvalue λ, then for any scalar $a \neq 0$, $a\mathbf{v}$ is also an eigenvector, with the same eigenvalue[18]. This means that the eigenvectors, unlike the eigenvalues, aren't uniquely defined. Instead, we usually *choose* eigenvectors to have nice properties, e.g., length 1. Also, $-\mathbf{v}$ is always another eigenvector, so two calculations of eigenvectors might end up with different signs.

5. If $\mathbf{v}$ and $\mathbf{u}$ are eigenvectors with the same eigenvalue λ, then so is $a\mathbf{v} + b\mathbf{u}$, for any scalars a, b. (Check this yourself!)

6. For fixed λ, the eigenvectors solve the equation $(\mathbf{w} - \lambda\mathbf{I})\mathbf{v} = 0$. This matrix equation is equivalent to a system of n linear equations in the n unknowns $(v_1, v_2, \ldots v_n)$. If there are no redundant (linearly dependent) equations in this system, then the solution is unique, up to the over-all scaling factor we just saw in (4). But if there are redundant equations, we can have a whole space of solutions, the **eigenspace** of λ. The number of redundant equations gives the dimension of this eigenspace, a.k.a. the **geometric multiplicity** of λ. When there are multiple, linearly-independent eigenvectors with eigenvalue λ, we can *choose* orthogonal eigenvectors to represent the whole eigenspace.

7. If $\mathbf{w}$ is symmetric, and $\mathbf{v}$ and $\mathbf{u}$ are eigenvectors with distinct eigenvalues $\lambda_1 \neq \lambda_2$, then $\mathbf{v}^T\mathbf{u} = 0$, and we say the vectors are orthogonal[19]. *Remark*: The orthogonality of eigenvectors actually holds for many asymmetric matrices as well, so long as they are **normal**, meaning $\mathbf{w}^T\mathbf{w} = \mathbf{w}\mathbf{w}^T$. The proof is similar, but more involved.[20].

8. What is *generally* true, of most matrices, is that the eigenvectors span the whole n-dimensional space. (There are "defective" matrices where this is not true.) In those cases, we can stack the eigenvectors column-wise into a matrix, say $\mathbf{e}$, and have $\mathbf{we} = \mathbf{e}\lambda$, where λ is the diagonal matrix of the eigenvalues. But, because the eigenvectors are (by assumption) linearly independent, $\mathbf{e}^{-1}$ exists, and we have the **eigendecomposition**
$$\mathbf{w} = \mathbf{e}\lambda\mathbf{e}^{-1}$$

8.2 Eigen-properties of some special types of matrices

8.2.1 Symmetric matrices

If $\mathbf{w} = \mathbf{w}^T$, then all the eigenvalues are real (i.e., none are complex). Because eigenvectors with different eigenvalues are distinct in this case, and we can always normalize all the eigenvectors, we have $\mathbf{e}^{-1} = \mathbf{e}^T$, and $\mathbf{w} = \mathbf{e}\lambda\mathbf{e}^T$.

8.2.2 "Stochastic" matrices

A matrix is called **stochastic** if $w_{ij} \geq 0$ for all i, j, and if $\sum_{j=1}^{n} w_{ij} = 1$ for all i. Some very important properties follow from this:

1. The largest eigenvalue is exactly 1.
2. The eigenvectors corresponding to the eigenvalue 1 have only non-negative entries.

[18] By the ordinary rules for matrix multiplication, $\mathbf{w}(a\mathbf{v}) = a\mathbf{w}\mathbf{v} = a\lambda\mathbf{v}$, because $\mathbf{v}$ is an eigenvector.

[19] Since $\mathbf{w}$ is symmetric, $\mathbf{w} = \mathbf{w}^T$, we have $\mathbf{u}^T\mathbf{w}\mathbf{v} = \mathbf{v}^T\mathbf{w}\mathbf{u}$. Because these are eigenvectors, this means

$$\mathbf{u}^T\lambda_1\mathbf{v} = \mathbf{v}^T\lambda_2\mathbf{u} \quad (70)$$
$$\lambda_1\mathbf{u}^T\mathbf{v} = \lambda_2\mathbf{v}^T\mathbf{u} \quad (71)$$

but since $\mathbf{v}^T\mathbf{u} = \mathbf{u}^T\mathbf{v}$ (it's a 1×1 matrix, which is always symmetric), we get

$$(\lambda_1 - \lambda_2)(\mathbf{u}^T\mathbf{v}) = 0$$

and we assumed $\lambda_1 \neq \lambda_2$.

[20] Here's a counter-example. The matrix $\begin{bmatrix} 2 & 0 \\ 2 & 3 \end{bmatrix}$, has eigenvalues are 3 and 2, and the corresponding eigenvectors are $[0, 1]$ and $[0.4472136, -0.8944272]$, which aren't orthogonal.

3. All other eigenvalues have magnitude ≤ 1.

These are all consequences of a result in linear algebra called the Frobenius-Perron theorem; you can take it on trust, or read Axler (1996).

8.2.3 Projection / idempotent matrices

A matrix is called **idempotent**[21] when $\mathbf{w}^2 = \mathbf{ww} = \mathbf{w}$. It follows[22] that the only possible eigenvalues are 0 and 1.

Projecting on to a linear subspace (line, plane, hyper-plane, etc.) always corresponds to an idempotent matrix. (Points already in the space aren't affected by the projection, so projecting twice is the same as projecting once.) The dimension of the space corresponds to the number of eigenvalues equal to 1.

[21] From Latin *idem*, "the same", and *potens*, "power", hence something which is the same as its powers.

[22] If $\mathbf{v}$ is an eigenvector of $\mathbf{w}$ with eigenvalue λ, then clearly $\mathbf{w}^2\mathbf{v} = \mathbf{wwv} = \mathbf{w}\lambda\mathbf{v} = \lambda\mathbf{wv} = \lambda^2\mathbf{v}$, so λ^2 is an eigenvalue of $\mathbf{w}^2$. But, since we're assuming $\mathbf{w}$ is idempotent, $\mathbf{w}^2 = \mathbf{w}$, so all eigenvalues must obey the equation $\lambda^2 = \lambda$, and the only two solutions of that are $\lambda = 0$ and $\lambda = 1$.

9 Appendix: Some numerical-methods stuff; Newton's method

_This section is optional, but explains how to use Newton's method to optimize functions, or to solve a nonlinear equation or system of equations.

9.1 Newton's method for solving an equation with one unknown

Say we want to solve the equation $g(x) = 0$ for one scalar unknown x. (Solving the equation $g(x) = c$ is the same as solving $g(x) - c = 0$, so we can just re-define the function we care about.) **Newton's method** is an iterative way of doing this, which starts with an initial guess, call it x_0, and then refines it successively. The key step in Newton's method is to approximate the function g as a *linear* function, by taking a Taylor expansion around x_0:

$$g(x) \approx g(x_0) + (x - x_0)g'(x_0)$$

where g' is the first derivative function. Now this is easy to solve for x:

$$g(x_0) + (x - x_0)g'(x_0) = 0 \tag{72}$$

$$x - x_0 = -\frac{g(x_0)}{g'(x_0)} \tag{73}$$

$$x = x_0 - \frac{g(x_0)}{g'(x_0)} \tag{74}$$

Notice that if our initial guess x_0 happened to be a solution, because $g(x_0) = 0$, then we get it back again. Otherwise, we call this x_1, and do it again. In general,

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}$$

Notice that if $g(x_k) > 0$, whether x_{k+1} is bigger or smaller than x_k depends on the sign of $g'(x_k)$. If $g'(x_k) > 0$, then g increases as we move to the right of x_k, so we should move to the left to make it smaller. All of the other permutations for signs of $g(x_k)$ and $g'(x_k)$ work similarly — Newton's method always adjusts us to move close to $g(x) = 0$.

We keep going through these iterations until we hit on an exact solution, or the difference between $g(x_k)$ and $g(x_{k+1})$ becomes so small we don't care any more, or we get tired.

9.1.1 Example: Extracting square roots

Say we want to find $\sqrt{c}$. This is clearly the same as finding the x which solves $x^2 = c$. That in turn is the same as finding the x which solves $x^2 - c = 0$, so say $g(x) = x^2 - c$. Thus $g'(x) = 2x$. Starting from some guess x_0, the iteration is

$$x_{k+1} = x_k + \frac{c - x_k^2}{2x_k}$$

Or: see how far off the current guess is, divide by the current guess, and add that to the current guess. (If the guess is too small, $c > x_k^2$, make it bigger; if the guess is too big, $c < x_k^2$, make it smaller.) Even more specifically, re-write this as

$$x_{k+1} = \frac{2x_k^2 + (c - x_k^2)}{2x_k} = \frac{c + x_k^2}{x_k} = \frac{x_k + c/x_k}{2}$$

which is the average of x_k and c/x_k. You can convince yourself that if $x_k > \sqrt{c}$ then $c/x_k < \sqrt{c}$ and vice versa, so this procedure involves averaging an over-estimate and an under-estimate, and should have a certain intuitive plausibility to it.

This is the oldest instance I know of what we now call Newton's method; this way of finding square roots is often called "Hero" or "Heron's" method (or "algorithm"), because the oldest surviving text which lays it

out explicitly was written by the Hellenistic[23] mathematician Heron of Alexandria around +60. There are reasons to think it is much older and was known in Babylonia, but this isn't certain (Robson 2007, especially pp. 102ff).

9.2 Optimizing a function of one unknown

Suppose we want to minimize a function $g(x)$ of one unknown x. We remember from calculus that, at a minimum, we generally have $g'(x) = 0$. We agree to ignore the exceptions. We therefore want to solve the equation $g'(x) = 0$. But we've just seen how to solve an equation with Newton's method. So for optimization, the iteration is

$$x_{k+1} = x_k - \frac{g'(x_k)}{g''(x_k)}$$

so we need to take a second derivative.

(Another way to think of this is to imagine approximating $g(x)$ not by a linear function but a quadratic, because quadratics are easy to optimize, just like linear equations are easy to solve.)

Depending on the choice of the starting guess x_0, this might converge to only a local rather than a global minimum; it might even converge on a local maximum. (But you can check that by looking at the sign of g''.)

9.3 Optimizing a function of many unknowns

Suppose now that x is really a vector of dimension d, say $x = [x(1)\ x(2)\ \ldots x(d)]$. We still want to minimize $g(x)$. The condition for a minimum has become $\nabla g(x) = \mathbf{0}$, which is to say that

$$\begin{bmatrix} \frac{\partial g}{\partial x^{(1)}} \\ \frac{\partial g}{\partial x^{(2)}} \\ \vdots \\ \frac{\partial g}{\partial x^{(d)}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

We can do a Taylor expansion here, too:

$$\nabla g(x) \approx \nabla g(x_0) + (x - x_0)\nabla\nabla g(x_0)$$

Here $\nabla\nabla g(x_0)$ is the $d \times d$ matrix of second partial derivatives of g at x_0, so the i, j entry is $\frac{\partial^2 g}{\partial x^{(i)} \partial x^{(j)}}$, which is also called the **Hessian**. Setting this to zero and solving gives

$$x_{k+1} = x_k - \left(\nabla\nabla g(x_k)\right)^{-1} \nabla g(x_0)$$

Fortunately for us, R has a very efficient version of Newton's method built in, in the `optim()` function with `method="BFGS"`[24].

[23] In addition to his mathematical work, Heron also designed, and apparently built, many different mechanical devices, including, most astonishingly, a kind of steam engine. He is often described as a "Greek mathematician", but this is a bit anachronistic. We know he was from Alexandria in Egypt, but that is compatible with any number of ethnic ancestries, including, of course, Egyptian. We also know that he wrote in Greek, but at the time Greek was the main language of scholarship over much of the world, from the Atlantic Ocean to the Hindu Kush mountains; calling every ancient scientist who wrote in Greek a "Greek scientist" would be a bit as though our descendants in the year 4000 called every contemporary scientist writing in English today an "English scientist". (Calling him "Roman" because he lived in the Roman Empire would be as though people in the year 4000 called every contemporary scientist in a rich democracy "American".)

[24] While the basic Newton's method really is as simple as I sketched above, it can run in to some complications which "industrial-strength" optimization code tries to handle. (For instance, what do we do if the Hessian isn't invertible?) One of the most important wrinkles is that Newton's method calls for matrix inversion. This can get time-consuming when d is large! Finding $\nabla g(x_0)$ requires taking d partial derivatives, which requires $O(d)$ operations on the computer. This is small compared to finding the Hessian, which requires finding $O(d^2)$ partial derivatives. But even that is small compared to inverting the Hessian, which done straightforwardly would take $O(d^3)$ operations. (There are more efficient algorithms which can get this down to $O(d^{2.37\cdots})$ but it's still slower than $O(d^2)$.) Many algorithmic implementations of Newton's method get complicated because they try to avoid having to re-compute the inverse Hessian from scratch at each step. The Broyden, Fletcher, Goldfarb and Shanno [BFGS] method is a particular way of approximating the inverse Hessian using only gradient vectors. Most users don't get this far in to the details; those who do are usually happy to trade some inaccuracy in finding the optimum for much faster running time.

References

Axler, Sheldon. 1996. *Linear Algebra Done Right*. Berlin: Springer-Verlag.

Bulliet, Richard W. 1979. *Conversion to Islam in the Medieval Period: An Essay in Quantitative History*. Cambridge, Massachusetts: Harvard University Press. https://doi.org/10.4159/harvard.9780674732810.

Burman, Prabir, Edmond Chow, and Deborah Nolan. 1994. "A Cross-Validatory Method for Dependent Data." *Biometrika* 81:351–58. https://doi.org/10.1093/biomet/81.2.351.

Farebrother, Richard William. 1999. *Fitting Linear Relationships: A History of the Calculus of Observations 1750–1900*. New York: Springer-Verlag. https://doi.org/10.1007/978-1-4612-0545-6.

Hacking, Ian. 1990. *The Taming of Chance*. Cambridge, England: Cambridge University Press. https://doi.org/10.1017/CBO9780511819766.

Hodrick, Robert J., and Edward C. Prescott. 1997. "Postwar U.S. Business Cycles: An Empirical Investigation." *Journal of Money, Credit, and Banking* 29:1–16. https://doi.org/10.2307/2953682.

Kafadar, Karen. 1996. "Smoothing Geographical Data, Particularly Rates of Disease." *Statistics in Medicine* 15:2539–60. https://doi.org/10.1002/(SICI)1097-0258(19961215)15:23<2539::AID-SIM379>3.0.CO;2-B.

Klein, Judy L. 1997. *Statistical Visions in Time: A History of Time Series Analysis, 1662–1938*. Cambridge, England: Cambridge University Press.

Kolmogorov, A. N., and S. V. Fomin. 1975. *Introductory Real Analysis*. New York: Dover.

Marx, Karl. 1936. *The Process of Capitalist Production*. Vol. 1. Capital: A Critique of Political Economy. New York: The Modern Library.

Paige, Robert L., and A. Alexandre Trindade. 2010. "The Hodrick-Prescott Filter: A Special Case of Penalized Spline Smoothing." *Electronic Journal of Statistics* 4:856–74. https://doi.org/10.1214/10-EJS570.

Porter, Theodore M. 1986. *The Rise of Statistical Thinking, 1820–1900*. Princeton, New Jersey: Princeton University Press.

Racine, Jeff. 2000. "Consistent Cross-Validatory Model-Selection for Dependent Data: *hv*-Block Cross-Validation." *Journal of Econometrics* 99:39–61. https://doi.org/10.1016/S0304-4076(00)00030-0.

Robson, Eleanor. 2007. "Mesopotamian Mathematics." In, edited by Victor J. Katz, 85–186. Princeton, New Jersey: Princeton University Press.

Rogers, Everett M. 2003. *Diffusion of Innovations*. Fifth. New York: Free Press.

Shalizi, Cosma Rohilla. n.d. *Advanced Data Analysis from an Elementary Point of View*. Cambridge, England: Cambridge University Press. http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV.

Stigler, Stephen M. 1986. *The History of Statistics: The Measurement of Uncertainty Before 1900*. Cambridge, Massachusetts: Harvard University Press.

Wood, Simon N. 2004. "Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models." *Journal of the American Statistical Association* 99:673–86. http://www.maths.bath.ac.uk/~sw283/simon/papers/magic.pdf.

———. 2006. *Generalized Additive Models: An Introduction with R*. Boca Raton, Florida: Chapman; Hall/CRC.