Lecture 4: Regularization and Smoothing

Introduction to Time Series, Fall 2024

Ryan Tibshirani

Relevant reading: Chapter 2.3 (smoothing) of Shumway and Stoffer (SS); Chapters 3.3 and 7.7 (smoothing) of Hyndman and Athanasopoulos (HA).

1 Trouble in high dimensions?

- As in the regression lecture, let's suppose that we seek $\beta \in \mathbb{R}^p$ such that for given samples $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (predictor and response pairs), $i = 1, \ldots, n$,

$$y_i \approx x_i^\mathsf{T} \beta, \quad i = 1, \ldots, n$$

  (As explained in that lecture, our notation omits the intercept from the model, but that is done without a loss of generality, since it can always be obtained by appending a coordinate value of 1 to the start of each x_i)

- Equivalently, we can write $y \in \mathbb{R}^n$ for the response vector (with i^{th} component y_i) and $X \in \mathbb{R}^{n \times p}$ for the feature matrix (with i^{th} row x_i), and say that we are seeking β such that $y \approx X\beta$

- Recall, the least squares estimates of the coefficients are given by solving

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n (y_i - x_i^\mathsf{T} \beta)^2 \iff \min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \tag{1}$$

  where we use $\|\cdot\|_2$ for the Euclidean or ℓ_2 norm of a vector, defined for $a \in \mathbb{R}^d$ as $\|a\|_2^2 = \sum_{i=1}^d a_i^2$

- If $p \leq n$ and $\text{rank}(X) = p$ (here $\text{rank}(X)$ denotes the rank of the matrix X), then this produces the unique solution

$$\hat{\beta} = (X^\mathsf{T} X)^{-1} X^\mathsf{T} y \tag{2}$$

- But if $p > n$, which means we have more features than samples, which we often call the "high dimensional" (or "overparametrized") setting, then we are in trouble ... the matrix $X^\mathsf{T} X$ cannot be invertible, so the expression in (2) isn't even well-defined

- Moreover, the least squares optimization problem (1) does not have a unique solution in this case. Indeed, are you'll show on the homework, it $\tilde{\beta}$ is one solution, then any other vector of the form

$$\hat{\beta} = \tilde{\beta} + \eta, \quad \text{where } \eta \in \text{null}(X) \tag{3}$$

  also solves (1), where $\text{null}(X)$ is the null space of the matrix X:

$$\text{null}(X) = \{\eta \in \mathbb{R}^p : X\eta = 0\}$$

- When $\text{rank}(X) < p$, the null space $\text{null}(X)$ is nontrivial, and since it is a linear space: $\eta \in \text{null}(X) \implies c\eta \in \text{null}(X)$ for any $c \in \mathbb{R}$, we see that from one least squares solution $\tilde{\beta}$, we can generate *infinitely many others* in (3)

- Furthermore, we can always take the "one least squares solution" to be:

$$\tilde{\beta} = (X^\mathsf{T} X)^+ X^\mathsf{T} y \tag{4}$$

- Here A^+ denotes the *generalized inverse* (also called the *Moore-Penrose pseudoinverse*) of a matrix A. If you don't know what that is, then it doesn't really matter for this lecture, but you can think of it precisely as follows: among all solutions in (1), the solution in (4) is the unique one having the smallest ℓ_2 norm $\|\tilde{\beta}\|_2$
- So, now we come to the discussion of specific *troubles*. There are actually two distinct troubles. The first trouble involves the interpretation of the coefficients themselves. If we are interested in such interpretations, then the $p = n$ barrier is the end of the road for least squares. Why? Once $p > n$, and we find any least squares solution $\tilde{\beta}$ with $\tilde{\beta}_j > 0$ for some j, then we can always find[1] another solution $\hat{\beta}$ of the form (3) with $\hat{\beta}_j < 0$. You will prove this on the homework. Thus we cannot even consistently interpret the sign of any of any estimated coefficient (let alone its magnitude)
- The second trouble involves prediction. The $p = n$ barrier is generally disastrous for least squares prediction. If $p < n$ and $\hat{y}_{\text{new}} = x_{\text{new}}^{\mathsf{T}}\hat{\beta}$ is the least squares prediction at a new predictor value x_{new} (for $\hat{\beta}$ the usual least squares coefficients in (2)), whose associated response is y_{new}, then under fairly standard conditions for regression theory, the prediction MSE behaves as:

$$\mathbb{E}\big[(y_{\text{new}} - \hat{y}_{\text{new}})^2\big] \approx \sigma^2 \frac{p}{n-p}$$

  for large n and p, where σ^2 is the error variance. What do we notice? This explodes as p approaches n. Big problem!
- This is being driven by the *variance* of the predictions from least squares, which grows out of control as p approaches n
- (An interesting side note: what happens with the prediction MSE when $p > n$? The answer may surprise you. The MSE associated with (4) is actually quite interesting and in some ways exotic when $p > n$. Typically we need p to be *much larger* than n (away from the $p = n$ barrier) in order for it to be well-behaved. This has been the topic of a recent flurry of research in statistics in machine learning ... we won't cover this, but feel free to ask about it in office hours)

2 Regularization

- *Regularization* to the rescue! This will finesse both of the problems described above: it leads to nontrivial coefficient estimates, and often leads to more accurate predictions, by reducing variance
- In the regression setting, a general approach for regularization moves us from (1) to solving:

$$\min_{\beta} \|y - X\beta\|_2^2 + \lambda P(\beta) \tag{5}$$

  Here $P : \mathbb{R}^p \to \mathbb{R}_+$ is a penalty function, and $\lambda \geq 0$ is a tuning parameter (or regularization parameter), trading off the importance of the squared loss and penalty (first and second terms in (5))
- In other words, the larger the value of λ, the more weight we put on penalizing large values of $P(\beta)$, which results in estimates that we call more "regularized"
- Arguably the three canonical choices for penalties are based on the ℓ_0, ℓ_1, and ℓ_2 norms:

$$P(\beta) = \|\beta\|_0 = \sum_{j=1}^{p} 1\{\beta_j \neq 0\},$$

$$P(\beta) = \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|,$$

$$P(\beta) = \|\beta\|_2^2 = \sum_{j=1}^{p} \beta_j^2.$$

[1] Technically, this is only true if $\text{null}(X) \not\perp e_j$, where e_j is the j^{th} standard basis vector.

- These give rise to what we call *best subset selection*, the *lasso*, and *ridge regression*, respectively
- (Note that calling $\|\cdot\|_0$ the "ℓ_0 norm" is a misnomer, as it is not actually a norm: it does not satisfy positive homogeneity, i.e., $\|a\beta\|_0 = a\|\beta\|_0$ for all $a > 0$. It would be more accurate to call it the "ℓ_0 pseudonorm", but nearly everybody just calls it the "ℓ_0 norm")
- Critically, $\|\cdot\|_0$ is *not convex*, while $\|\cdot\|_1$ and $\|\cdot\|_2$ are convex (note that any norm is a convex function). This makes best subset selection a nonconvex problem, and one that is generally hard to solve in practice except for small p. Meanwhile, the lasso and ridge regression are both defined by convex optimization problems, and are generally much easier to compute. We won't focus on best subset selection further in this lecture. (Though it was the topic of a flurry of work in the operations research literature a few years ago ... which you can ask about in office hours if you are curious)
- Lastly, an important note: when there is an explicit intercept β_0 in the model, we typically do not want to penalize it, *and so we modify the penalty $P(\beta)$ so that it excludes β_0*. This will be implicit in everything below
- Ok, really lastly, penalties like ℓ_1 and ℓ_2 only make sense if all the features (columns of X) are on the same scale (why?). Thus an important and standard preprocessing step is to scale each feature (column of X) so that it has unit norm

2.1 Ridge

- The *ridge* estimates of the regression coefficients are given by solving

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n (y_i - x_i^\mathsf{T}\beta)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

or equivalently

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \tag{6}$$

- The problem in (6) has closed-form solution, verified by differentiating the criterion and setting the result equal to zero,

$$\hat{\beta} = (X^\mathsf{T}X + \lambda I)^{-1} X^\mathsf{T} y \tag{7}$$

where I is in the $p \times p$ identity matrix. This *always exists* (the matrix $X^\mathsf{T}X + \lambda I$ is always invertible), regardless of the relative sizes of n, p

- Figure 1 gives an example of the ridge regression coefficient estimates as a function of λ, fit on the cardiovascular mortality regression data, with many features derived from lags of particulate levels

2.2 Lasso

- The *lasso* estimates of the regression coefficients are given by solving

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^n (y_i - x_i^\mathsf{T}\beta)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

or equivalently

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \tag{8}$$

- There are a number of key differences between the ridge (6) and (8) problems (and estimator). First, the lasso problem does not have a closed-form solution (it does not even necessarily have a unique solution; but it is essentially always unique if we have random features from a continuous distribution)
- A second key difference is that the lasso estimates of the regression coefficients are *sparse*. In other words, solving the lasso problem results in a vector $\hat{\beta}$ with many components exactly equal to zero, and a larger choice of λ will result in more zeros

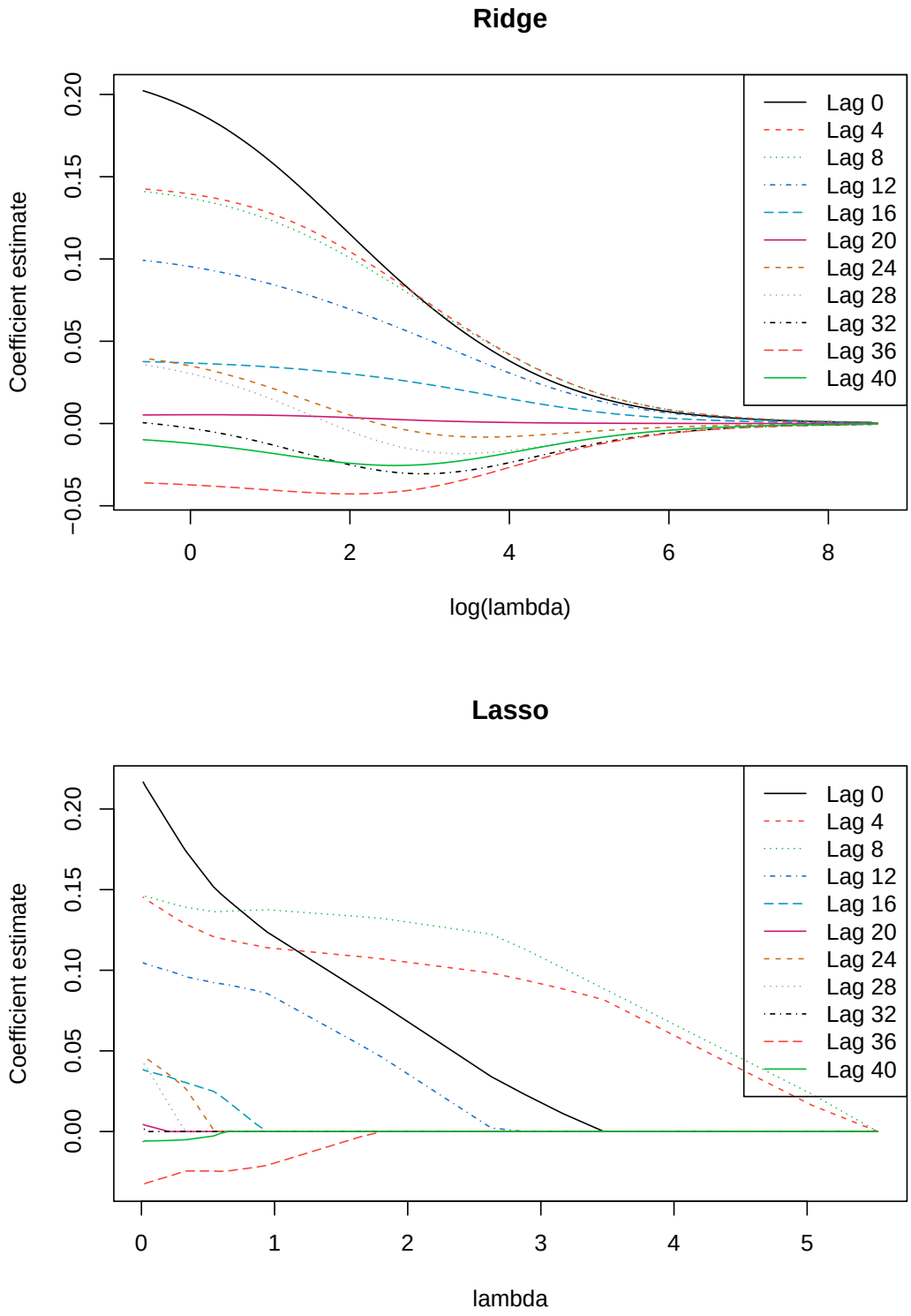

Figure 1: Ridge and lasso estimates on the cardiovascular mortality regression problem, with lags of particulate levels as features. (The intercept is estimated without a penalty, and omitted from the plots for visualization purposes).

- This doesn't happen with ridge regression, whose coefficient estimates are generically *dense*. Figure 2 is the "classic" picture used to explain this, and we will talk through its interpretation in lecture

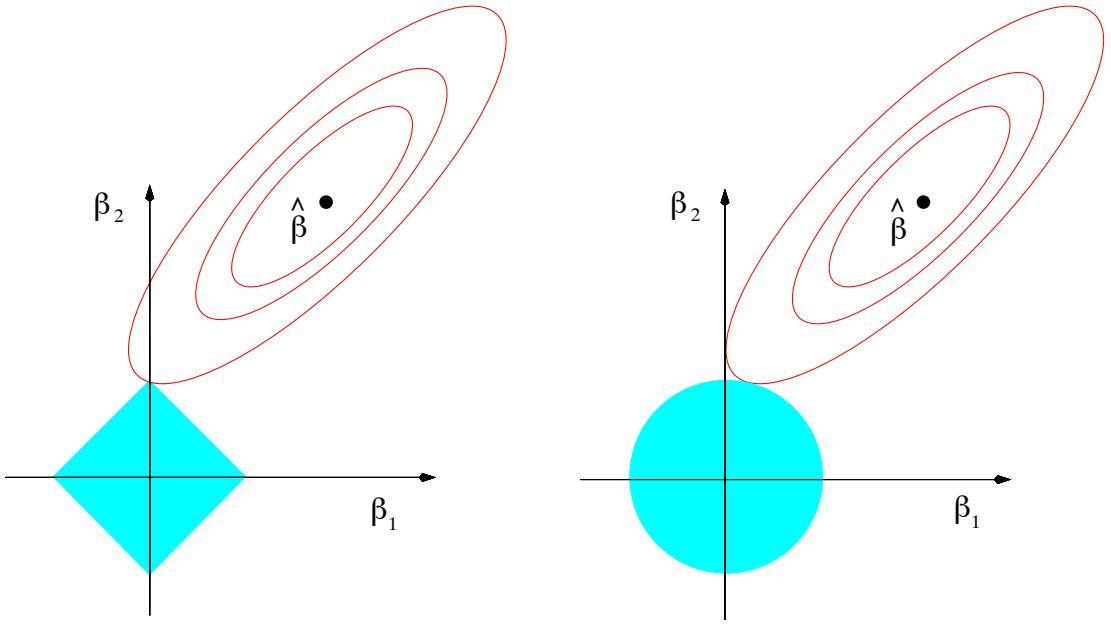

Figure 2: The "classic" illustration comparing lasso and ridge, each in constrained form (from "Elements of Statistical Learning" by Hastie, Tibshirani, and Friedman).

- See Figure 1 again for an example of the lasso coefficients on the cardiovascular mortality data. We can see that as λ grows, it sets many of the coefficients of lagged features to zero
- In general, this sparsity-inducing behavior of the ℓ_1 penalty allows the lasso to perform *variable selection* in the working linear model. By zero-ing out some coefficients exactly, it discards some features from having any predictive influence in the fitted model. Which precise features it discards is chosen based on the data. Many people like sparsity because it leads to better interpretability

2.3 Discussion: which one?

- We should be clear that the lasso is not "better" than ridge in any general sense, and neither is ridge "better" than the lasso. They each can help tremendously with stabilizing coefficient estimates so as to lead to improved predictive accuracy. They each do so by regularizing in different ways
- The most basic question to ask: is a sparse linear model likely to be a good (or desirable) approximation to the true regression function? If so, then the lasso can outperform (or be preferable) to ridge. On the other hand, in problems where there are many underlying features that are relevant for prediction, ridge can outperform the lasso
- (And often times people combine the two penalties which gives rise to the *elastic net*)
- There is a lot more to say—in terms of connections to other ideas in statistics, extensions, and so on—but we won't be able to cover it in this class. We'll simply view ridge and lasso as tools that allow us to consider many more features than we would otherwise feel comfortable including in traditional regression models, and then regularize in order to control variance (stabilize estimates)
- In time series regression, in the vein of examples we studied in the last lecture, this would allow us to include *many lags* of a feature of interest, or a few lags of *many external covariates*, and so on, and then apply a ridge or lasso penalty

- Then, you might wonder: how would we select λ? In fact, you already know the answer (for problems with a predictive focus): use time series cross-validation!
- That is, define a grid of λ values, fit ridge or lasso estimates for each λ, let each one make predictions, and select the value that yields the best CV error. You will practice this on the homework, where you'll also use the `glmnet` package to solve the ridge and lasso problems

2.4 Interlude: coordinate descent for the lasso

- As a very quick interlude, here is an algorithm used to compute lasso estimates in practice, called *coordinate descent*. In fact, this is what is used by the `glmnet` package internally. We repeat, cycling over $j = 1, 2, \ldots, p$ until convergence (until the coefficient estimates do not really change anymore), the following steps (with x_j denoting the j^{th} column of X):
  - Compute the j^{th} partial residual: $r_j = y - \sum_{\ell \neq j} x_\ell \hat{\beta}_\ell$
  - Compute the univariate least squares coefficient of r_j on x_j:

$$\check{\beta}_j = \frac{x_j^T r_j}{x_j^T x_j}$$

  - Update the j^{th} lasso coefficient by "soft-thresholding" the above:

$$\hat{\beta}_j = \begin{cases} \check{\beta}_j + 2\lambda & \text{if } \check{\beta}_j < -2\lambda \|x_j\|_2^2 \\ \check{\beta}_j - 2\lambda & \text{if } \check{\beta}_j > +2\lambda \|x_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

    where recall λ is the lasso regularization parameter
- (Note that similarity here between this algorithm and the connection between marginal and multiple regression that we covered previously. In fact, this algorithm is still valid when $\lambda = 0$, and it can be used to compute least squares coefficients!)

3 Smoothing

- Shifting gears, we will now talk about *smoothing*, which for us will be a tool that is mainly used for retrospective rather than prospective estimation (as in forecasting)
- As with regression, most smoothing techniques are not actually specific to time series data, so our notation will be generic to reflect this. There are so many smoothers out there, and we will choose three ones to discuss that are generic in principle, but are quite popular (and in part, have roots in) the time series context
- Recall the signal plus noise model from our earlier lectures,

$$y_i = \theta_i + \epsilon_i, \quad i = 1, \ldots, n$$

  where ϵ_i, $i = 1, \ldots, n$ are white noise errors, and θ_i, $i = 1, \ldots, n$ denote the unknown means that we want to estimate, assumed to be smoothly varying across the index i
- Two broad classes of smoothers of interest are as follow. The first are *linear filters*, which are (weighted) local averages, of the form

$$\hat{\theta}_i = \sum_{j=-k}^{k} a_j y_{i-j}, \quad i = 1, \ldots, n \tag{9}$$

  for some weights a_j, $j = -k, \ldots, k$

- The second are *penalized least squares* smoothers, which are given by solving

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^n (y_i - \theta_i)^2 + \lambda P(\theta) \tag{10}$$

  for some penalty function $P : \mathbb{R}^n \to \mathbb{R}_+$ and tuning parameter $\lambda \geq 0$

- You should be able to clearly see the connection between smoothing and regularization, with (10) being a special case of (5) (when $X = I$, the identity matrix). Note, the choice of penalties that are common and useful in smoothing, as we'll see below, are more complex than simple ℓ_1 or ℓ_2 penalties

- There is a broader connection between smoothing and regression, embodied by what is called an *additive model*. In one sentence: this is a regression, but with individual features replaced by nonlinear transforms of features, where the nonlinearities are fit by smoothers. We'll very briefly return to this at the end

3.1 Linear filters

- A *moving average* (MA), either of centered or trailing type, is one of the basic and widely-used examples of a linear filter, of the form (9). For a window of (odd) length $m = 2k + 1$, a centered MA smoother uses weights

$$a_{-k} = \cdots a_0 = \cdots = a_k = \frac{1}{m}$$

  For a window of (odd or even) length $m = k + 1$, a trailing MA smoother uses weights

$$a_{-k} = \cdots = a_{-1} = 0 \quad \text{and} \quad a_0 = \cdots = a_k = \frac{1}{m}$$

- (For either, centered or trailing MA smoothers, or really linear filters in general, there will be annoying boundary issues to pay attention to ... some software packages may just pad the response vector with 0s in order to deal with them; but a more principled approach is probably to renormalize the weights at the boundaries so that the filter is averaging over the "right" number of observations; and the safest approach is to just omit estimates at the boundaries, i.e., report NAs)

- Figure 3 displays an example of centered MA smoothing on the Southern Oscillation Index (SOI), which measures air pressure in the Central Pacific Ocean. We can see that the results look fairly accurate (they track the known cycles which occur every 3-7 years, due to the El Nino effect), but the estimates look a bit "choppy"

- We can get smoother-looking estimates by using a smoother weight sequence (with larger k). This is precisely what *kernel smoothing* does: it uses $k = n$ and

$$a_j = \frac{K(j/b)}{\sum_{i=-n}^n K(i/b)}, \quad j = 1, \ldots, n$$

  for a particular *kernel function* $K : \mathbb{R} \to \mathbb{R}$ and *bandwidth* b. Think of the bandwidth as a tuning parameter, just like the regularization level λ in ridge or lasso

- A common choice of kernel is the Gaussian kernel, $K(u) = e^{-u^2/2}$. Figure 3 gives an example of Gaussian kernel smoothing on the SOI data again. We can see that the estimates look more smooth

- (Note that a centered MA is actually kernel smoothing with a "boxcar" kernel: $K(u) = 1\{|u| \leq b\}$)

3.2 Hodrick-Prescott filter

- The *Hodrick-Prescott filter*, or simply HP filter, is a penalized estimator of the form (10), defined by

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^n (y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n-2} (\theta_i - 2\theta_{i+1} + \theta_{i+2})^2 \tag{11}$$

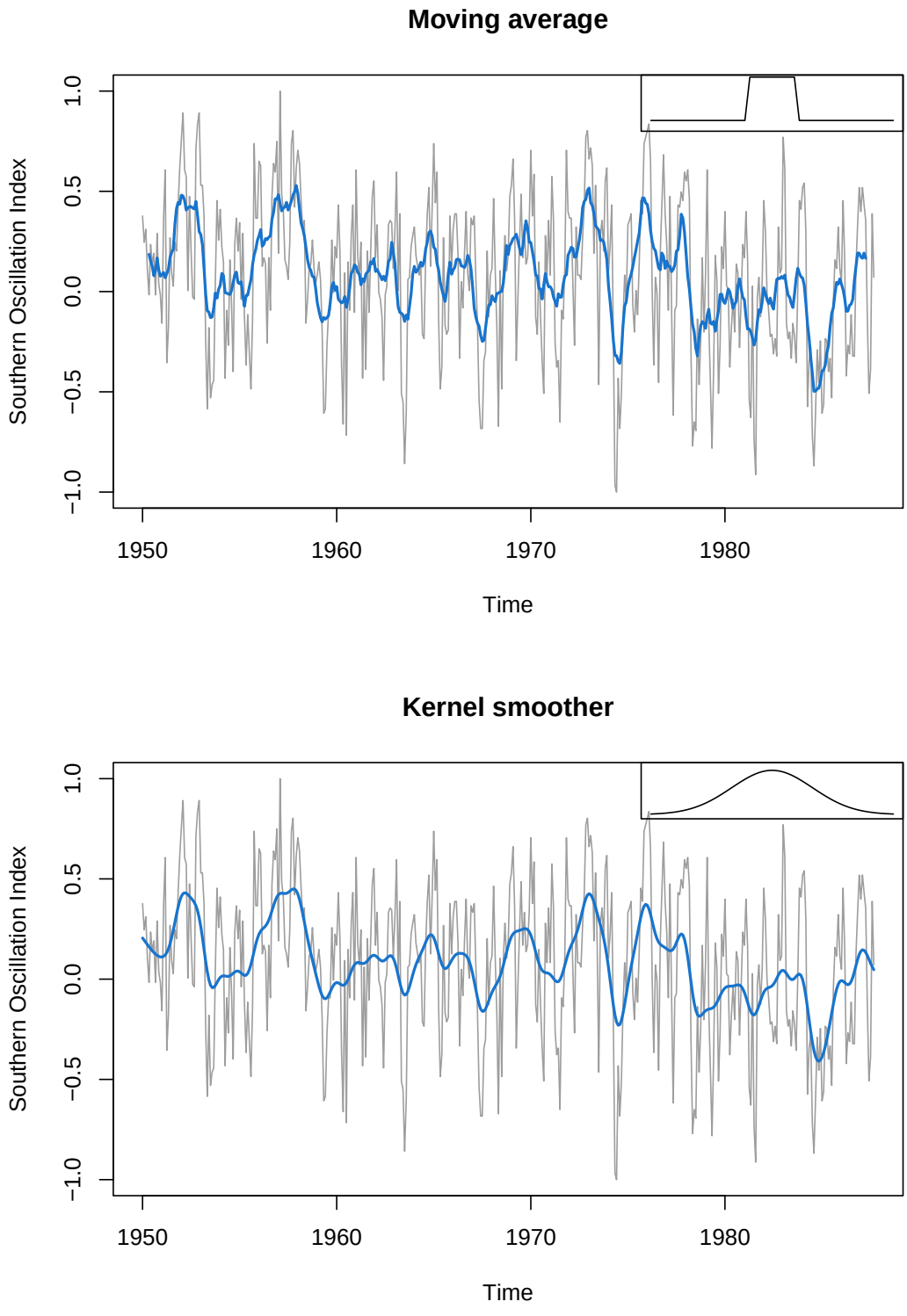

Figure 3: Moving average and Gaussian kernel smoothing estimates fit to the Southern Oscillation Index (from SS).

- We can see that the penalty is squared ℓ_2 norm of the terms $\theta_i - 2\theta_{i+1} + \theta_{i+2}$, $i = 1, \ldots, n-2$. Each of these are a *second difference* of θ, centered at a particular index
- The HP filter is named after two econometricians who proposed the idea in the early 1980s. But it is worth noting that very similar ideas were around much, much early (the idea of penalizing according to squared first differences dates back to the 1899, and according to squared third differences dates back to 1923)
- Like the ridge problem, which is similar in spirit (but aimed at solving a different problem), the HP filter problem (11) has an exact solution. First define the second difference matrix $D \in \mathbb{R}^{(n-2)\times n}$ by

$$D = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & & & \cdot & & & & & \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & -2 & 1 \end{bmatrix} \tag{12}$$

  and then rewrite the HP filter optimization problem (11) as

$$\min_{\theta \in \mathbb{R}^n} \|y - \theta\|_2^2 + \lambda \|D\theta\|_2^2 \tag{13}$$

- The HP filter solution can be derived by differentiating the criterion in (13), setting it equal to zero, and solving, which yields

$$\hat{\theta} = (I + \lambda D^\mathsf{T} D)^{-1} y \tag{14}$$

- Figure 4 shows an example of the HP filter fit to the winning men's times from the Boston marathon, at a few different values of the regularization level λ. (What happens to the HP filter as $\lambda \to \infty$?)
- The HP filter has a computational advantage a linear filter with smooth weights, such as the Gaussian kernel smoother (the HP filter can be computed in linear-time because D is banded). In terms of their qualitative behavior, the HP filter and a Gaussian kernel smoother are pretty similar
- In fact, though it not obvious at all, the HP filter admits an *equivalent* form as a kernel smoother, for a special implicit kernel (you will study this as a bonus problem on the homework)
- The HP filter is worth knowing about because it is quite a popular tool in macroeconomics, where it is primarily used for time series decomposition: it is used to estimate a trend component in a time series, and then the residuals from this are used to estimate a cyclic component. However, recently, some econometricians have criticized this practice, on the basis that it can induce spurious correlations in the residuals, along with other concerns.[2]
- Another reason worth learning about the HP filter is that it provides us a natural bridge to another method that we'll cover next, which acts completely differently from any linear filter

3.3 Trend filter

- The ℓ_1 *trend filter*, or simply trend filter, is another penalized estimator of the form (10), defined by

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^n (y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n-2} |\theta_i - 2\theta_{i+1} + \theta_{i+2}| \tag{15}$$

- We can see that the penalty is the ℓ_1 norm of second differences of θ. Thus, compared do the HP filter, the only difference is in the use of an ℓ_1 norm penalty, and not squared ℓ_2 norm penalty

[2] See Hamilton "Why you should never use the Hodrick-Prescott filter" (2017), but also Hodrick "An exploration of trend-cycle decomposition methodologies in simulated data" (2020), for a rebuttal which challenges some of the ideas proposed by Hamilton. Hodrick finds empirically that the HP filter does a better job of isolating the cyclic component (in simulation) than the regression method proposed by Hamilton.

- Just as in (13), we can rewrite the trend filter (15) more compactly as

$$\min_{\theta \in \mathbb{R}^n} \|y - \theta\|_2^2 + \lambda \|D\theta\|_1 \tag{16}$$

  where $D \in \mathbb{R}^{(n-2)\times n}$ is the second difference matrix in (12)

- Swapping the squared ℓ_2 penalty with an ℓ_1 penalty on second differences results in a *very* different behavior in the estimator. Think of the analogy of ridge versus the lasso. Trend filtering yields estimates that are *sparse* in second differences: if we denote by $\hat{\theta}$ the solution vector in (16), then many components of $D\hat{\theta}$ will be exactly equal to zero, and a larger choice of λ will generally result in more zeros

- Since

$$\hat{\theta}_i - 2\hat{\theta}_{i+1} + \hat{\theta}_{i+2} = 0 \iff \hat{\theta}_{i+1} = \frac{\hat{\theta}_i + \hat{\theta}_{i+2}}{2}$$

  this means that many components $\hat{\theta}_i$ will lie on the line defined by their neighbors

- Or in other words, the trend filter solution $\hat{\theta}$ will have a *piecewise linear* structure, with a kink at each point i such that $\hat{\theta}_{i+1} \neq (\hat{\theta}_i + \hat{\theta}_{i+2})/2$. These kink points (also called knot points) are chosen adaptively, based on the data

- Figure 4 shows an example of the trend filter on Boston marathon data again. The piecewise linear structure, and qualitative difference to the HP filter, is very clear. (What is the behavior of trend filtering as $\lambda \to \infty$?)

- So, how do you choose the regularization level λ in the trend filter or HP filter, or bandwidth b in a kernel smoother? Cross-validation! (Are you tired of that answer by now?) This is actually closer to traditional CV, rather than time series CV (since we are not in a forecasting setting, where we are predicting the future), but just more structured in how we form the folds

- To tune any one of the smoothers described above, we can define (say) 5 folds using every 5^{th} point in the sequence. That is, we leave out every 5^{th} point, fit the smoother on the remaining points, over a grid of tuning parameter values, and calculate the squared error on the held-out points.[3] Doing this 4 more times, where each time we shift the indices of the held-out points forward by one, we will be able to compute the average error over all points, and use this to select the tuning parameter value. You'll get to see this on the homework

- (An important side note: trend filtering can be extended to model a piecewise polynomial of an arbitrary degree, not just linear; as in the above, the knots in this piecewise polynomial will be chosen adaptively based on the data. So it can generate smoother-looking fits, like the HP filter or kernel smoothing, but the key difference is that it has a property called *local adaptivity*, which comes from its ability to select knots based on the data. Ask about this in office hours if you are curious to hear more!)

4 Additive models

- To gear up to very briefly discuss additive models, we have to first make clear the following point: *each one of the smoothers described in the last section can be extended to a problem where the input points in the underlying smoothing problem are arbitrary:* x_i, $i = 1, \ldots, n$ *(rather than simply* $x_i = i$*, as would be the case in standard time series)*

- Now let's introduce the *additive model*. This extends the basic multivariate linear regression model:

$$y_i \approx \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}, \quad i = 1, \ldots, n$$

[3] To form an estimate at each held-out point, we can use linear interpolation of the estimates at its neighbors.

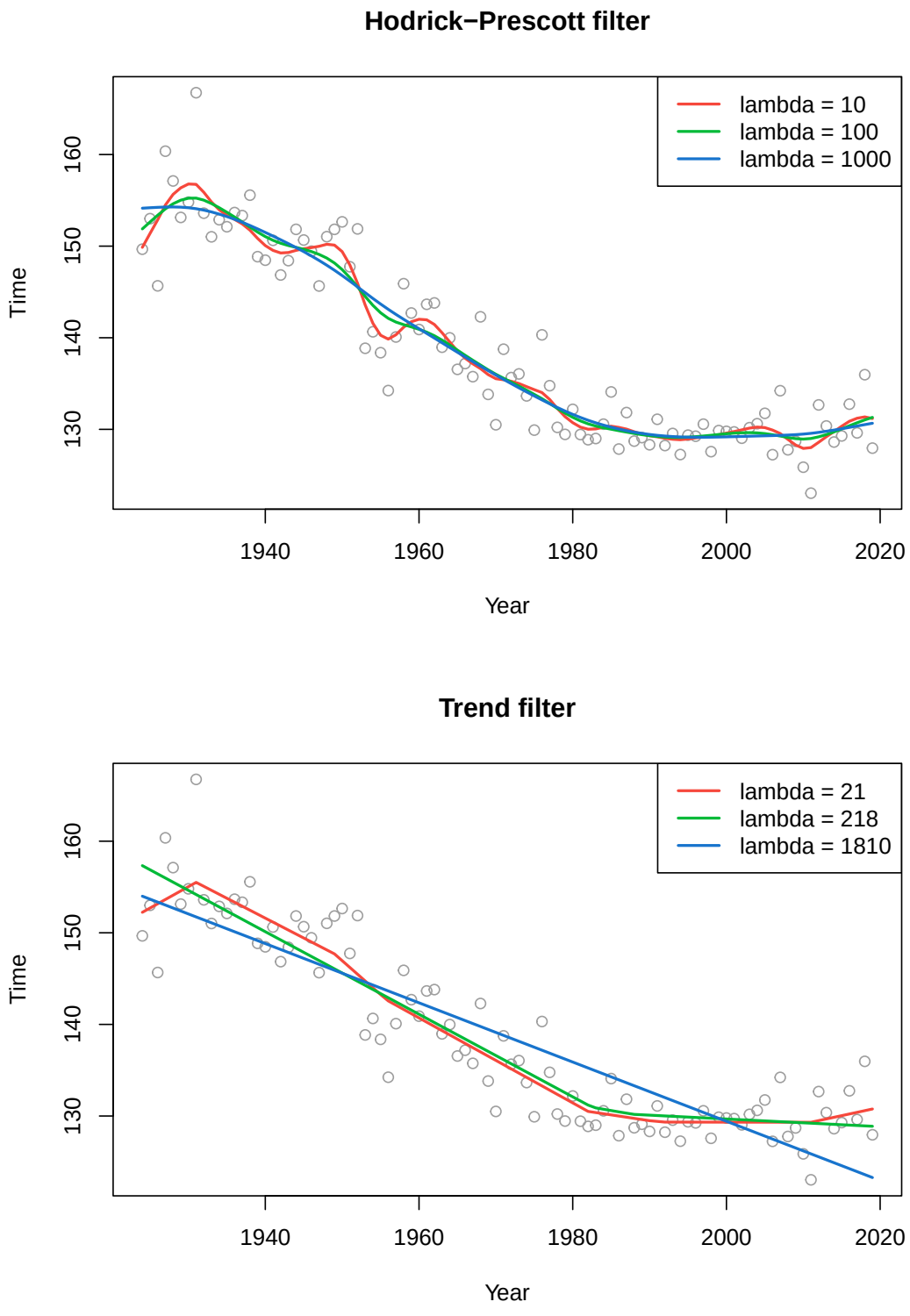

Figure 4: HP filter and trend filter estimates fit to the winning men's Boston marathon times (from HA), each at a few different values of λ.

to

$$y_i \approx \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}), \quad i = 1, \ldots, n$$

where each f_j is a smooth transformation to be estimated

- In other words, we have moved from estimating linear transformations of the features to *nonlinear* ones, which we fit using a smoother (traditional packages use kernel smoothing, or spline smoothing, which is similar to using the HP filter)
- This can be super useful, and hopefully you'll learn about this in more detail in your regression class (or some advanced modeling class). For us, it is mostly just an idea mentioned in passing. But towards the end of the course, when we cover the Prophet forecasting model, you will see it in action