Exercises 5: Sparsity

Jared Fisher
Last Updated October 11, 2016

Broadly speaking, there are three ways to think about enforcing sparsity in a statistical model.[1]

- The old-school frequentist way, based on classical hypothesis tests (chi-squared tests, F tests, likelihood ratio tests, etc), perhaps coupled with greedy algorithms for model selection.
- The Bayesian way, in which sparsity is baked into the prior distribution on some parameter.[2]
- The new-school frequentist way, based on penalizing the likelihood function in a way that encourages sparsity, and computing a *maximum penalized likelihood* estimate.

These viewpoints all have their advantages and disadvantages. One the major advantages of the penalized-likelihood view of sparsity (especially versus the Bayesian view) is scalability to large data sets. That makes it particularly suited to big-data applications, which is why we'll pursue it here, and for much of the rest of the course.

1 Penalized likelihood and soft thresholding

(A) Define the function

$$S_\lambda(y) = \arg\min_\theta \; \frac{1}{2}(y-\theta)^2 + \lambda|\theta| \,. \tag{1}$$

The intuition here is that θ is a parameter of a statistical model, and y is data. The first (quadratic) term rewards good fit to the data, while the second term rewards θ for being "simpler" (i.e. closer to zero). $S_\lambda(y)$ returns an estimate for θ that blends these two goals.

First show (in a trivial one- or two-liner) that the quadratic term in the objective above is the negative log likelihood of a Gaussian distribution with mean θ and variance 1.

$$\begin{aligned} N(y|\theta,1) &= \frac{1}{\sqrt{2\pi}} exp\left\{-\frac{1}{2}(y-\theta)^2\right\} \\ \ell n N(y|\theta,1) &= -\frac{1}{2}\ell n(2\pi) - \frac{1}{2}(y-\theta)^2 \\ -\ell n N(y|\theta,1) &\propto \frac{1}{2}(y-\theta)^2 \end{aligned}$$

Then prove that

$$S_\lambda(y) = \mathrm{sign}(y)\cdot(|y|-\lambda)_+ \,,$$

where $a_+ = \max(a,0)$ is the positive part of a. This is a basic exercise in `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-253-convex-analysis-and-optimization-spring-2` `lecture-notes/MIT6_253S12_lec12.pdf`subdifferential calculus, if you happen to know this subject. But I am not assuming that you do, and you can also prove the statement using ordinary differential calculus if you split it up into cases ($\theta > 0$, $\theta < 0$, and $\theta = 0$).[3] To build intuition, you should try

[1] In this context, sparsity means that some of the parameters are zero. This is different from the notion of sparsity considered previously, where some of the *features* in a regression problem were zero.

[2] E.g. `http://www-stat.wharton.upenn.edu/~edgeorge/Research_papers/fastN96.pdf`here.

[3] If you know about constrained optimization, you could also introduce a slack variable z and write the objective as

$$\frac{1}{2}(y-\theta)^2 + \lambda|z| \,,$$

and then do the minimization subject to the constraint that $\theta = z$.

plotting the objective in Equation 1 for various y and λ, and verifying that $S_\lambda(y)$ indeed obtains the minimum.[4]

To begin, let $\ell(y, *) = \frac{1}{2}(y-\theta)^2 + \lambda|\theta|$.

$$\begin{aligned}\frac{\partial \ell}{\partial \theta} &= -(y-\theta) + \lambda \frac{\partial}{\partial \theta}|\theta| \\ &= \begin{cases} \theta - y + \lambda & \theta > 0 \\ \theta - y - \lambda & \theta < 0 \\ 0 - y + \lambda\partial|\theta| & \theta = 0 \end{cases}\end{aligned}$$

For $\theta > 0$,

$$\begin{aligned} 0 &\overset{set}{=} \hat{\theta} - y + \lambda \\ \hat{\theta} &= y - \lambda \\ \Rightarrow 0 &< y - \lambda \\ \Rightarrow \lambda &< y. \end{aligned}$$

Thus for $y > \lambda$, $\hat{\theta} = y - \lambda$ is optimal. It is similarly shown that for $y < -\lambda$, $\hat{\theta} = y + \lambda$ is optimal. Lastly, For the $\theta = 0$ case we need to employ the subdifferential/subgradient.

$$\begin{aligned} |\theta| &\geq |0| + b \cdot (\theta - 0) \\ |\theta| &\geq b\theta \\ \Rightarrow b &\in [-1, 1] \end{aligned}$$

So the subdifferential is $[-1, 1]$. Note this includes zero. And as this is a convex function, it obtains a global maximum here (right?). Thus, for $\theta = 0$,

$$\begin{aligned} \frac{\partial \ell}{\partial \theta} &\overset{set}{=} 0 = -y + b\lambda \\ &\Rightarrow b = y/\lambda \\ &\Rightarrow \frac{y}{\lambda} \in [-1, 1] \\ &\Rightarrow y \in [-\lambda, \lambda] \end{aligned}$$

Thus $\theta = 0$ is optimal for $y \in [-\lambda, \lambda]$. Hence

$$\begin{aligned} S_\lambda(y) &= arg \min_\theta \frac{1}{2}(y-\theta)^2 + \lambda|\theta| \\ &= \begin{cases} y - \lambda, & y > \lambda \\ y + \lambda, & y < -\lambda \\ 0, & y \in [-\lambda, \lambda] \end{cases} \\ &= \text{sign}(y) \cdot (|y| - \lambda)_+ \end{aligned}$$

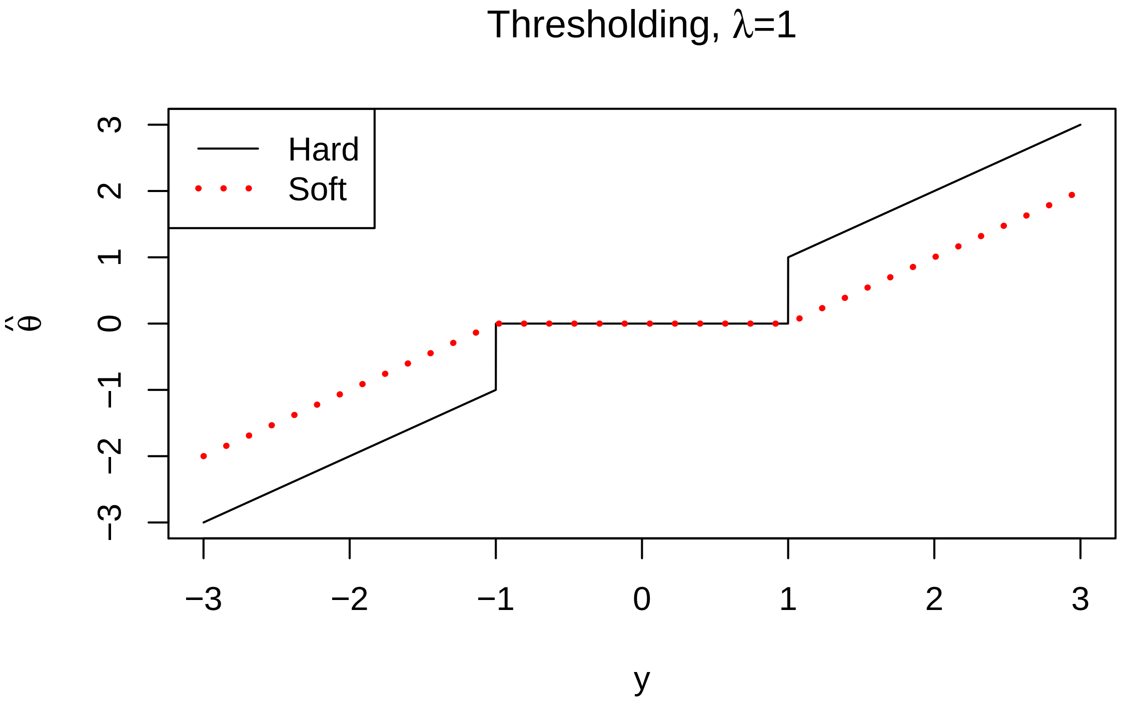
(Thanks to Mingzhang and Carlos' guidance in class, as well as clarification on structuring the argument from Jennifer's writeup!)

$S_\lambda(y)$ is called the *soft thresholding* function with parameter λ. Plot this as function of y for a few different parameters of λ. You'll see how it encourages sparsity in a "soft" way, especially if you compare it to the hard-thresholding function

$$H_\lambda(y) = \begin{cases} y & \text{if } y \geq \lambda \\ 0 & \text{otherwise}. \end{cases}$$

I plot $H_\lambda(y)$ and $S_\lambda(y)$ below. Soft thresholding produces a continuous function.

[4] Recall the "curve" function in R, which acts like a graphing calculator.



(B) Here's a simple toy example to illustrate how soft thresholding can be used to enforce sparsity in statistical models.

Suppose we observe data from the following statistical model:

$$(z_i \mid \theta_i) \sim N(\theta_i, \sigma_i^2)\,.$$

That is, there are n different means θ_i, and we observe 1 normally distributed observation for each one. We allow that each observation has a different variance σ_i^2; for now we'll assume these are known. This is called the Gaussian sequence model, or the normal-means problem. It looks like a toy problem, but is `http://statweb.stanford.edu/~imj/GE06-11-13.pdf`surprisingly useful in a wide variety of applications, from curve fitting to image denoising to genome-wise association studies.[5]

Now suppose we believe that a lot of the θ_i's are zero—i.e. that the vector $\theta = (\theta_1, \ldots, \theta_n)^T$ is sparse. Consider an estimator for each θ_i of the form

$$\widehat{\theta}(y_i) = S_{\lambda\sigma_i^2}(y_i)\,,$$

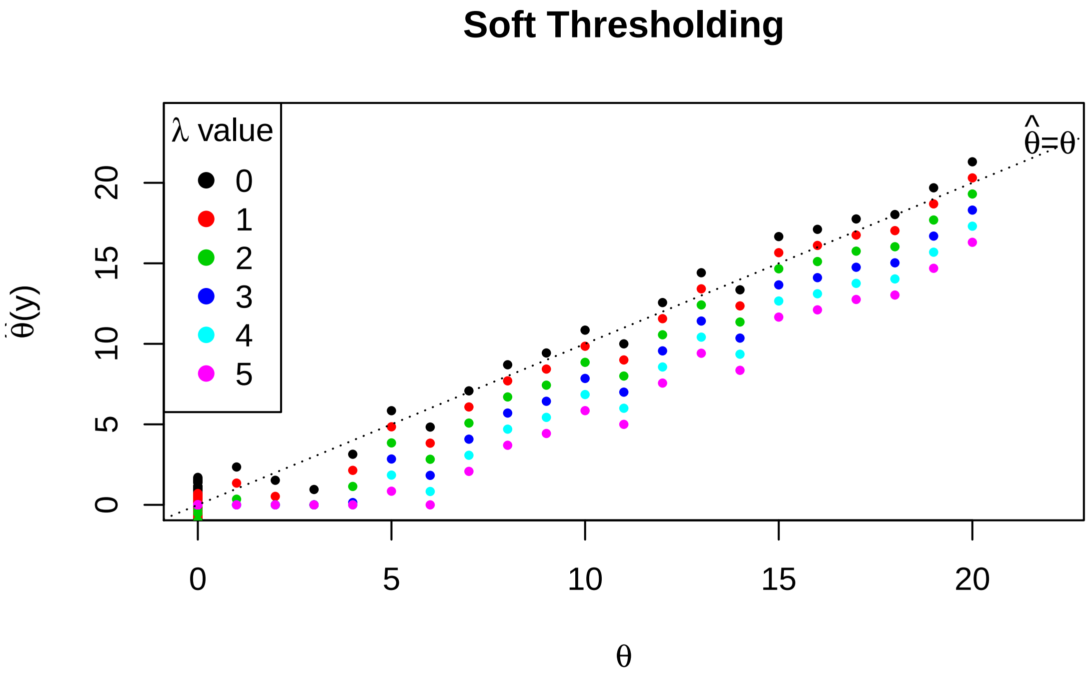
where S is the soft thresholding operator defined above with parameter $\lambda\sigma_i$. (Side question: why $\lambda\sigma_i^2$ for the soft-thresholding parameter?) Because you scale by the standard deviation, and the units will be wrong if using σ^2 instead.

Try the following intuition-building exercise.

1. Choose some sparse vector θ and the corresponding σ_i^2's (it's OK for them all to be equal). You have freedom in deciding what the nonzero elements of θ should be, and how sparse it should be. (Ideally there would be some tunable parameter that you could use to ratchet up or down the sparsity level.)

[5] The simplest example is probably curve-fitting using an orthogonal basis of functions, like Fourier polynomials or wavelets; see the link provided.

2. Simulate one data point $(z_i \mid \theta_i) \sim N(\theta_i, \sigma_i^2)$ for each θ_i.
3. Compute $\widehat{\theta}(y_i) = S_{\lambda\sigma_i^2}(y_i)$ across a discrete grid of different λ values. Plot $\widehat{\theta}(y_i)$ versus θ_i, and observe how the soft-thresholding function both *selects* certain θ_i's by sparsifying the estimate, as well as *shrinks* the nonzero estimates $\widehat{\theta}(y_i)$ towards 0 (and towards each other).



This plot uses 80% sparsity across $n = 100$ points, where the remaining points are given $\theta = 1, ..., 20$. (Thanks to Jennifer's in-class demonstration that known integer values of θ are prettier here.) Also assume all $\sigma_i = 1$.

4. Plot the mean-squared error of your estimate as a function of λ:

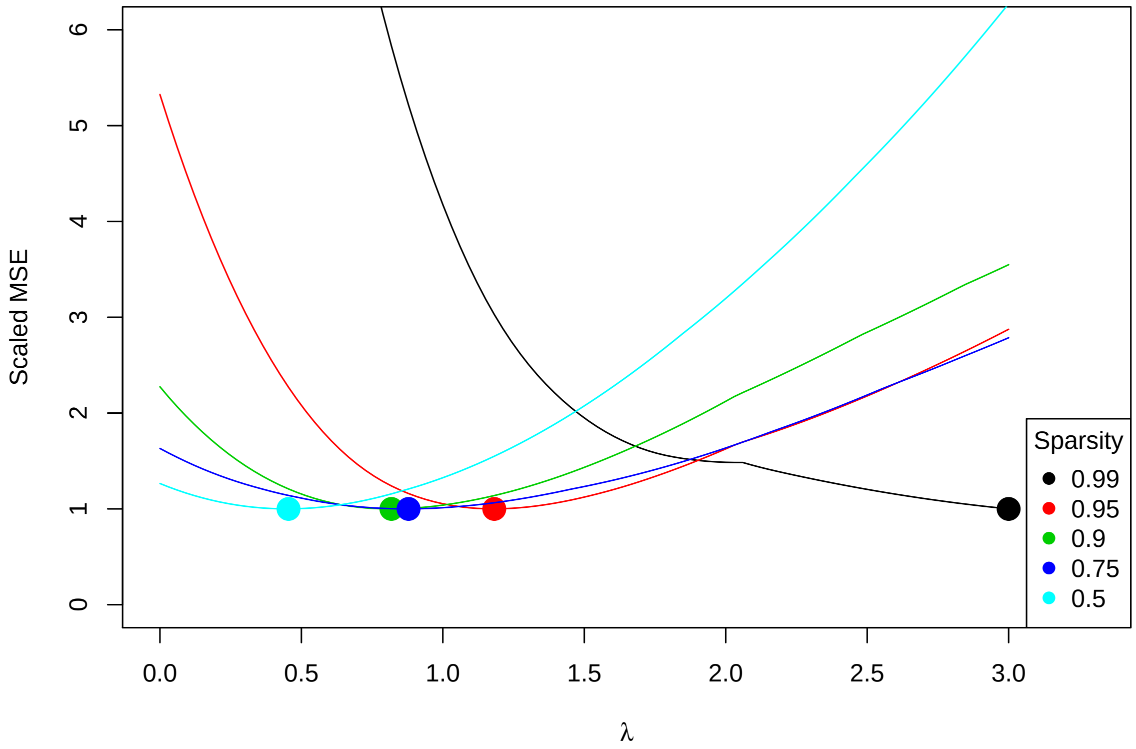
$$\text{MSE}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left\{\widehat{\theta}(y_i) - \theta_i\right\}^2 .$$

Make sure that the MSE actually obtains a minimum across the grid of λ values you have chosen.

Try this for several different configurations of θ. How does the optimal λ change as the sparsity in θ changes?[6]

Note: like I said, this is an intuition-building exercise. You could never choose λ this way for a real problem, because it requires knowledge of the truth.

[6] Here it's enough to assess the optimal λ using the good old-fashioned "plot and point" strategy for optimization. That is, you plot the function, point at the minimum, and say proudly, "Here's the minimum."



The plot above shows how MSE is affected by λ for different levels of sparsity in θ. Clearly, the λ that minimizes MSE increase as θ becomes more sparse. Note that each curve comes from a different randomly chosen vector θ and generated data, so I've scaled each MSE to minimize to 1.

2 The lasso

Although a soft-thresholding approach to the normal-means problem is actually very useful in practice, this utility is not immediately apparent. On the other hand, a generalization of this idea to regression, called the lasso,[7] both looks and is immediately useful.

Consider the standard linear regression model

$$y = X\beta + e \,,$$

where y is an n-vector of responses, X is an $n \times p$ features matrix whose ith row x_i is the vector of features for observation i, and e is a vector of errors/residuals.

The lasso involves estimating β as the solution to the penalized least-squares problem[8]

$$\hat{\beta} = \arg\min_{\beta} \; \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \,,$$

[7] "Lasso" stands for "least absolute shrinkage and selection operator. It was proposed in a `http://statweb.stanford.edu/~tibs/lasso/lasso.pdf`classic paper by Robert Tibshirani.

[8] Note: some will write the "fit" part of the objective function as

$$\frac{1}{2n}\|y - X\beta\|_2^2 \,,$$

where n is the sample size. Translating between these two problems involves multiplying λ by a factor of n.

where $\|\beta\|_1$ is the ℓ_1 (pronounced "ell one") norm of the coefficient vector:

$$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j| \, .$$

As you can see, the penalty function is just like the absolute-value penalty you used on the normal-means problem, generalized to the vector case. And just as in in the normal means problem, this penalty function will have the effect of both selecting a set of nonzero β_i's (i.e. sparsifying the estimate) as well as shrinking the nonzero β_i's toward 0. The bigger λ, the more aggressive the shrinkage effect.

Note: we typically leave the intercept in a lasso fit unpenalized. We can accomplish this by explicitly introducing an intercept, e.g. writing the objective as

$$\frac{1}{2n} \|y - (\alpha \mathbf{1} + X\beta)\|_2^2 + \lambda \|\beta\|_1 \, ,$$

where α is a scalar intercept and $\mathbf{1}$ is a vector of all 1's. Or we can leave the problem in its original form above, and assume that both the response variable y and all columns of the predictor matrix have been standardized have a mean of 0 (in which case there is no need for an explicit intercept). For the rest of these exercises, we'll assume that the variables have been standardized in this way.

To read more about lasso regression, consult Chapter 3.4.2 of *The Elements of Statistical Learning*, or the `http://statweb.stanford.edu/~tibs/lasso/lasso.pdf`original paper by Robert Tibshirani.

(A) For now, we won't worry about *how* the lasso model is fit. Instead, we'll use pre-existing software to fit it:

In R: the package `glmnet`, `https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html`described here.

In Python: `sklearn.linear_model.Lasso` in `http://scikit-learn.org/stable/index.html`scikit-learn.

That way, we can build intuition for its properties, as well as come to an appreciation for some of the statistical questions that arise in fitting the model. If you want to learn about how these functions fit the lasso model, read about coordinate descent in Chapter 3.8.6 of *The Elements of Statistical Learning*, or in `http://arxiv.org/pdf/0708.1485.pdf`this paper.

Download the data on diabetes progression in 442 adults from the Data folder on the class website. There are two files here.

diabetesY.csv: the response variable for each patient. This is the result of a blood test that provides a quantitative measure of disease progression one year after baseline (e.g. at diagnosis).

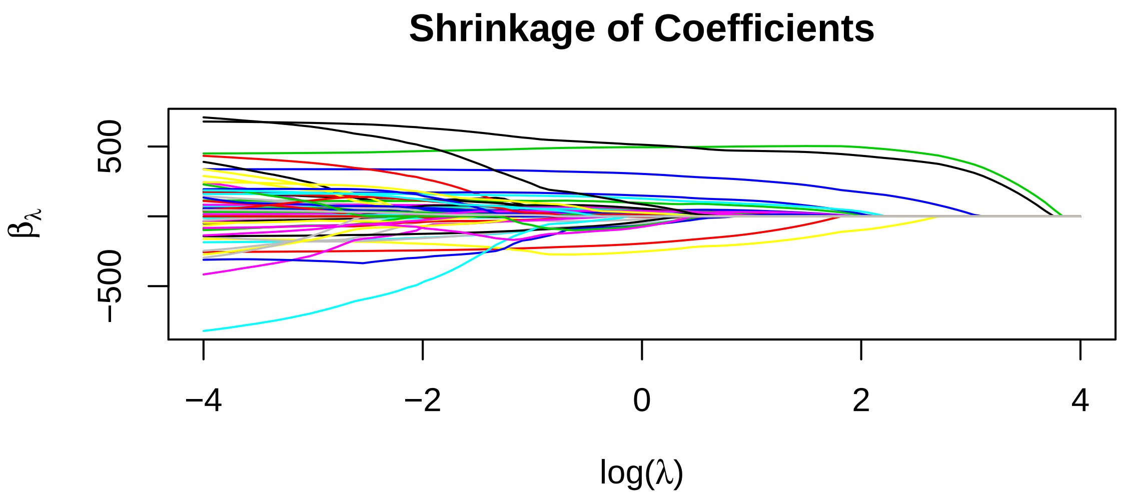
diabetesX.csv: 10 baseline patient variables, age, sex, BMI, cholestorol measurements, etc. Also here are all 10 quadratic terms of the form x_{ij}^2 and all 45 possible pairwise interactions of the form $x_{ij} \cdot x_{ik}$. This leads to 65 total variables: 10 linear main effects and 10 quadratic main effects from the baseline variables, and 45 interactions (45 = 10 choose 2). The 10 baseline variables are standardized to have zero mean and unit Euclidean norm (i.e. the sum of squared entries in the first 10 columns is 1).

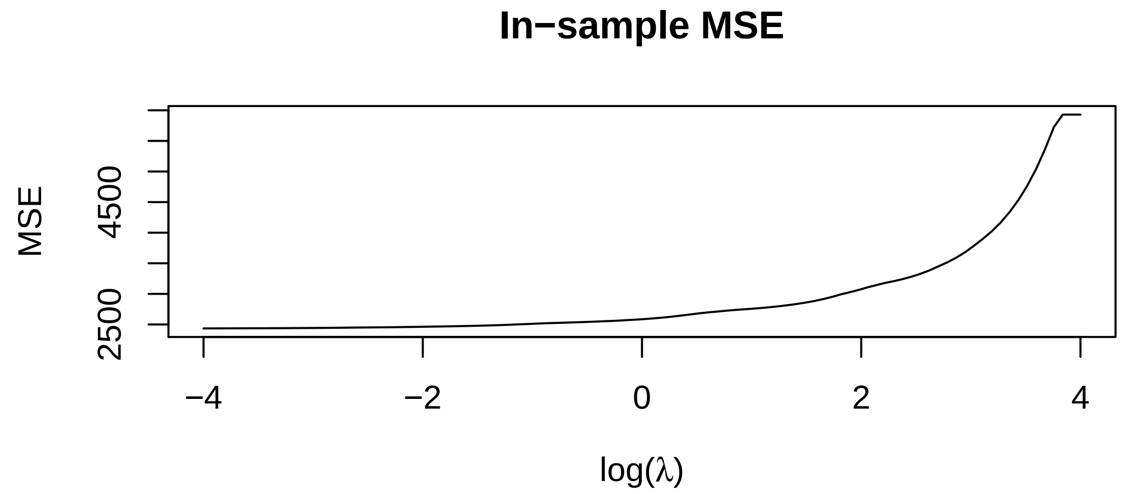
Fit the lasso model across a range of λ values (which `glmnet` does automatically) and plot the solution path $\hat{\beta}_\lambda$ as a function of λ, just like Figure 3.10 in *Elements*.[9] (Note: your horizontal axis can just be λ, or $\log \lambda$.)

In addition, you should track the in-sample mean-squared prediction error of the fit across the solution path:

$$\text{MSE}(\hat{\beta}_\lambda) = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \hat{\beta}_\lambda)^2 = \frac{1}{n} \|y - X\hat{\beta}_\lambda\|_2^2 \, .$$

[9] For Python users, there is a "warm start" option to the lasso fitter, which can use the last solution as an initial guess for the next call. This is great for fitting a solution path.





(B) A natural way to choose λ is to minimize the expected out-of-sample prediction error. Suppose that $(x_\star, y_\star)$ is a future data point from the same population/data-generating process as the original data. The goal would be to make the expected error

$$\text{MOOSE}(\hat{\beta}_\lambda) = E\left\{(y_\star - \hat{y}_\star)^2\right\} = E\left\{(y_\star - x_\star^T \hat{\beta}_\lambda)^2\right\}$$

as small as possible. Here the expected value is taken under what probability distribution generates (x, y) pairs, and "MOOSE" stands for mean out-of-sample squared error.[10] Of course, we don't have any "future data" lying around, so we have to estimate this quantity using the data we have. The in-sample mean-squared error, $\text{MSE}(\hat{\beta}_\lambda)$, is generally an optimistic estimate of this quantity: out-of-sample error tends to be worse, on average, than in-sample error, and we need some way of quantifying how much worse.

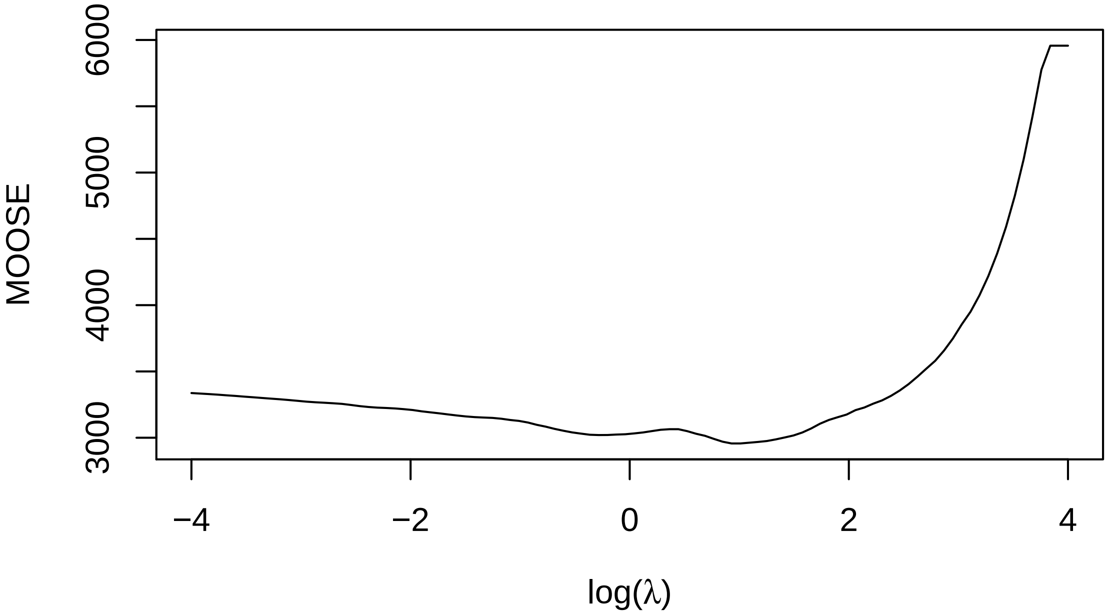
[10] MOOSE is not a standard acronym, but shouldn't it be?! Why yes, of course!

One way is using cross validation. If you're unfamiliar with this concept, read about it in Chapter 7.10 of *Elements* (you may need to track back to the beginning of chapter 7 to pick up their notation). The essential idea is to split your data set into "training" and "testing" sets. Then you fit the model on the training set, and compute the average out-of-sample prediction error on the test set. This provides an estimate of the MOOSE. (You actually average this estimate over multiple such train/test splits, to reduce the influence of randomness in the split itself.)

I leave it to you to figure out the details of cross validation, including how big your training and testing sets should be. Again, Chapter 7.10 *Elements* has good guidelines here. Note: I expect you to write your own code for doing the train/test splits and computing out-of-sample error, wrapped around the base function for fitting the lasso regression model. If you're in R, you can use the `cv.glmnet` function as a sanity check on, but not a replacement for, your results.

Plot your cross-validated estimate of MOOSE($\hat{\beta}_\lambda$) across the solution path, as a function of λ. How does it compare with the *in-sample* mean-squared error from (A)?

The in-sample MSE decreases as λ goes to zero. In other words, in sample fit does not want shrinkage, it wants all 64 coefficients! But that implies a degree of overfitting, which we see below. In the MOOSE from LOOCV (leave-one-out cross validation) below, we see that $\lambda \approx 2.5$ is more appropriate, which implies 15 nonzero coefficients.



(C) Cross validation is one particularly simple way, based on the idea of resampling your data, to estimate the generalization error of a model. (Its reliance on resampling means that it shares a lot in common with `https://en.wikipedia.org/wiki/Bootstrapping_(statistics)`the bootstrap as a way to estimate the standard error of a model parameter.)

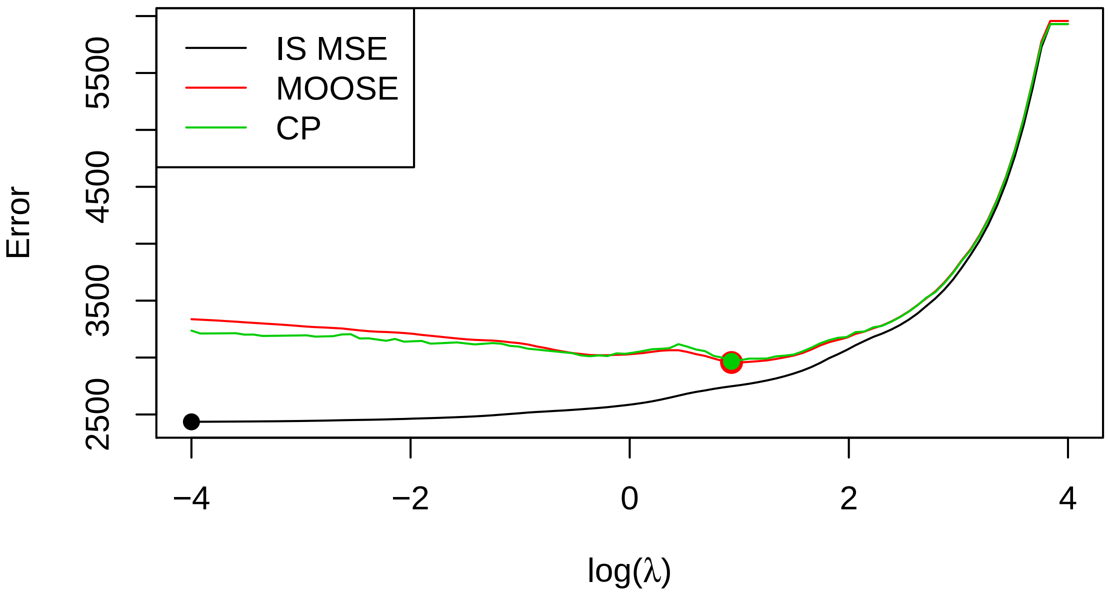
However, cross-validation isn't the only way to estimate generalization error. Another such way is called the C_p statistic, proposed by Collin Mallows (and therefore often called *Mallows' C_p*). For a linear regression model, the C_p statistic is defined as

$$C_p(\hat{\beta}_\lambda) = \text{MSE}(\hat{\beta}_\lambda) + 2 \cdot \frac{s_\lambda}{n} \hat{\sigma}^2 \,,$$

where s_λ is the degrees of freedom of the fit (i.e. the number of nonzero parameters selected at that particular value of λ), and $\hat{\sigma}^2 = \text{var}(\epsilon)$ is an estimate of the residual variance. You can interpret the C_p statistic as the in-sample mean-squared error, plus a penalty for in-sample optimism. As you add parameters to a regression model, MSE goes down and the penalty goes up.

Compute (and plot) the C_p statistic across the solution path, as a function of λ. How does it compare to the in-sample MSE, and to the cross-validated estimate of generalization error from (B)? Show these all on the same plot. Do they lead to similar choices of λ?

The C_p statistic closely follows the cross-validated MOOSE, particularly as λ increases. Most importantly, they appear to agree on a minimum. This minimum from C_p also implies 15 non-zero coefficients. The in-sample MSE clearly prefers overfitting.



Note 1: In general, the main tradeoffs between cross-validation and the C_p statistic are these:

- Cross validation is more robust, in the sense that it does not require that you believe the model is right in order to provide a decent estimate of generalization error. But it is more computationally expensive, and less statistically efficient if the underlying model is approximately right.
- C_p (or related statistics like AIC, BIC, etc) has the advantage that it does not require the repeated "split and refit" of cross validation, and is thus less computationally expensive. It is also more statistically efficient if the model is right. But it is not as robust as cross validation to violations of the underlying modeling assumptions (like linearity).

See the "Estimating prediction error" paper below for more detail.

Note 2: You need to plug an estimate of σ^2 into the C_p statistic. A typical way to proceed is to use the unbiased estimate of σ^2 arising from the ordinary-least-squares solution $\hat{\beta}_{\text{OLS}}$:

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_{i=1}^{n} (y_i - x_i^T \hat{\beta}_{\text{OLS}}^2) \, ,$$

i.e. the model fit to all p variables. Note: this assumes no intercept due to variable centering; if you have an intercept, substract an extra degree of freedom in the denominator (i.e. $n - p - 1$).

Another natural estimator for the residual variance σ^2 is to use, for each λ, the formula

$$\hat{\sigma}_\lambda^2 = \frac{1}{n - s_\lambda} \sum_{i=1}^{n} (y_i - x_i^T \hat{\beta}_\lambda^2) \, ,$$

where again s_λ is the degrees of freedom of the fit. This formula parallels the usual formula given for an unbiased estimate of the error variance for the OLS model.

To be honest, I am not sure if there is any theory supporting the use of one or the other of these variance estimators in the context of the C_p statistic. Mostly I have seen people use the former estimator, based on the OLS fit. The relevant papers here are these, also linked from the class website:

- `https://projecteuclid.org/euclid.aos/1194461726)`Degrees of freedom of the lasso fit
- `https://arxiv.org/abs/1311.5274`Estimating the residual variance from the lasso fit.
- `https://people.eecs.berkeley.edu/~jordan/sail/readings/archive/efron_Cp.pdf`Estimating prediction error. This paper has a much more extensive discussion and list of references about the idea of the C_p statistic and its generalizations.