Exercises 5: Sparsity

(A) Define the function

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

First show that the quadratic term in the objective above is the negative log likelihood of a Gaussian distribution with mean θ and variance 1. Then prove that

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

where $a_+ = \max(a,0)$ is the positive part of a.

If $\theta < 0$:

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

Then take the derivative and set it equal to 0.

$$-y+\theta-\lambda = 0$$

$$\theta = y+\lambda$$

but $\theta < 0$, so $\theta = \min\{y+\lambda, 0\}$.

If $\theta > 0$

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

Then take the derivative and set it equal to 0.

$$-y+\theta+\lambda = 0$$

$$\theta = y-\lambda$$

but $\theta > 0$, so $\theta = \max\{y-\lambda, 0\}$.

If $\theta = 0$, then $\theta = 0$.

Combining these gives us:

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

$S_\lambda(y)$ 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

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

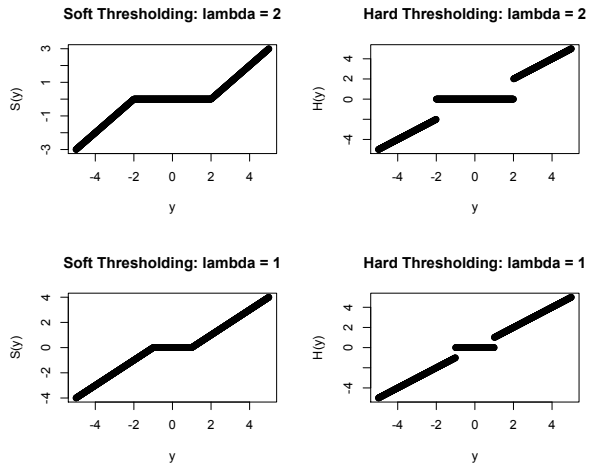

Figure 1: Soft ($S_\lambda(y)$) and Hard ($H_\lambda(y)$) Thresholding for Different Values of λ

(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.

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?)

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.)
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).

Figure 2: Soft Thresholding for Different Values of λ

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?[1]

[1]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."

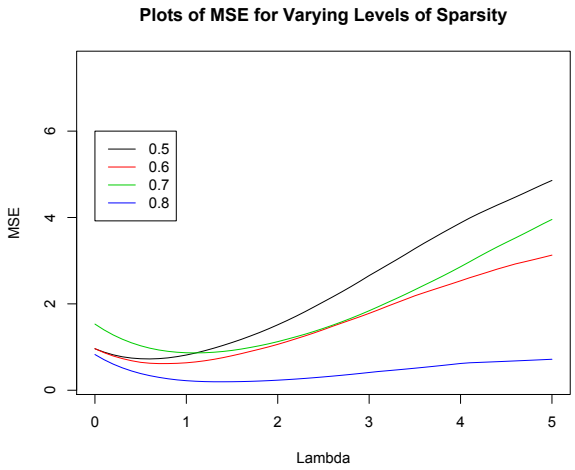

Figure 3: Plot of MSE Over Chosen λ for Varying Levels of Sparsity

1 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,[2] 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[3]

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

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

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

[2] "Lasso" stands for "least absolute shrinkage and selection operator. It was proposed in a classic paper by Robert Tibshirani.

[3] 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.

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.

(A) 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*.

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

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

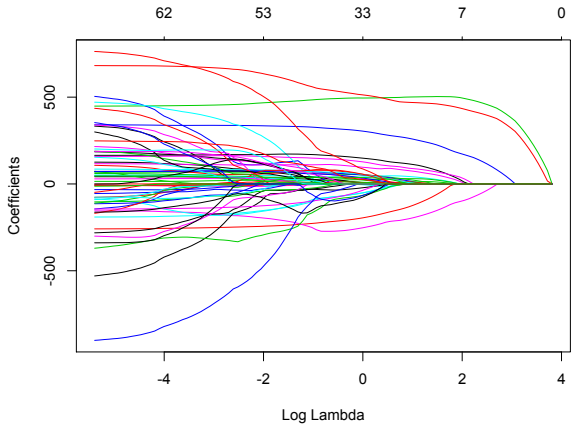

Figure 4: Solution Path

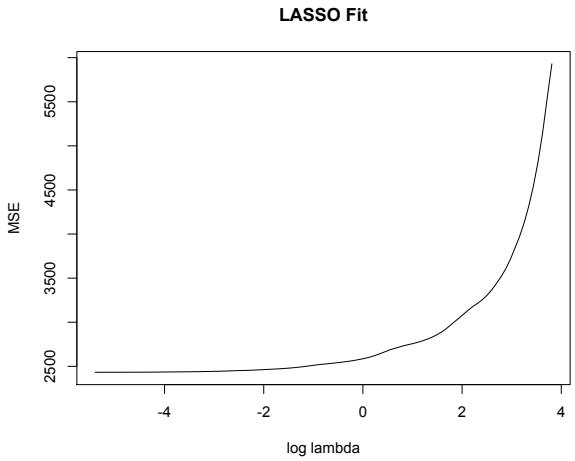

Figure 5: MSE of LASSO Fit

(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. 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.

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

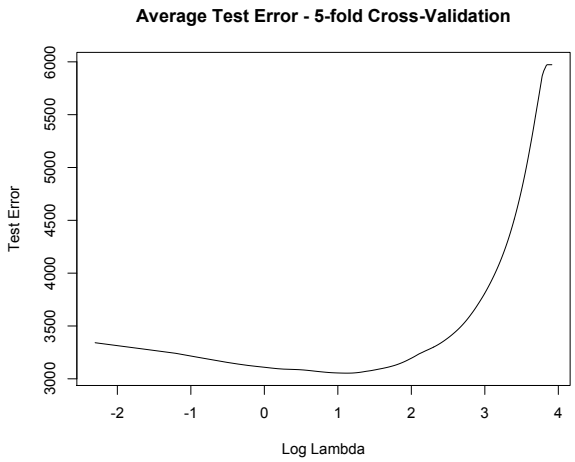

Figure 6: Average Test Error from 5-Fold Cross-Validation

The error from 5-fold cross-validation is higher than the MSE from the LASSO fit using all the data, and it is not monotonically increasing.

(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 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?

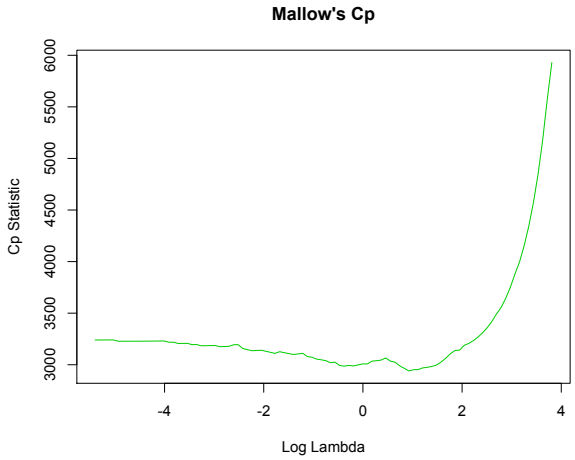

Figure 7: Plot of Mallow's Cp

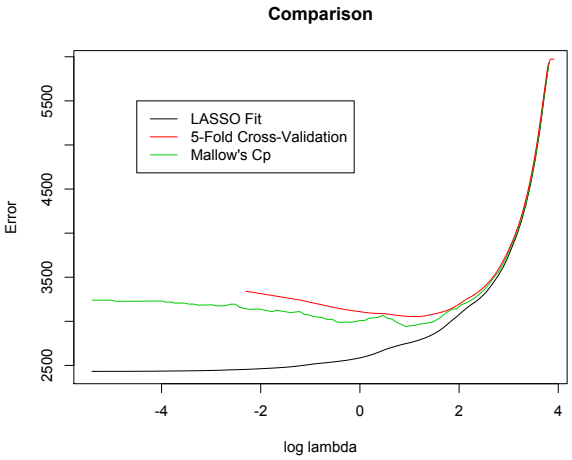

Figure 8: Comparison of LASSO Fit, 5-Fold Cross-Validation, and Mallow's Cp