Bi-level selection

Patrick Breheny

April 29

Introduction

- Our previous lecture introduced the idea of grouped variables and the idea of selecting important groups of variables, rather than individual variables
- However, there are often situations where we might be interested in selection at both the individual and group levels, or *bi-level selection*
- Our goal for today is to introduce two approaches for achieving bi-level selection, discuss some specific penalties, and apply the approach to a real data set

Introduction (cont'd)

- For example, last time we analyzed a data set in which genetic differences (SNPs) were grouped by the gene that they belong to
- Grouping made sense here: if the gene is unimportant to the response, we don't want to select any SNPs from it
- However, selecting individual SNPs also makes sense: just because a gene is important to the response doesn't mean that every single SNP is important
- This could be thought of as a situation in which the grouping is "soft": if feature A is in a group with feature B that we know is important, this means that feature A is more likely to be important, but this is not definite

Sparse group lasso

- One simple way of achieving bi-level selection is to include both a lasso and group lasso penalty:

$$Q(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) = L(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) + \lambda_1 \sum_j \sum_k |\beta_{jk}| + \lambda_2 \sum_j \|\boldsymbol{\beta}_j\|;$$

  this penalty is known as the *sparse group lasso* (SGL)
- Similar to the elastic net, it is common to reparameterize this penalty using λ and α, with $\lambda_1 = \alpha\lambda$ and $\lambda_2 = (1-\alpha)\lambda$ so that $\alpha = 1$ is equivalent to the lasso, $\alpha = 0$ is equivalent to the group lasso, and $\alpha = 0.5$ is a 50-50 mix

Derivative of the penalty

- To get some insight into how the penalty works, let's consider the partial derivative of the penalty with respect to $|\beta_{jk}|$, which I will denote in today's lecture as Δ_{jk}:

$$\Delta_{jk} = \lambda_1 + \begin{cases} \lambda_2 \frac{\beta_{jk}}{\|\boldsymbol{\beta}_j\|} & \text{if } \boldsymbol{\beta}_j \neq \mathbf{0} \\ \lambda_2 & \text{if } \boldsymbol{\beta}_j = \mathbf{0} \end{cases}$$

- In other words, if all the other elements of group j are zero, β_{jk} receives the full penalty of $\lambda_1 + \lambda_2$
- If, however, β_{jk} is located in a group with other important variables (i.e., with large coefficients), it receives a lesser penalty $\lambda_1 + \epsilon\lambda_2$, where $\epsilon \in [0, 1)$

Computing

- In terms of developing an algorithm to solve for $\widehat{\boldsymbol{\beta}}$, unfortunately there is no longer a closed-form solution at the individual or group level
- There would be, if we could assume $\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j = \mathbf{I}$ as we did with the group lasso
- Unfortunately, we can no longer apply the orthonormalization trick from the previous lecture – if we were to compute the orthonormalized group $\widetilde{\mathbf{X}}$, its columns would no longer correspond to the original columns of $\mathbf{X}$
- To put it a different way, we could achieve bi-level selection on orthonormalized scale, but this would be lost once we transformed back to the original scale

Computing (cont'd)

- One option would be to use a local linear approximation to the penalty, where we would end up with expressions like the one we just derived
- A different approach (used by the `SGL` package, which we will be using today) is to employ an idea known as generalized gradient descent, in which one calculates a direction (gradient) along which we will update $\boldsymbol{\beta}_j$, then applies a soft-thresholding operator along that gradient
- In a sense, this is like calculating an orthonormal approximation to $\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j$ and then using its closed form in the orthonormal case to carry out group-wise updates

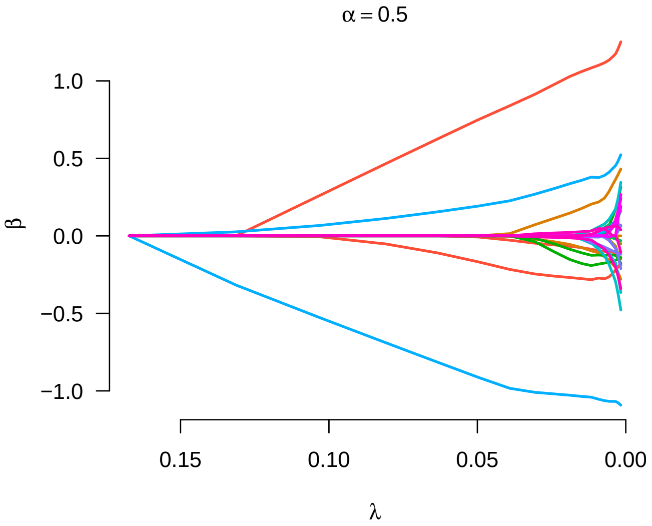
Other options; convexity

- The sparse group lasso adds the lasso and group lasso penalties
- In principle, one could imagine mixing other penalties (e.g., MCP + group lasso), but to my knowledge these have not been investigated
- One attractive feature of the SGL, however, is the fact that, since both lasso and group lasso are convex penalties, the resulting objective function is convex

Example

- To see an example of SGL in action, let's simulate some data with $n = 50$, $x_{ij}, \epsilon \overset{\perp\!\!\!\perp}{\sim} \mathrm{N}(0, 1)$ and
  - Coefficients in 10 groups of three ($p = 30, J = 10$)
  - One group with $\boldsymbol{\beta}_j = (1, -0.5, 0)$, another group with $\boldsymbol{\beta}_j = (-1, 0.5, 0)$, and the other eight groups with $\boldsymbol{\beta}_j = \mathbf{0}$
- We'll fit SGL models over $\alpha = 0, 0.1, 0.2, \ldots, 1$ and look at how the coefficient paths change

Example: SGL paths



Hierarchical framework

- An alternative approach is to apply penalties in a hierarchical manner, as opposed to an additive one
- For example, suppose we have an outer penalty, p_O, applied at the group level, and an inner penalty, p_I, applied at the individual feature level; the objective function would be

$$Q(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) = L(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) + \sum_j p_O\Big\{\sum_k p_I(|\beta_{jk}|)\Big\},$$

  where p_O and p_I would also depend on various tuning/regularization parameters
- For example, group lasso could be thought of in this framework, with $p_O(\theta) = \lambda_j\,|\theta|^{1/2}$ and $p_I(\beta) = \beta^2$

Derivative; insight

- Again, to gain insight into the nature of penalties of this type, let us consider the derivative with respect to (the absolute value of) an individual coefficient:

$$\begin{aligned}\Delta_{jk} &= p_O'\Big(\sum_k p_I(|\beta_{jk}|)\Big)p_I'(|\beta_{jk}|) \\ &= \lambda_O\lambda_I\end{aligned}$$

- In other words, thinking of λ_I as the penalty experienced by a coefficient in the ungrouped case, this rate of penalization is multiplied by a term λ_O that depends on the size of the group that the coefficient belongs to

Remarks

- In the hierarchical framework, then, group and individual penalties interact in a multiplicative manner, as opposed to an additive manner in a penalties like SGL
- Note that, for this to make sense, the outer penalty p_O must be nonconvex – i.e., its rate of penalization must be decreasing as the size of the group increases

Group exponential lasso

- As with additive penalties, one could imagine many possible combinations here; I will briefly discuss one called the *group exponential lasso* (GEL)
- Here, the inner penalty is the the lasso penalty, $p_I(\boldsymbol{\beta}_j) = \|\boldsymbol{\beta}_j\|_1$ and the outer penalty is the exponential penalty

$$p_O(\theta|\lambda,\tau) = \frac{\lambda^2}{\tau}\left\{1 - \exp\left(-\frac{\tau\theta}{\lambda}\right)\right\}$$

Derivative of the GEL penalty

- For the GEL penalty,

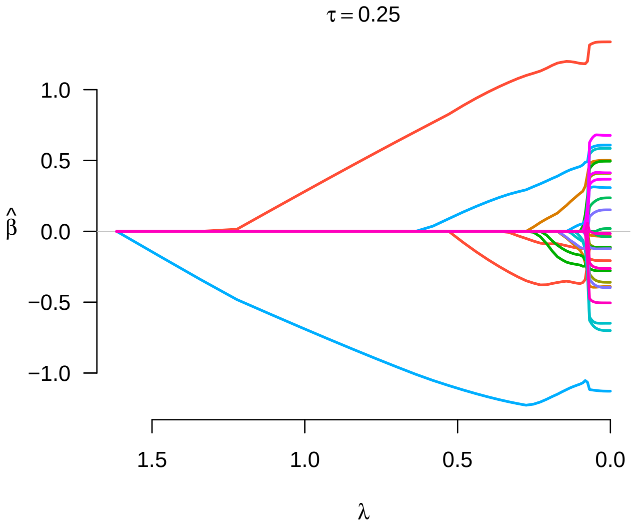
$$\Delta_{jk} = \lambda \exp\left\{-\frac{\tau}{\lambda}\,\|\boldsymbol{\beta}_j\|_1\right\}$$

- Thus, for a coefficient in a group with $\boldsymbol{\beta}_j = \mathbf{0}$, the penalty is λ, just as it is for the ordinary lasso
- When $\boldsymbol{\beta}_j \neq \mathbf{0}$, however, $\Delta_{jk} < \lambda$, with the rate of penalization decreasing exponentially as $\|\boldsymbol{\beta}_j\|_1$ increases
- Note that in this approach, the rate of penalization is the same for all features in a given group, so we could drop the subscript k

Computing

- Computing can be carried out in a relatively straightforward manner using the idea of local linear approximation that we discussed in earlier lectures
- To briefly address the ideas of convexity and convergence:
  - Because the penalty function is strictly nonconvex in $|\beta|$, the algorithm is guaranteed to converge by theory underlying MM algorithms
  - However, as with all iterative algorithms applied to nonconvex problems, we cannot guarantee convergence to a global minimum
- Here, τ is the parameter that controls the convexity of the objective function, with larger values of τ leading to increasingly nonconvex objectives

Example: GEL paths (same data as earlier)



Macular degeneration case study

- To illustrate how SGL and GEL work, and how they compare to lasso/group lasso, we will revisit our example from last time involving the case/control study of macular degeneration
- Here, $n = 800$, $p = 497$, $J = 30$, and the outcome is binary; for the sake of simplicity I'll focus only on the "grouping by gene" analysis

R code

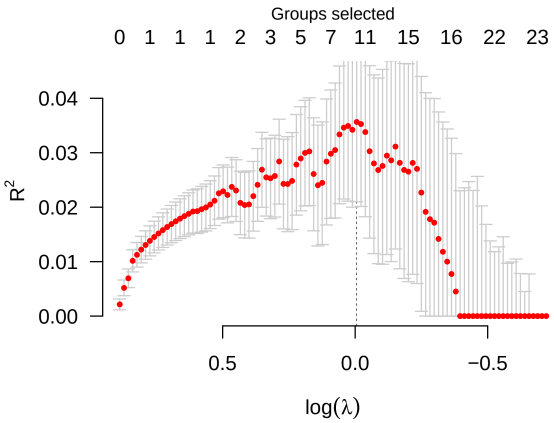
- An implementation of the SGL penalty is available from the R package SGL
- Its syntax is a little unconventional, and the package is not as well developed as some of the others (e.g., no plot function), but one can fit SGL models via:

```
cvSGL(list(x=X, y=y), index=nGene,
      type="logit", alpha=0.5)
```

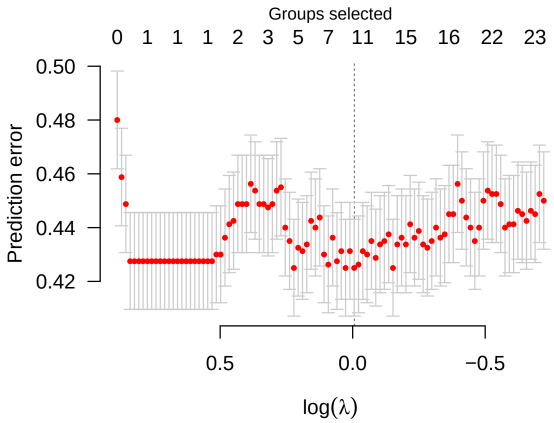
  note that SGL requires integer-indexing of genes
- The GEL penalty is available in grpreg; we have seen its syntax previously:

```
cv.grpreg(X, y, group=Gene, family="binomial",
          penalty="gel")
```

Results: GEL (R^2)



Results: GEL (ME)



Remarks

Here, GEL doesn't necessarily outperform either lasso or group lasso in terms of prediction, but does provide much more sparse solutions:

Method	R^2	PE	#Genes	#Variants
Lasso	0.04	0.42	32	32
Group lasso	0.05	0.40	24	405
GEL	0.04	0.42	11	25
SGL	0.01		30	358