Group lasso

Patrick Breheny

April 27

Introduction

- So far in this class, we have spent a lot of time talking about the selection of individual variables
- In many regression problems, however, predictors are not distinct but arise from common underlying factors
- The most obvious example of this occurs when we represent a categorical factor by a group of indicator functions, but this actually comes up fairly often:
    - Continuous features may be represented by a group of basis functions
    - Groups of measurements may be taken in the hopes of capturing unobservable latent variables

Potential advantages of grouping

- Today's lecture will look at these cases where features can be organized into related groups, and focus on methods for selecting important groups and estimating their effects
- One could, of course, still use methods like the lasso in these cases
- However, if there is indeed information contained in the grouping structure, by ignoring it these methods will likely be inefficient
- Furthermore, by selecting important groups of variables, we should obtain models that are more sensible and interpretable

Notation

To proceed in the grouped variable case, let us extend our usual notation as follows:

- We denote $\mathbf{X}$ as being composed of J groups $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J$, with K_j denoting the size of group j; i.e., $\sum_j K_j = p$
- As usual, we are interested in estimating a vector of coefficients $\boldsymbol{\beta}$ using a loss function L which quantifies the discrepancy between the observations $\mathbf{y}$ and the linear predictors $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} = \sum_j \mathbf{X}_j \boldsymbol{\beta}_j$, where $\boldsymbol{\beta}_j$ represents the coefficients belonging to the jth group
- Covariates that do not belong to a group may be thought of as a group of one

The group lasso penalty

- Consider, then, the following penalty, known as the *group lasso* penalty:

$$\mathbf{Q}(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) = \mathbf{L}(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) + \sum_j \lambda_j \left\|\boldsymbol{\beta}_j\right\|$$

- This is a natural extension of the lasso to the grouped variable setting: instead of penalizing the magnitude ($|\beta_j|$) of individual coefficients, we penalize the magnitude ($\|\boldsymbol{\beta}_j\|$) of groups of coefficients
- To ensure that the same degree of penalization is applied to large and small groups, $\lambda_j = \lambda\sqrt{K_j}$; we will discuss the reasoning behind this in a bit

Group orthonormal setting

- To gain insight into other penalties, we have considered the orthonormal setting in which $\mathbf{x}_j^T\mathbf{x}_k = 0$ for $j \neq k$
- The equivalent for the grouped variable case is to suppose that $\mathbf{X}_j^T\mathbf{X}_k = \mathbf{0}$ for $j \neq k$
- In what follows, we will also suppose that $\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j = \mathbf{I}$ for all j; we will discuss this condition further in the next section

Group orthonormal solution

Theorem: Suppose $\mathbf{X}_j^T\mathbf{X}_k = \mathbf{0}$ for $j \neq k$ and $\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j = \mathbf{I}$ for all j. Letting $\mathbf{z}_j = \frac{1}{n}\mathbf{X}_j^T\mathbf{y}$ denote the OLS solution, the value of $\boldsymbol{\beta}$ that minimizes

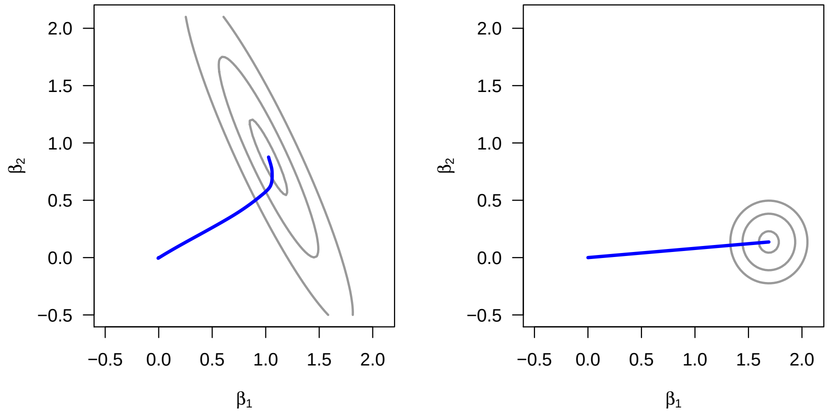
$$\frac{1}{2n}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \sum_j \lambda_j \left\|\boldsymbol{\beta}_j\right\|$$

is given by

$$\widehat{\boldsymbol{\beta}}_j = S(\|\mathbf{z}_j\|, \lambda_j)\frac{\mathbf{z}_j}{\|\mathbf{z}_j\|},$$

where $S(z, \lambda)$ denotes the soft-thresholding operator

Geometry of solution



Group MCP and SCAD

- The fact that group penalization reduces to a one-dimensional problem in the group orthonormal setting means that extending it to MCP and SCAD penalties is straightforward
- For example, if we replace the group lasso penalty with a group MCP penalty $P(\boldsymbol{\beta}) = \sum_j \mathrm{MCP}(\|\boldsymbol{\beta}_j\|; \lambda_j, \gamma)$, the solution is

  $$\widehat{\boldsymbol{\beta}}_j = F(\|\mathbf{z}_j\|, \lambda_j, \gamma)\frac{\mathbf{z}_j}{\|\mathbf{z}_j\|},$$

  where $F(z|\lambda_j, \gamma)$ is the firm thresholding penalty
- Likewise for SCAD and its thresholding solution

A closer look at the orthonormality assumption

- To solve for $\widehat{\boldsymbol{\beta}}$, one's first instinct might be to apply this closed-form solution to each group sequentially, as we did with coordinate descent
- We need to be careful, however: our closed form solution assumed $\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j = \mathbf{I}$, which is not the case in general
- Nevertheless, it turns out we can always make this assumption hold by transforming to the orthonormal case and then transforming back to obtain solutions on the original scale

Orthonormalizing $\mathbf{X}_j$

- Consider the singular value decomposition of group j:

$$\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j = \mathbf{Q}_j\mathbf{\Lambda}_j\mathbf{Q}_j^T,$$

  where $\mathbf{\Lambda}_j$ is a diagonal matrix containing the eigenvalues of $\frac{1}{n}\mathbf{X}_j^T\mathbf{X}_j$ and $\mathbf{Q}_j$ is an orthonormal matrix of its eigenvectors
- Now, we may construct a linear transformation $\widetilde{\mathbf{X}}_j = \mathbf{X}_j\mathbf{Q}_j\mathbf{\Lambda}_j^{-1/2}$ with the following properties:

$$\begin{aligned}\frac{1}{n}\widetilde{\mathbf{X}}_j^T\widetilde{\mathbf{X}}_j &= \mathbf{I}\\ \widetilde{\mathbf{X}}_j\widetilde{\boldsymbol{\beta}}_j &= \mathbf{X}_j(\mathbf{Q}_j\mathbf{\Lambda}_j^{-1/2}\widetilde{\boldsymbol{\beta}}_j)\end{aligned}$$

  where $\widetilde{\boldsymbol{\beta}}_j$ is the solution on the orthonormalized scale

Remarks

- This procedure is not terribly expensive from a computational standpoint; although computing the SVD requires $O(p^3)$ steps, the decompositions are being applied only to the groups, not the entire design matrix
- Furthermore, the decompositions need only be computed once initially, not with every iteration
- We could use a Cholesky decomposition here instead of a SVD, but the advantage of the SVD is that it works for groups that do not have full rank

Group standardization

- Orthonormalization is essentially the grouped-variable equivalent of standardization
- To explore this point further, note that we could consider penalizing on the scale of the linear predictors, not the coefficients themselves: $\text{Penalty} = \lambda \sum_j \|\boldsymbol{\eta}_j\|$, where $\boldsymbol{\eta}_j = \mathbf{x}_j \beta_j$ for the single-variable case or $\mathbf{X}_j \boldsymbol{\beta}_j$ in the grouped-variable case
- Note that this reduces to the lasso penalty for standardized design matrix and the group lasso penalty when the groups have been orthonormalized

Connection with χ^2/F tests

- As a final justification for group orthonormalization, consider the form of the (UMPI) χ^2/F test for adding a group in classical linear regression
- In that case, the test statistic for $H_0 : \boldsymbol{\beta}_j = \mathbf{0}$ takes the form

$$\|\mathbf{P}_j \mathbf{r}_0\|^2 \geq \sigma^2 \chi^2_{K_j, 1-\alpha}$$

(or $\geq \hat{\sigma}^2 F_{K_j, \text{rdf}, 1-\alpha}$), where $\mathbf{P}_j = \mathbf{X}_j (\mathbf{X}_j^T \mathbf{X}_j)^{-1} \mathbf{X}_j^T$ is the orthogonal projection operator for group j

- Now, since $\|\mathbf{P}_j \mathbf{r}_0\| \propto \|\mathbf{Z}_j\|$ for an orthonormal group, we can see that the UMPI χ^2 test is essentially equivalent to the group lasso inclusion condition $\|\mathbf{Z}_j\| > \lambda_j$, provided that λ_j includes a $\sqrt{K_j}$ term to account for the size of group j

Algorithm

With these ideas in place, we can apply the coordinate descent idea in a groupwise fashion; this algorithm is known as *group descent*, *blockwise coordinate descent*, or the "shooting algorithm"

```
repeat
    for $j = 1, 2, \ldots, J$
        $\mathbf{z}_j = \mathbf{X}_j^T \mathbf{r} + \boldsymbol{\beta}_j$
        $\boldsymbol{\beta}'_j \leftarrow S\left(\|\mathbf{z}_j\|, \lambda_j\right) \mathbf{z}_j / \|\mathbf{z}_j\|$
        $\mathbf{r}' \leftarrow \mathbf{r} - \mathbf{X}_j(\boldsymbol{\beta}'_j - \boldsymbol{\beta}_j)$
until convergence
```

For MCP/SCAD, we would replace the soft thresholding step with the appropriate thresholding operator

Remarks

- Although there is an initial cost in terms of computing the SVD for each group, once this is done the cost per iteration for the group descent algorithm is simply $O(np)$
- Because the penalty is separable in terms of the groups $\boldsymbol{\beta}_j$, and because we are updating whole groups at once, the algorithm is guaranteed to decrease the objective function with every step and to converge to a minimum, as it was for the ordinary lasso
- Extensions to other loss functions (GLMs, etc.) follow along the lines we discuss in the previous topic

Mullins study

- As a case study, we will consider data from a study at the University of Iowa investigating the genetic causes of primary open-angle glaucoma (POAG) and age-related macular degeneration (AMD)
- In the study, 400 AMD patients and 400 POAG patients were recruited, and their genotype determined at 500,000 genetic loci, the idea being that each group could serve as the control group group for the other disease
- We will not consider all 500,000 genetic loci, but rather a subset of 532 SNPs from 30 genes that have been previously indicated in AMD

Grouping

- There are two ways we could consider grouping the data here, and for the sake of illustration, we will do both
- At each genetic loci, there are three possibilities: AA, AB, BB, where A and B stand for the two alleles present in the population at those loci
- Thus, we could consider constructing a design matrix with indicators for each possibility and group by genetic locus ($p = 1596, J = 532$)
- Alternatively, we could represent each locus by the number of B alleles (0/1/2) and group by the gene that the locus belongs to ($p = 532, J = 30$)

grpreg

- Fitting group lasso (and group MCP/SCAD) models can be carried out using the `R` package `grpreg`
- In what follows, suppose `X` is the 800×532 matrix of 0/1/2 counts, `XX` is the $800 \times 1,596$ matrix of indicators, `Gene` is a length 532 vector denoting the gene that each column of `X` belongs to, and `Locus` is a length 1,596 vector denoting the locus that column of `XX` belongs to
- Here, `Locus` and `Gene` are both factors or unique IDs (integers, character strings) that can be coerced into factors

grpreg (cont'd)

- The syntax of `grpreg` is similar to `glmnet` and `ncvreg`, although we must also pass the function a `group` option that describes the grouping structure
- So, for the first approach (group by gene) we would have

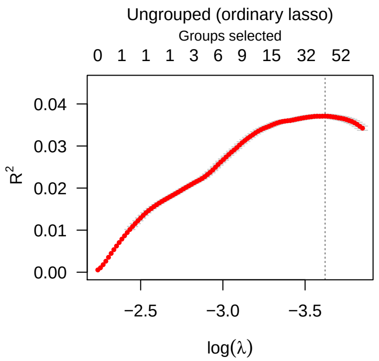
  ```
  cv.grpreg(X, y, group=Gene, family="binomial")
  ```

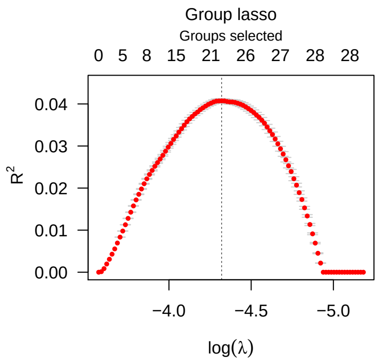
  while for the second (group by locus) we would have

  ```
  cv.grpreg(XX, y, group=Locus, family="binomial")
  ```

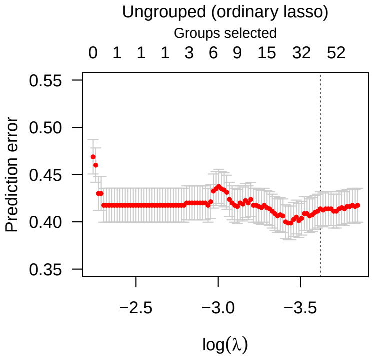
- The same sorts of downstream methods such as `plot(cvfit)`, `summary(cvfit)`, and `predict(cvfit)` are available after the models have been fit

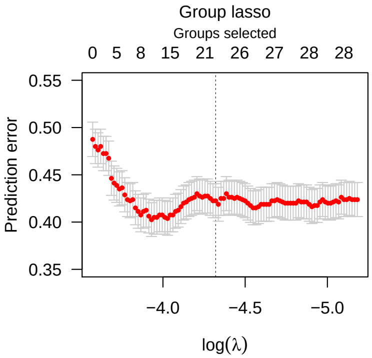
R^2: Grouping by gene





Misclassification error: Grouping by gene

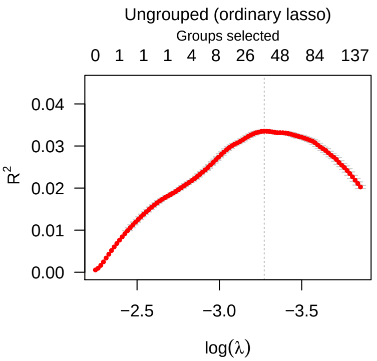


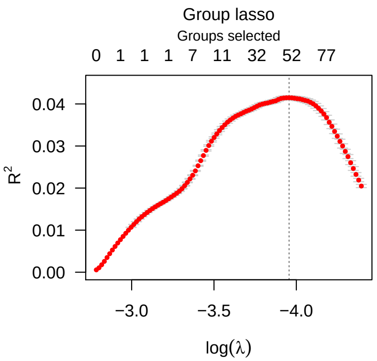


Remarks

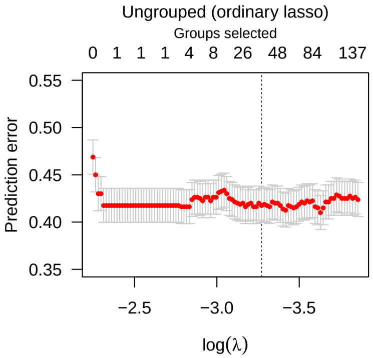
- In this case, there is a subtle, but not overwhelming, advantage to grouping
- Grouping achieves a slightly higher R^2, 0.040 to 0.037, but the two methods have the same minimum misclassification error of 0.40
- However, at their respective λ values for which that minimum misclassification error is achieved, the ordinary lasso selects loci ranging across 20 of the 30 genes, while the group lasso confines its selects to just 11 genes, achieving greater sparsity at the group level and, likely, more interpretable results

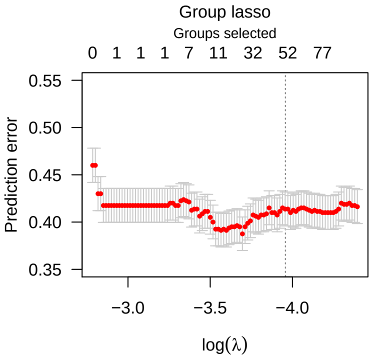
R^2: Grouping by locus





Misclassification error: Grouping by locus





Remarks

- Incorporating grouping information is perhaps a bit more convincing in this case, as the grouped approach appears to confer a clear advantage in terms of more accurate predictions
- The group lasso model achieves a better R^2 (0.041 to 0.034) and misclassification error (0.39 to 0.41) than the ordinary lasso