Sparse interaction modeling

Patrick Breheny

May 1, 2025

Introductions

- Today we'll be discussing how to include interactions in penalized regression models
- In principle, of course, you could just create a giant matrix containing all the main effects and two-way interaction terms and then apply the lasso (this is known as the all pairs lasso, or APL)
- However, this has a clear drawback: the number of interactions is far larger than the number of main effects, so our selected model likely consists mostly of interactions
    - For example, if we have 100 features, there are 4,950 two-way interactions

Different penalization

- A simple way to address this would be to set a different penalty factor for the main effects and interactions, with the interactions facing a higher penalty
- This, too, is unsatisfying because it doesn't respect the well-established practice among statisticians of only considering an interaction if the corresponding main effects are also in the model
- This general concept is usually called *hierarchy* or sometimes *heredity*

Strong and weak hierarchy

- This principle exists in two different forms
  - **Strong hierarchy:** An interaction is allowed only if both main effects are included
  - **Weak hierarchy:** An interaction is allowed if either main effect is included
- It is worth noting that from a scientific perspective, there is no guarantee that either form of hierarchy exists, but there are many practical reasons for preferring hierarchy (interpretation, power, cost)

Group lasso setup

- One approach is to set the problem up as a group lasso problem
- To illustrate, let's consider the simplest possible scenario, in which we have two features $\mathbf{x}_1$ and $\mathbf{x}_2$, along with their interaction $\mathbf{x}_{1:2}$
- Now let us construct a 5-column design matrix with columns $(\mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_{1:2})$, where we will consider the first and second columns as groups containing just a single element, and the last three belonging to a combined group (i.e., `group = c(1, 2, 3, 3, 3)`)

Interpretation and latent variable representation

- The idea behind this approach is that we are parsing the main effect of $\mathbf{x}_1$ into two latent portions: the pure main effect portion and the portion belonging to the interaction group
- Letting γ denote the the coefficients for this expanded design matrix, the main effect for $\mathbf{x}_1$ would then be

$$\begin{aligned}\mathbf{x}_1\gamma_1 + \mathbf{x}_1\gamma_3 &= \mathbf{x}_1(\gamma_1 + \gamma_3) \\ &= \mathbf{x}_1\beta_1\end{aligned}$$

- As a consequence of this setup, if we select $\mathbf{x}_{1:2}$, we are guaranteed to also include its two main effects in the model as well

Simulation example

- To see how well this works, let's simulate some data under the following conditions:
  - $n = 70$, $p = 20$; so 210 potential features
  - $\mathbf{x}_j$, ε all drawn from $\mathrm{N}(0, 1)$
  - $y_i = \mathbf{x}_{i1} + \mathbf{x}_{i4} - \mathbf{x}_{i1}\mathbf{x}_{i2} + \varepsilon_i$
- We'll fit both an ordinary lasso and this latent variable group lasso model (this is implemented in the `R` package `glinternet`), using cross-validation to select λ for both models

Results: Lasso

- The ordinary lasso model selects 13 variables: 2 main effects and 11 interaction terms
- Of note, it does select all the true effects
- However, of the 11 interaction terms, none of them have both main effects selected
- The maximum cross-validated R^2 achieved by the model is 0.793

Results: `glinternet`

- The latent variable group lasso approach selects 8 variables: 6 main effects and 2 interaction terms (including the true interaction)
- By construction, both main effects (e.g., $\mathbf{x}_1$ and $\mathbf{x}_2$) are included for each selected interaction ($\mathbf{x}_{1:2}$ and $\mathbf{x}_{8:12}$)
- Both models have essentially the same R^2 (0.793 versus 0.789), although the hierarchy-respecting **glinternet** is more sparse and easier to interpret

Computational challenges

- An important consideration for all interaction methods is that in high dimensions, even *constructing* a matrix with all the interactions is expensive (both to calculate and to store in memory)
- For example, if $p = 20,000$, we would need to store a matrix $\mathbf{X}$ that contains 200 billion interactions
- For this reason, **glinternet** does not actually calculate and store the full $\mathbf{X}$ matrix with all interactions present

Group lasso KKT conditions

- Instead, it relies on the group lasso KKT conditions:

$$\|\tfrac{1}{n}\mathbf{X}_j^\top(\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}})\|_2 = \lambda_j \qquad \widehat{\boldsymbol{\beta}}_j \neq \mathbf{0}$$
$$\|\tfrac{1}{n}\mathbf{X}_j^\top(\mathbf{y}-\mathbf{X}\widehat{\boldsymbol{\beta}})\|_2 \leq \lambda_j \qquad \widehat{\boldsymbol{\beta}}_j = \mathbf{0}$$

- Note that we don't necessarily need to store all the $\mathbf{X}_j$ matrices or include them in a CD algorithm; we can include *some* of them but check the KKT conditions at the end to make sure we didn't miss anything
- This is what **glinternet** does: store $\mathbf{X}_j$ only when it enters the model
- Note: this idea is useful outside of interaction modeling as well, where it is known as *active-set cycling*

sprintr

- The same basic idea appears in an alternative approach known as *reluctant interaction modeling*, which is implemented in an R package called **sprintr** (sparse reluctant interaction modeling in R)
- Unlike **glinternet**, **sprintr** abandons the hierarchy principle and instead tries to encourage main effects by giving them priority in the model fitting process

Reluctant interaction modeling

Specifically, letting $\mathbf{X}$ denote the matrix of main effects and $\mathbf{Z}$ the matrix of interactions, reluctant interaction modeling proceeds as follows:

(1) Fit a lasso model using only $\mathbf{X}$ and calculate the residuals $\mathbf{r}$

(2) Screen the interactions $\mathbf{Z}$ for correlation with $\mathbf{r}$; note any interactions above a specified threshold τ:

$$\frac{1}{n}\left|\mathbf{z}_j^\top \mathbf{r}\right| > \tau$$

(3) Fit a lasso model using both $\mathbf{X}$ and $\mathbf{Z}_\mathcal{S}$, where $\mathbf{Z}_\mathcal{S}$ represents the subset of interactions selected in step (2)

Details

- In practice, rather than specify a cutoff τ, the **sprintr** package selects the top m interactions
- This can be specified by the user, although by default, the number of candidate interactions is set to $n/log(n)$
- Simulations suggest that using cross-validation to select this number has a relatively insignificant effect, and is not worth the computational burden
- Nevertheless, we must still carry out cross-validation over a two-dimensional grid of λ values: the value of λ in step 1 and the value of λ in step 3

Usage of the package

- Using the package is straightforward:

```
cv_sprint <- cv.sprinter(x, y)
```

- With our simulated data, **sprintr** selected 4 coefficients: the two true main effects, the true interaction, and one spurious interaction
- R^2 is comparable to the other two approaches (0.774), and execution is faster than **glinternet** (0.238 versus 1.344 seconds, although a higher-dimensional benchmark would be more relevant)

Modifying variables

- Instead of finding pairwise interactions among a set of features, a different sort of interaction problem is present when we have p potential predictors and a small set of q effect modifiers
- For example, an effect modifier might be age, sex, or treatment; in the specific context of genetics, these are known as *gene-by-environment* interactions

Varying coefficient model

- Consider the following varying-coefficient model

$$y = \beta_0 + \mathbf{z}^\top \boldsymbol{\theta}_0 + \mathbf{x}^\top \boldsymbol{\beta} + \sum_{j=1}^{p} x_j \cdot \mathbf{z}^\top \boldsymbol{\theta}_j + \varepsilon$$

- In words:
  - Each feature x_j has a base coefficient β_j
  - But it also has an interaction term based on a linear combination of effect modifiers
  - So the effective coefficient for x_j is $\beta_j + \mathbf{z}^\top \boldsymbol{\theta}_j$

Context-dependent effects

- In particular, if $\boldsymbol{\theta}_j = \mathbf{0}$, then the effect modifiers don't affect feature j
- Note that feature j may still have an impact in this scenario, but that impact is the same regardless of age, sex, etc.
- On the other hand, when $\boldsymbol{\theta}_j \neq \mathbf{0}$, the feature has a context-dependent effect: the effect of the gene is different for treated and untreated patients, or for smokers and nonsmokers, etc.

Pliable lasso

- To estimate $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, Tibshirani and Friedman (2020) proposed the *pliable lasso*, which imposes the penalty

$$\alpha\lambda \sum_{j,k} |\theta_{j,k}| + (1-\alpha)\lambda \sum_{j=1}^{p} \{\|(\beta_j, \boldsymbol{\theta}_j)\|_2 + \|\boldsymbol{\theta}_j\|_2\}$$

- The second term of the penalty is designed to enforce the weak hierarchy constraint that $\boldsymbol{\theta}_j$ can be nonzero only if β_j is nonzero
- Note that as $\alpha \to 1$, all the interactions are penalized to zero and the solutions approach that of the regular lasso

Pollution data

- As an example application, the authors used the pliable lasso to model air pollution (PM2.5 concentration) in five Chinese cities over 5 years
- The predictors included things like humidity, wind speed, dew point, time of year, etc. (29 in all)
- The effect modifier was location: specifically, $\mathbf{z}$ is a length 5 vector indicating the city

Results

Table 2. Pollution data: directions of the estimated effects for the fitted pliable lasso model.

	Beijing	Chengdu	Guangzhou	Shanghai	Shenyang
Humidity	⇑			↓	
NW wind	↓		↑	↑	↓

NOTE: A double arrow indicates a stronger effect.

One possible explanation for the varying effect of humidity is air turnover: high in Shanghai, a coastal city; low in Beijing, an inland city bordered by mountains that trap air masses