Supplemental Material for:

Fine-mapping additive and dominant SNP effects using group-LASSO and Fractional Resample Model Averaging

Jeremy Sabourin, Andrew B. Nobel, and William Valdar

September 25, 2014

Contents

1 Empirical tuning of the randomized penalties

1.1 Introduction

In the main manuscript we consider the use of resample model averaging under two types of penalties: a standard penalty, i.e. under constant penalization; and a 'randomized' penalty, i.e. under random perturbation of the predictors' penalization. The purpose of the randomized penalty is to perturb the level of penalization on individual predictors to address the 'instability' of the LASSO or group LASSO with highly correlated predictors. This was originally proposed by Meinshausen and Bühlmann (2010). In their "randomized LASSO", the penalty applied to each predictor in each resample is randomly chosen from $\{\lambda, \lambda/c\}$, for some pre-specified $c \in [0, 1]$ (see also Bühlmann and van de Geer 2011). In our manuscript, we extend this to the group LASSO, proposing a "randomized group LASSO" in which SNP-specific weights $\mathbf{r}^{(k)} = (r_1^{(k)}, \ldots, r_m^{(k)})$ are drawn independently in each resample as $r_j = c^{z_j}$ where $z_j \overset{\text{iid}}{\sim} \text{Bin}(1, p_c)$.

The degree of perturbation is controlled by the randomization parameter c, and the frequency at which we apply this up-weighted penalization is p_c. Meinshausen and Bühlmann (2010) used $p_c = 0.5$ and advocated choosing $c \in [0.2, 0.8]$, stating that there was little change in the performance of the procedure within this region. Although our evaluations based on the full AUC are consistent with their findings, we found that the performance based on initial AUC differs based on the value of c. In this Supplemental section, we further examine how choices of c and p_c affect performance, seeking to find values that provide near optimal performance based on AUC and that are also consistent among randomized procedures.

1.2 Results and Discussion

To examine how the choice of the randomized penalty parameters affect the performance of our method, we re-analyzed 100 simulations from setting G using LLARRMA-ras, i.e., standard LLARRMA using the randomized LASSO penalty, and LLARRMA-rdawg for a grid of values of both c and p_c.

Figure 1 displays the initial and full AUCs from LLARRMA-ras, i.e., resample model averaging using the randomized LASSO. Examining the full AUC, we observe the same findings as Meinshausen and Bühlmann (2010), whereby the value of c does not play a large role in overall performance. A more significant role, however, is played by p_c: Specifically, examining the initial AUCs, we observe that the choice of c is more influential in initial AUC than in the full AUC.

Figure 2 displays the initial and full AUCs from LLARRMA-rdawg, i.e., FReMA using the randomized group LASSO. We observe a similar trend in the performance of the randomized group LASSO as for the

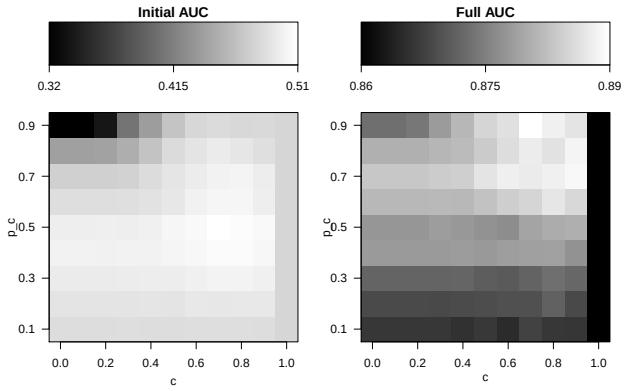

Figure 1: Heatmap of initial and full AUC for LLARRMA-ras on simulation 3B.

randomized LASSO. We observe that under both settings, we obtain the best initial AUC for parameters around the values of $c = 0.7$ and $p_c = 0.5$.

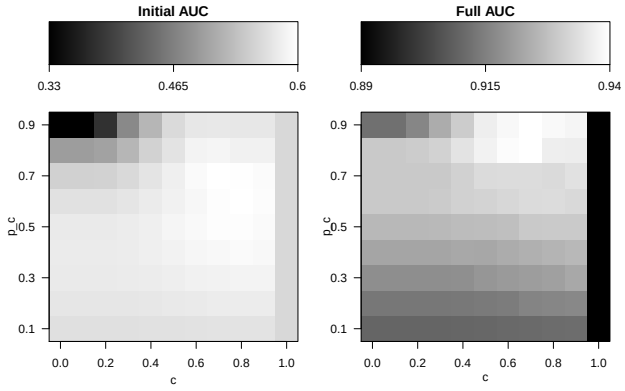

Figure 2: Heatmap of initial and full AUC for LLARRMA-rdawg on simulation 3B.

We also found that the choice of randomization parameter c depends on the data's correlation structure; how the random perturbation of predictor penalties effects a method may depend on the relationship between the variables (i.e. correlation or LD). In our simulations of loci based on real data, $c \in [0.6, 0.8]$ performed optimally. In particular, the combination $c = 0.7$ and $p_c = 0.5$ when $K \geq 250$ performs well under all settings considered here. More generally, however, and especially for very different correlation structures, we recommend dataset-specific calibration.

2 Implementation and efficiency of randomized penalization

In Eq. (3) of the main text, we described the randomized group LASSO as a weighted penalty of the form

$$\text{pen}(\boldsymbol{\beta}; \mathbf{r}) \;=\; \sum_{j=1}^{m} r_j^{-1} \sqrt{\beta_{a,j}^2 + \beta_{d,j}^2}\,, \tag{1}$$

where $r_1, \ldots, r_m \in (0, \infty)$ are the randomized SNP-specific weights. The additional weight r_j^{-1} in the penalty that differentiates the group LASSO and the randomized group LASSO can be achieved in multiple ways. Here we describe two approaches: an indirect approach that is useable in any group LASSO package that that incurs minor additional CPU and memory costs; and a direct approach that removes the additional computational costs but is not available in all group LASSO packages. We also briefly compare the two implementation's CPU times.

2.1 Implementations of the randomized group LASSO

2.1.1 Indirect method The randomized group LASSO (and randomized LASSO) can be implemented by rescaling the data matrix prior to passing it to the fitting procedure. Specifically, if all variables that belong to group j (i.e., a_j and d_j) are scaled by constant r_j, the result of this is the estimated coefficients being multiplied constant r_j^{-1} (i.e., $\hat{\beta}_{a,j}$ rescales to $r_j^{-1}\hat{\beta}_{a,j}$ and $\hat{\beta}_{d,j}$ rescales to $r_j^{-1}\hat{\beta}_{d,j}$). Plugging in the resulting $\boldsymbol{\beta}$ from the rescaled data into Eq. (3) of the main paper with no weights, we have

$$\text{pen}(\boldsymbol{\beta}; \mathbf{1}) \;=\; \sum_{j=1}^{m} \sqrt{(r_j^{-1}\beta_{a,j})^2 + (r_j^{-1}\beta_{d,j})^2} \;=\; \sum_{j=1}^{m} r_j^{-1} \sqrt{\beta_{a,j}^2 + \beta_{d,j}^2}\,. \tag{2}$$

As rescaling the data has no impact on the likelihood, we have achieved the desired randomized group LASSO fit.

This approach involves creation and storage a rescaled version of the data matrix on each resample, and so is likely to result in only a minor computational burden for a typical sized GWAS locus (as we observed in our simulations); but this overhead scales with the size of the data set and so can become appreciable with larger data sets. We also note that when using this indirect method for randomized penalization, it is required that the fitting package does not internally standardize the variables, or has an option to not standardize the data matrix, as this will remove the r_j^{-1} factor in the penalty.

2.1.2 Direct method The randomized group LASSO (and randomized LASSO) can be implemented directly in packages that allow for the user to specify group (or variable) specific penalty weights. Under this setting, the penalty weights $\mathbf{r}^{-1}$ are directly used in the optimization. This removes the need for any overhead computation that was used by the indirect method. Using this approach the randomized group LASSO (or randomized LASSO) incurs minimal computational cost.

Although in most LASSO packages penalty weights are standard, for group LASSO they are not as yet. In situations where fitting packages lack the option to do this direct method, one must use the indirect method.

2.2 CPU times

To compare the relative efficiencies of the randomized versions of LLARRMA considered in the main paper, in Table 1 we report the mean and standard deviation of the CPU time in minutes of each version. For the LASSO, we include both the direct and indirect implementations of the randomized LASSO, but for the group LASSO we are unable to include the direct version as the r/grplasso package does not have the necessary options. We observe that the direct randomized implementation has nearly identical CPU time as their standard penalization counterpart and that the increase in CPU time for the the indirect method is minor. For additive only models, the indirect method typically added about 30 seconds to the CPU time and models with additive and dominance components added about one minute. These time are minor, and consistent with what we would expect for rescaling the data matrix 250 times.

	Type of Penalization		
LLARRMA	Standard	Randomized	
Model		direct	indirect
-as	1.594(0.184)	1.431(0.201)	2.099(0.242)
-das	3.514(0.428)	3.46(0.489)	5.286(0.54)
-aw	2.432(0.393)	2.41(0.386)	3.282(0.431)
-daw	6.295(0.911)	6.168(0.828)	9.238(0.922)
-dawg	16.372(2.212)	–	17.343(1.471)

Table 1: Mean and standard deviations of CPU time (in minutes) based on 500 simulations from setting G using K=250 resamples.

Even without the ability to implement the randomized penalty directly, it can be accomplished indirectly with small overhead. For most GWAS loci, it is unlikely that the use of the randomized penalization will result in any significant increase of CPU time, but for very large data sets or more complex models that

require more model predictors per SNP, the indirect implementation could start to add a more noticeable amount of CPU time to the analysis. Our initial release of R/FReMA uses an indirect implementation of the randomized group LASSO, but in future releases we would like to find a suitable alternative that has all needed options for LLARRMA-rdawg without needing any indirect options that increase CPU time, even if it is only a minor increase.

3 Dominance Model Parameterization

Our procedure models SNPs effects using predictors that would be obtained based on orthogonal contrasts of the three genotype states into additive and dominance components (hereafter, the $A + D$ parameterization). Alternatively, one may select to model directly on an ANOVA-style genotype state model, as in for example Yang et al. (2010). In a purely likelihood based fit these two model alternatives are equivalent, but under penalized estimation, when coefficient values are estimated subject to a penalty, the two models are distinct and can give different results.

To demonstrate this difference we conducted an additional 500 simulations from each SNP effect model and analyzed them with LLARRMA-dawg using the two different parameterizations. Table 2 displays the initial AUC of both models. We observe that in all but one setting the the $A + D$ contrast based penalty model outperforms the ANOVA-style model. Under SNP effect architectures where only one genotype level contains signal (study C and D) we observe similar performance between the models, but across all architectures where signal is present at more than one genotype level the $A + D$ model is far superior.

Locus effect architecture	Parametrization		p-value of
(simulation study)	$A + D$	ANOVA-style	difference
Additive (A)	**0.392**	0.288	<2e-16
Minor Dom. (B)	**0.429**	0.306	<2e-16
Major Dom. (C)	0.344	**0.358**	.001
Heterosis (D)	**0.475**	0.462	.01
General Dom. (E)	**0.385**	0.336	<2e-16
Mostly add. (F)	**0.399**	0.311	<2e-16
Mostly non-add. (G)	**0.385**	0.336	<2e-16

Table 2: Initial AUC, expressed as a percentage of its theoretical maximum, for all simulation studies. For each simulation study, the best performing model is bolded. P-values are based on a paired t-test for significance between initial AUCs given by the two parameterizations.

References

Bühlmann P, van de Geer S. 2011. Statistics for High-Dimensional Data: Methods, Theory and Applications. Heidelberg: Springer.

Meinshausen N, Bühlmann P. 2010. Stability selection. J Roy Stat Soc B 72:417–473.

Yang C, Wan X, Yang Q, Xue H, Yu W. 2010. Identifying main effects and epistatic interactions from large-scale SNP data via adaptive group Lasso. BMC Bioinformatics 11 Suppl 1:S18. doi:10.1186/1471-2105-11-S1-S18.