Pretraining and the Lasso

Erin Craig[1] Mert Pilanci[2] Thomas Le Menestrel[3]
Balasubramanian Narasimhan[4] Manuel A. Rivas[1]
Stein-Erik Gullaksen[6,7] Roozbeh Dehghannasiri[1]
Julia Salzman[1,5] Jonathan Taylor[4] Robert Tibshirani[1,4]

[1]Department of Biomedical Data Science, Stanford University, Stanford CA, USA
[2]Department of Electrical Engineering, Stanford University, Stanford CA, USA
[3] Institute for Computational and Mathematical Engineering, Stanford University, Stanford CA, USA
[4]Department of Statistics, Stanford University, Stanford CA, USA
[5] Department of Biochemistry, Stanford University, Stanford CA, USA
[6] Department of Medicine, Hematology Section, Haukeland University Hospital, Bergen, Norway
[7] K.G. Jebsen Centre for Myeloid Blood Cancer, Department of Clinical Science, University of Bergen, Bergen, Norway

Abstract

Pretraining is a popular and powerful paradigm in machine learning to pass information from one model to another. As an example, suppose one has a modest-sized dataset of images of cats and dogs, and plans to fit a deep neural network to classify them from the pixel features. With pretraining, we start with a neural network trained on a large corpus of images, consisting of not just cats and dogs but hundreds of other image types. Then we fix all of the network weights except for the top layer(s) (which makes the final classification) and train (or "fine tune") those weights on our dataset. This often results in dramatically better performance than the network trained solely on our smaller dataset. In this paper, we ask the question "Can pretraining help the lasso?". We develop a framework for the lasso in which a model is fit to a large dataset, and then fine-tuned using a smaller dataset. This latter dataset can be a subset of the original dataset, or it can be a dataset with a different but related outcome. This framework has a wide variety

of applications, including stratified models, multinomial responses, multi-response models, conditional average treatment estimation and even gradient boosting. In the stratified model setting, the pretrained lasso pipeline estimates the coefficients common to all groups at the first stage, and then group-specific coefficients at the second "fine-tuning" stage. We show that under appropriate assumptions, the support recovery rate of the common coefficients is superior to that of the usual lasso trained only on individual groups. This separate identification of common and individual coefficients can also be useful for scientific understanding.

Keywords: Pretraining, Transfer learning, Supervised learning, Lasso

1 Introduction

Pretraining is a popular and powerful tool in machine learning. As an example, suppose you want to build a neural net classifier to discriminate between images of cats and dogs, and suppose you have a labelled training set of say 500 images. You could train your model on this dataset, but a more effective approach is to start with a neural net trained on a much larger corpus of images, for example IMAGENET (1000 object classes and 1,281,167 training images). The weights in this fitted network are then fixed, except for the top layers which make the final classification of dogs vs cats; finally, the weights in the top layers are refitted using our training set of 500 images. This approach is effective because the network pretrained on a large corpus can discover potentially predictive features for our discrimination problem. This paper asks: is there a version of pretraining for the lasso? We propose such a framework.

This work was motivated by a study carried out with Genentech [McGough et al., 2023]. The authors curated a pancancer dataset, consisting of 10 groups of patients with different cancers. Some of the cancer classes are large (e.g. breast, lung) and some are smaller (e.g. head and neck). The goal is to predict survival times from a large number of features (labs, genetics, ...), approximately 2,000 in total. They compare two approaches: (a) a "pancancer model", in which a single model is fit to the training set and used to make predictions for all cancer classes and (b) separate (class specific) models, each used to make predictions for one class. The authors found that the two approaches produced very similar results, with the pancancer model offering a small advantage in test set C-

index for the smaller classes (e.g. head and neck cancer). Presumably this occurs because of the insufficient sample size for fitting a separate head and neck cancer model, so that "borrowing strength" across a set of different cancers can be helpful.

This led us to consider a framework where the pancancer model is adaptively blended with individual models, allowing each to "learn" from the pancancer model while identifying effects specific to each group. *Importantly, this framework is not specific to grouped data; rather it can be applied to any setting where we wish to share information from one model to another.* This paradigm is somewhat closely related to the ML pretraining mentioned above. It also has similarities to *transfer learning*.

This paper is organized as follows. In Section 2 we review the lasso, describe the pretrained lasso, and show an example with real data. Section 3 discusses related work. In Section 4 we demonstrate the generality of the idea, detailing a number of different "use cases" in Sections 4.1 – 4.6. We move beyond linear models in Section 4.7, illustrating an application of the pretrained lasso to gradient boosting. Real data examples are shown throughout the paper, including applications to cancer, genomics, and chemometrics. Section 5 establishes some theoretical results for the pretrained lasso. In particular we show that under the "shared/ individual model" discussed earlier, the new procedure enjoys improved rates of support recovery, as compared to the usual lasso. We end with a discussion in Section 6.

2 Pretraining the lasso

2.1 Review of the lasso

For the Gaussian family with data $(x_i, y_i), i = 1, 2, \ldots n$, the lasso has the form

$$\text{argmin}_{\beta_0,\beta} \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{1}$$

Varying the regularization parameter $\lambda \geq 0$ yields a path of solutions: an optimal value $\hat{\lambda}$ is usually chosen by cross-validation, using for example the `cv.glmnet` function in the R language package `glmnet` [Friedman et al., 2010].

Before presenting our proposal, two more background facts are needed. In GLMs and

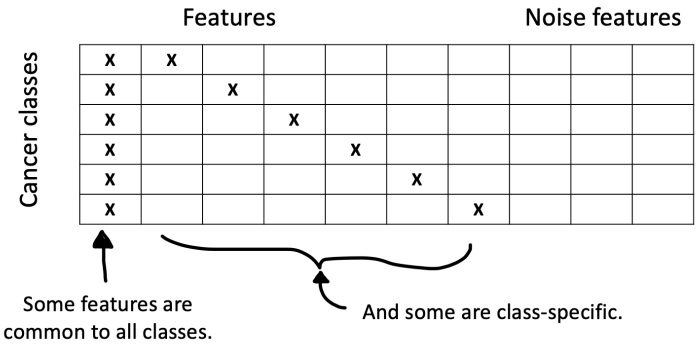

Figure 1: *Conceptual model: some features are predictive for all or most classes, some are specific to each class, and some are noise.*

ℓ_1- regularized GLMs, one can include an *offset*: this is a pre-specified n-vector that is included as an additional column to the feature matrix, but whose weight β_j is fixed at 1. Secondly, one can generalize the ℓ_1 norm $\sum_j |\beta_j|$ to a *weighted* norm $\sum_j \text{pf}_j|\beta_j|$ where each $\text{pf}_j \geq 0$ is a *penalty factor* for feature j. At the extremes, a penalty factor of zero implies no penalty and means that the feature will always be included in the model; a penalty factor of $+\infty$ leads to that feature being discarded.

2.2 The algorithm

Suppose we express our data as an $N \times p$ feature matrix X and a target N-vector y, and we want to do supervised learning via the lasso. In the training set, suppose further that each observation falls in one of K pre-specified classes, and therefore the rows of our data are partitioned into groups $X_1, \ldots, X_K$ and $y_1, \ldots, y_K$.

As shown in Figure 1, we imagine that the features are roughly divided into three types: *common features* that are predictive in most or all classes, *individual features*, predictive in one particular class and *noise features* with little or no predictive power. Note that the common features may have different effect sizes across classes. Our proposal for this problem is a two-step procedure, with the first step aimed at discovering the common features and the second step focused on recovery of the individual features.

For simplicity, we assume here that y is a Gaussian response (y can also be any member of the GLM family, such as binomial, multinomial, or Cox survival). Our model has overall mean and slope components μ_0 and β_0, and class-wise means and slopes: $\mu_k, \beta_k, k = 1, 2, \ldots K$.

$$y_k = (\mu_0 + \mu_k) + X_k(\beta_0 + \beta_k) + \varepsilon_k \text{ for, } k = 1, 2, \ldots K, \tag{2}$$

where (X_k, y_k) is the subset of observations in group k. Note that β_0 is shared across all classes k; this is intended to capture the common features. Then β_k captures features that are unique to each class, and may additionally adjust the coefficient values in β_0. Note that the parameters are not all identifiable as stated: but as we will see, we use an ℓ_1 penalty in its estimation, which makes them identifiable.

We fit this model in two steps, the first of which is aimed at discovering β_0, the coefficients shared across groups. We train an *overall* model using all the data:

$$\hat{\mu}_0, \hat{\beta}_0 = \underset{\mu,\beta}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{K} \| y_k - (\mu \mathbf{1} + X_k \beta) \|_2^2 + \lambda ||\beta||_1, \tag{3}$$

for some choice of λ (e.g the value minimizing the CV error). Define $S(\hat{\beta}_0)$ to be the support set (the nonzero coefficients) of $\hat{\beta}_0$.

Now for each group k, we fit a *class specific* model aimed at discovering class specific features β_k: we find $\hat{\beta}_k$ and $\hat{\mu}_k$ such that

$$\begin{aligned} \hat{\mu}_k, \hat{\beta}_k = \underset{\mu,\beta}{\operatorname{argmin}} \frac{1}{2} \| y_k - (1-\alpha) \left(\hat{\mu}_0 \mathbf{1} + X_k \hat{\beta}_0 \right) - (\mu \mathbf{1} + X_k \beta) \|_2^2 + \\ \lambda \sum_{j=1}^{p} \left[I(j \in S(\hat{\beta}_0)) + \frac{1}{\alpha} I(j \notin S(\hat{\beta}_0)) \right] |\beta_j|. \end{aligned} \tag{4}$$

We choose λ through cross-validation, and $\alpha \in [0, 1]$ is a hyperparameter. The class specific models (with the offset from the overall model) are then used for prediction in each group.

When $\alpha = 0$, this very nearly returns the overall model, and when $\alpha = 1$ this is equivalent to fitting a class specific model for each class. This property is the result of the inclusion of two terms that interact with α.

First, the offset $(1-\alpha) \left(\hat{\mu}_0 \mathbf{1} + X_k \hat{\beta}_0 \right)$ in the loss determines how much the prediction from the overall model influences the class specific models. When the response is Gaussian, using this term is the same as fitting a residual: the target is $y_k - (1-\alpha) \left(\hat{\mu}_0 \mathbf{1} + X_k \hat{\beta}_0 \right)$. That is, the class specific model can only find signal that was left over after taking out the overall model's contribution. When $\alpha = 0$, the class specific model is forced to use

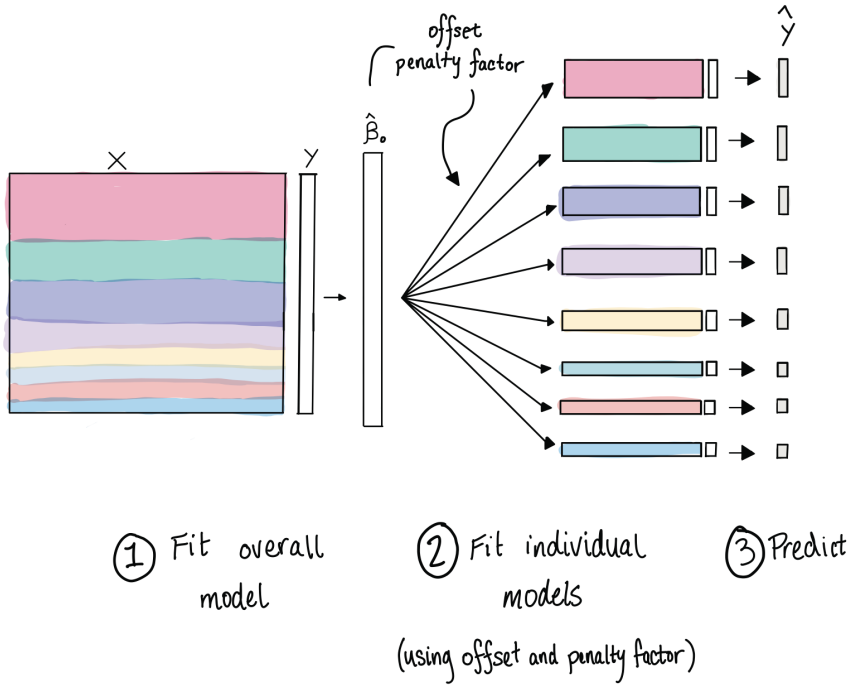

Figure 2: *Workflow for the pretrained lasso, as applied to grouped data.*

the overall model, and when $\alpha = 1$, the overall model is ignored.

Second, the usual lasso penalty is modified by a penalty factor that is 1 for β_j in the support of the overall model $S(\hat{\beta}_0)$ and $\frac{1}{\alpha}$ off the support. When $\alpha = 0$, the class specific model is only able to use features in the support of the overall model: the penalty factor is ∞ off the support. When $\alpha = 1$, the penalty factor is 1 everywhere, and all variables are penalized equally as in the usual lasso.

Remark 1. In our numerical experiments and theoretical analysis (Section 5), we find that the transmission of both ingredients— the offset and penalty factor— are important for the success of the method. The offset captures the first stage model, while the penalty factor captures its support.

The pretrained lasso algorithm is summarized in Algorithm 1. For clarity we express the computation in terms of the R language package `glmnet`, although in principal this could be any package for fitting ℓ_1-regularized generalized linear models and the Cox survival model. A roadmap of the procedure is in Figure 2. As we will describe in Section 4, our proposed paradigm is far more general: we first describe the simplest case for ease of exposition.

We again note that using $\alpha = 0$ is similar to using the overall model for each class: it uses the same support set, but "fine-tunes" the weights (coefficients) to better fit the

Algorithm 1 Pretrained lasso with fixed input groups

1. Fit a single ("overall") lasso model to the training set, using for example `cv.glmnet` in the R language. From this, choose a model (weight vector) $\hat{\beta}_0$ along the λ path, using e.g. lambda.min — the value minimizing the CV error.

2. Fix $\alpha \in [0, 1]$. Define the `offset` and `penalty factor` as follows:

   - Define $\texttt{offset} = (1 - \alpha) \cdot (X_k \hat{\beta}_0 + \hat{\mu}_0)$.
   - Let S be the support set of $\hat{\beta}_0$. Define the penalty factor `pf` as $\texttt{pf}_j = I(j \in S) + \frac{1}{\alpha} \cdot I(j \notin S)$.

   For each class $k \in 1, \ldots, K$, fit an individual model using `cv.glmnet` and the `offset` and `penalty.factor`. Use these models for prediction within each group.

specific group. When $\alpha = 1$ the method corresponds to fitting k separate class-specific models.

Remark 2. The forms for the offset and penalty factor were chosen so that the family of models, indexed by α, captures both the individual and overall models at the extr. We have not proven that this formulation is optimal in any sense, and a better form may exist.

Remark 3. We can think of the pretrained lasso as a simple form of a Bayes procedure, in which we pass "prior" information — the offset and penalty factor — from the first stage model to the individual models at the second stage.

2.3 Example: TCGA pancancer dataset

We applied pretraining to the public domain TCGA pancancer dataset [Goldman et al., 2020]. After cleaning and collating the data, we were left with 714 patients and 20,531 gene expression values. The patients fell into one of 3 cancer classes as detailed in Table 1. The outcome was DFI (disease-free interval): there were 160 events. For computational ease, we filtered the genes down to the 2000 genes having the largest absolute Cox PH score, and all methods used this filtered dataset. The data was divided into a training and testing sets with an 80%/20% split. We used cross-validation to select α for each group. Results are in Figure 3.

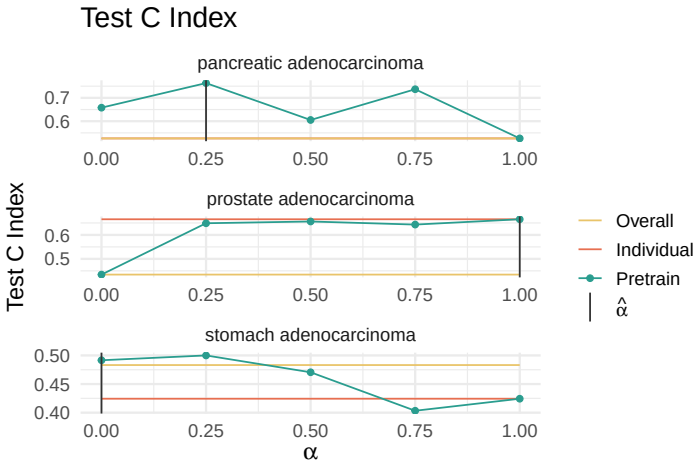

Figure 3: *TCGA dataset: C-index values for different models across 3 cancer classes. The vertical lines indicate the value of $\hat{\alpha}$ chosen by cross-validation for each class.*

Table 1: *Sample size, support size and test C-index for each fitted model. "Total C-index" refers to the C-index computed using the full dataset. Pretraining results use the cross-validated choice of α for each class, as in Figure 3. Pretraining outperforms the overall and individual models, and maintains a small support size.*

	Pancreatic	Prostate	Stomach	Total
Sample size	72	384	258	714
Support size				
Overall	–	–	–	16
Pretrain	22	27	16	33
Individual	1	11	40	52
Test C-Index				
Overall	0.52	0.43	0.48	0.62
Pretrain	**0.76**	**0.67**	**0.49**	**0.69**
Individual	0.53	**0.67**	0.42	0.63

2.4 Pretraining using an external dataset

Here we examine the setting where there is a large external dataset with multiple classes (denoted by D_1), and we have a smaller training set (D_2) from just one class (say class 1). Our goal is to make accurate predictions for class 1. This follows our analogy to using ImageNet (D_1) to train a large neural network, and then to fine tune using a smaller dataset of cats and dogs (D_2). We consider four different approaches to this problem: (1) Fit the lasso with cross-validation (`cv.glmnet`) to D_2; (2) Run the pretrained lasso, using D_1, D_2 for the two stages; (3) Combine the data from D_1 and D_2 for class 1, run `cv.glmnet` on this class 1 data; (4) Combine all of D_1 with D_2, run `cv.glmnet`. This is illustrated in Figure 4.

Note that (1) does not require access to D_1, while (2) requires just the offset and penalty factor from the lasso model fit to D_1. On the other hand, (3) uses class 1 data

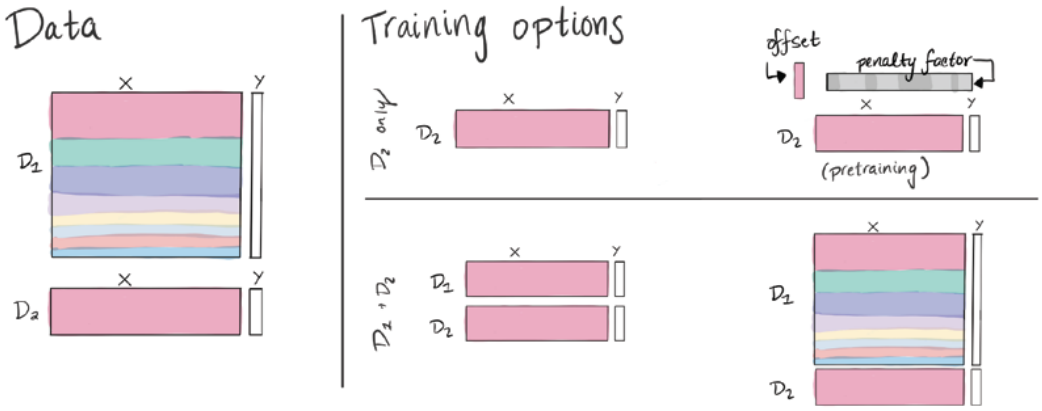

Figure 4: *Options for modeling with an external dataset D_1 and a smaller training dataset D_2. (The bottom row depicts options when D_1 is available at train time.)*

from D_1, while (4) requires *all* of the data D_1. We wish to compare approaches (1) and (2), which do not require access to D_1, with the other two approaches.

We conducted a simulation study to evaluate these four approaches. We simulated a 10-group dataset D_1 with 100 samples per group and 500 features, and an additional dataset D_2 with 100 samples drawn from the same distribution as group 1 in D_1. For each group g, the coefficients β_g were generated as a mixture of shared coefficients and group-specific coefficients, with blending controlled by a parameter ρ:

$$\beta_g = \rho \cdot \beta_{\text{shared}} + (1 - \rho) \cdot \beta_{g,\text{ indiv}},$$

where β_{shared} is identical across groups and $\beta_{g,\text{ indiv}}$ is group specific. The supports for $\beta_{g,\text{ indiv}}$ are non-overlapping across groups. When $\rho = 1$, all groups share the same support and coefficient values; when $\rho = 0$, each group has distinct support (no shared effects). Intermediate values of ρ reflect a mixture of shared and individual effects.

For each simulation, we fit models using all four approaches, and compared performances using prediction squared error (PSE) on a held-out test set. Even when there is no shared support, pretraining using all of D_1 did not hurt. When there is shared support, pretraining performs nearly as well as having extra data from D_1. Performance for $\rho = 0, 0.5, 1$ is shown in Figure 5. In Appendix A.2, we repeat this experiment in the setting where the groups share *support*, but the effect sizes are different across groups.

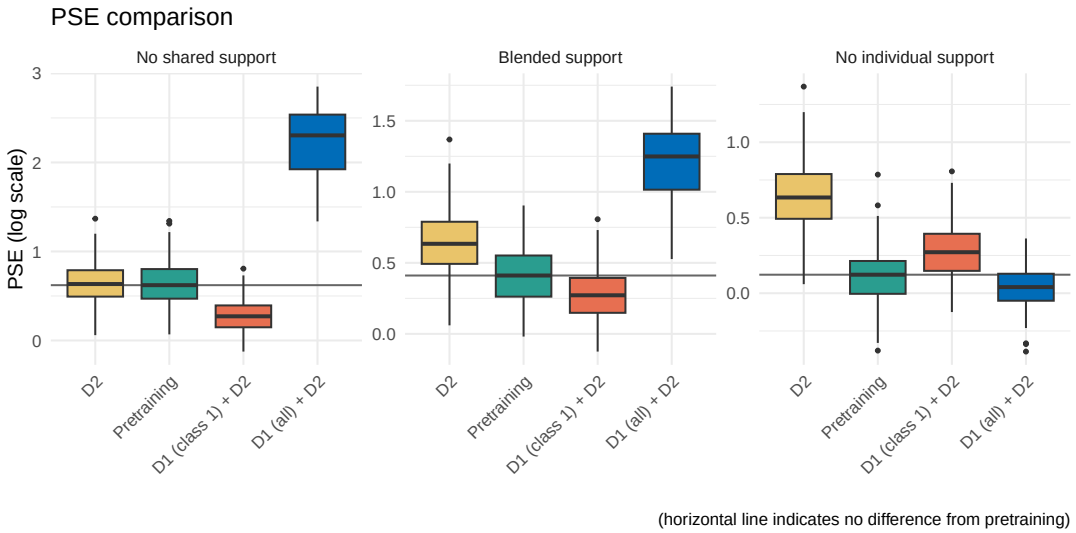

Figure 5: *Comparison of approaches for modeling with a large (usually inaccessible) dataset D_1 and a smaller dataset D_2. When D_1 is informative, pretraining using only D_2 and the coefficients from a model trained with D_1 performs nearly as well as having access to D_1 directly. When D_1 is not informative, pretraining with D_1 does not hurt.*

3 Related work

3.1 Data shared lasso

The data shared lasso [Gross and Tibshirani, 2016] (DSL) is a closely related approach for modeling data with a natural group structure. It solves the problem

$$\left(\hat{\beta}, \hat{\Delta}_1, \ldots, \hat{\Delta}_K\right) = \text{argmin} \frac{1}{2} \sum_i \left(y_i - x_i^T(\beta + \Delta_{k_i})\right)^2 + \lambda \left(\|\beta\|_1 + \sum_{k=1}^{K} r_k \|\Delta_k\|_1 \right). \quad (5)$$

It jointly fits an overall coefficient vector β that is common across all K groups, as well as a modifier vector Δ_k for each group k. The parameter r_k in the penalty term controls the size of $\|\Delta_k\|_1$, and therefore determines whether the solution should be closer to the overall model β (for r_k large) or the individual model $\beta + \Delta_k$ (for r_k small).

DSL is analogous to pretrained lasso in many ways. Both approaches fit overall and individual models, and both have a parameter (r_k or α_k) to balance between the two. One important difference between DSL and pretraining is the use of `penalty.factor` in pretraining. For $0 < \alpha < 1$, pretraining encourages the individual models to use the same features that are used by the overall model, but allows them to have different values. DSL has no such restriction relating β to the modifier Δ. Additionally, because pretraining is performed in two steps, it is flexible: researchers with large datasets can train and share

overall models that others can use to train an individual model with a smaller dataset.

3.2 Laplacian regularized stratified models

Stratified modeling fits a separate model for each group. *Laplacian regularized* stratified modeling [Tuck and Boyd, 2021] incorporates regularization to encourage separate group models to be similar to one another, depending on a user-defined structure indicating similarity between groups. For example, we may expect lymphoma and leukemia to have similar models because they are both blood cancers, and we could pre-specify this when model fitting. While pretrained lasso uses information from an *overall* model, laplacian regularized stratified modeling uses similarities *between groups*.

3.3 Reluctant interaction modeling

Reluctant interaction modeling [Yu et al., 2019] is a method for training a lasso model using both main and interaction effects, while (1) prioritizing main effects and (2) avoiding the computational challenge of training a model using all p^2 interaction terms. It uses three steps: in the first, a model is trained using main effects only. Then, a subset of interaction terms are selected based on their correlation with the *residual* from the first model; the intention is to only consider interactions that may explain the remaining signal. Finally, a model is fit to the residual using the main effects and the selected set of interaction effects. Though it has a different goal than pretraining, Reluctant Interaction Modeling shares high level properties with our current proposal.

3.4 Mixed effects models

Mixed effects models jointly find *fixed* effects (common to all the data) and *random* effects (specific to individual instances). A linear mixed effect model has the form

$$y = X\beta + Z\theta + \varepsilon, \tag{6}$$

where X consists of features shared by all instances and Z consists of features related to individual instances. Both pretrained lasso and mixed effects modeling aim to uncover two components; in pretraining, however $X = Z$ and we seek to divide β into overall and

group-specific components.

4 Pretrained lasso: a wide variety of use cases

We have described the main idea for lasso pretraining, as applied to data with grouped observations: a model is fit on a large set of data, an offset and penalty factor are computed, and these components are passed on to a second stage, where individual models are built for each group. It turns out that the pretraining idea for the lasso is a general paradigm: it is a method for passing information from one model to another, and there are many different ways that it can be applied. Typically the pipeline has only two steps, as in the example above; but in some cases it can consist of multiple steps, as made clear next. Below is a (non-exhaustive) list of potential use cases, *the common feature of which is the passing of an offset and penalty factor from one model to the next.*

1. **Input grouping:**

   The rows of X are partitioned into groups. These groups may be:

   (a) Pre-specified (Section 2.2), e.g. cancer classes, age groups, ancestry groups. The pancancer dataset described above is an example of this use case.

   (b) Pre-specified but different in training and test sets (Section 4.4), e.g. different train and test patients.

   (c) Learned from the data via a decision tree (Section 4.5).

2. **Multinomial response:** Predict class membership with > 2 classes, e.g. predict a disease subtype or a cell type (Section 4.1). We wish to leverage shared signal across classes, and to flexibly identify signal specific to each class.

3. **Multi-response:** Predict a matrix of responses. There are two special cases: time-ordered columns, where the same target is measured at different points in time, and mixed targets, where the target columns are of different types, e.g. quantitative, survival, or binary/multinomial. This is illustrated in Sections 4.2 and 4.3.

4. **Conditional average treatment effect estimation**. This is similar to the input grouping case; groups are defined by the levels of a treatment variable (Section 4.6).

5. **Gradient boosting:** We fit a gradient boosting model, and use the individual trees as features in a pretraining pipeline (Section 4.7).

Of course, other scenarios are possible.

4.1 Multinomial response models

Suppose now that we have no grouping on the rows of X; instead we have K response classes and wish to fit a multinomial model. It is much the same as our earlier algorithm, the only difference is the way in which the models are combined at the end. Algorithm 2 describes the procedure in detail, and this is applied to real data in Section 4.1.1.

Algorithm 2 Pretrained lasso for multinomial responses

1. At the first stage, let $B_{p\times K}$ be the coefficient matrix. Fit a grouped multinomial model to all classes: use two-norm penalties on the rows of B (i.e. $\sum_{j=1}^{p} \|\beta_{j,\cdot}\|_2$).
2. Second stage: for each class k, define the offset equal to the kth column of $(1-\alpha)X\hat{B}$. Define S_k to be the support of the kth column of $\hat{B}$. Use penalty factor $I(j \in S_k) + \frac{1}{\alpha} \cdot I(j \notin S_k)$. Fit a two class model for class k vs the rest using the offset and penalty factor.
3. Classify each observation to the class having the maximum probability across all of the one versus rest problems.

4.1.1 Multinomial response example: classifying cell types

We applied the pretrained lasso together with SPLASH [Chaung et al., 2023, Kokot et al., 2023], a new approach to analyzing genomics sequencing data. SPLASH is a statistics-first alignment-free inferential approach for genomic sequence analysis. SPLASH is applied directly to raw sequencing reads and returns k-mers which show statistical variation across samples. Here we used the output of SPLASH run on 10x muscle cells (2,760 cells from the 10 most common muscle cell types in donor 1) from the Tabula Sapiens consortium [Consortium et al., 2022], a comprehensive human single-cell atlas. SPLASH yielded about 800,000 (sparse) features.

We divided the data into 80% train and 20% test sets so that the distribution across the 10 cell types was roughly the same in training and testing, and we used cross-validation to select the pretraining hyperparameter α. Figure 6 shows results across a range of α

values on held-out data. We measure performance with the test data in two ways: (1) we compute the misclassification rate by treating this as a 10 class classification task and (2) we compute the *sum* of misclassification rates by treating this as a set of 10 one vs. rest problems. Using the $\hat{\alpha}$ selected by cross-validation, pretraining outperforms the overall and individual models by both metrics. As a single 10-class problem, the misclassification rate is 0.220 vs 0.339 or 0.235 for the overall and individual models, and it uses fewer selected features than the individual models (1659 features vs 2110 features).

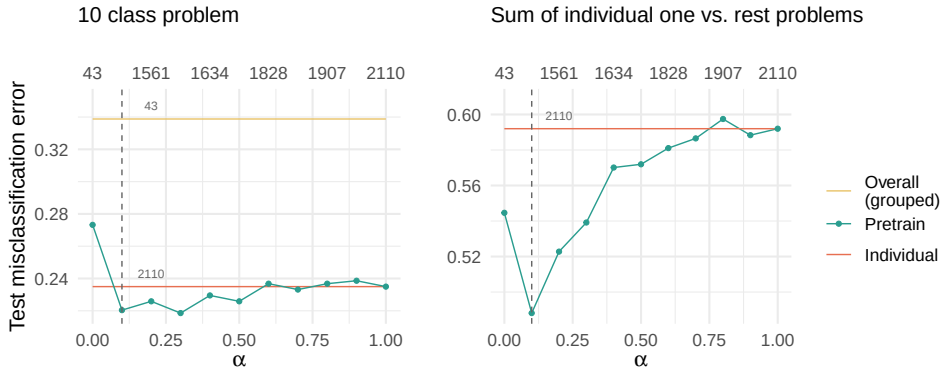

Figure 6: *Left: misclassification rate of the 10 class problem. Right: the sum of misclassification rates across 10 one-vs-rest problems. The vertical dashed line shows the value of the hyperparameter α chosen by cross-validation.*

An important open biological question is to determine which of the features selected by SPLASH are cell-type-specific or predictive of cell type. We tested whether the pretrained lasso could be used to determine which alternative splicing events found by SPLASH were predictive of cell type. Without tuning, the pretrained lasso reidentified cell-type-specific alternative splicing in MYL6, RPS24, and TPM2, all genes with established cell-type-specific alternative splicing [Olivieri et al., 2021]. In addition, SPLASH and the pretrained lasso identified a regulated alternative splicing event in Troponin T (TNNT3) in Stromal fast muscles cells which to our knowledge has not been reported before, though it is known to exhibit functionally important splicing regulation [Schilder et al., 2012]. These results support the precision of SPLASH coupled with the pretrained lasso for single cell alternative splicing analysis.

4.2 Multi-response models

Another interesting use case is the multi-response setting, where the outcome Y has $K > 1$ columns, and the data in these columns may be quantitative or integers. The multinomial

response discussed earlier can be expressed as a multi-response problem corresponding to one-hot encoding of the classes. But the multi-response setup is more general, and can be used in problems where each observation can fall in more than one class.

To apply the pretrained lasso here, we fit a grouped multi-response model (Gaussian or multinomial) to all of the columns, and then fit individual models to each column separately. In the Gaussian case, the first step uses the grouped multi-response loss:

$$\frac{1}{2}\sum_{k=1}^{K}\|y_k - X\beta_{\cdot,k}\|_2^2 + \lambda\sum_{j=1}^{p}\|\beta_{j,\cdot}\|_2, \tag{7}$$

where y_k is the k^{th} response and $\beta_{\cdot,k}$ are the corresponding coefficients. For a particular feature j, the penalty $\|\beta_{j,\cdot}\|_2$ forces $\beta_{j,k}$ to be zero or nonzero for all $k = 1, \ldots, K$. The second step uses pretraining as usual for each response: the penalty factor and offset for the k^{th} response are defined as in Algorithm 1 using the coefficients $\beta_{\cdot,k}$.

4.2.1 Multi-response example: chemometric data

Figure 7 shows an example taken from Skagerberg et al. [1992], simulating the production of low-density polyethylene. The data were generated to show that quality control could be performed using measurements taking during polyethylene production to predict properties of the final polymer. The authors simulated 56 samples with 22 features including temperature and solvent flow-rate, and 6 outcomes: number-average molecular weight, weight-average molecular weight, frequency of short chain branching, the content of vinyl groups and vinylidene groups. Figure 7 shows the LOOCV squared error over the 6 outcomes, using both the best common α (left) and outcome-specific α values (right). Pretrained lasso performs best in both settings.

4.3 Time ordered responses and chaining outcomes

Other multi-response settings include prediction of time-ordered responses, and outcomes of different types (e.g. quantitative, survival, binary). In both cases we apply pretrained lasso in a sequential fashion. We fit a model to the first outcome, compute the offset and penalty factor, and pass these to a model for the second outcome, and so on. This application requires a prior ordering of the outcomes. In real applications, when an

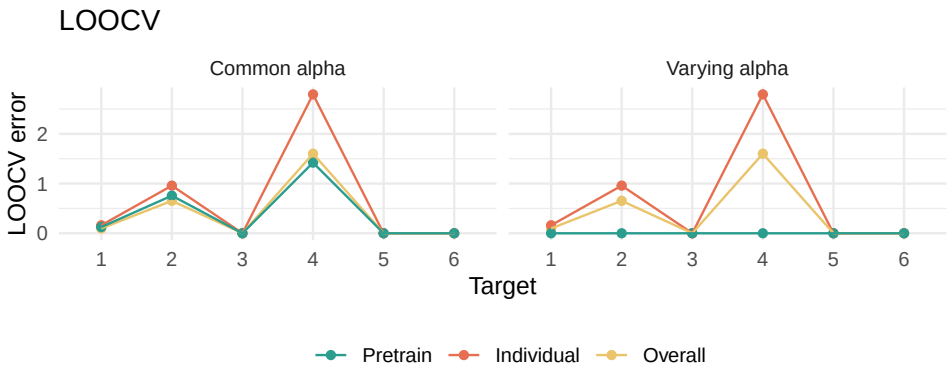

Figure 7: *Results for multi-response chemometrics example.*

ordering is not already defined, it might make sense to place the primary outcome measure in the first position, and the rest in decreasing order of importance.

4.3.1 Time ordered response example: mass cytometry cancer data

We applied pretraining to a dataset of mass cytometry measurements from 8 patients with chronic myeloid leukemia who are treated with the drug nilotinib [Gullaksen et al., 2017]. Our objective is to predict each patient's BCR::ABL1$^{\text{IS}}$ levels at 9 time points over 24 months, as this highly sensitive measure of residual disease is strongly correlated with progression-free survival. Figure 8 shows the results of pretraining applied to this dataset as described for time-ordered responses. For all time points, pretraining nearly matches or outperforms the alternative of fitting separate models.

Figure 8: *Results for the time ordered data example. Cross-validation was used to select the pretraining hyperparameter* $\alpha = 0.5$.

Remark 4. *Pretrained lasso fits an interaction model.* In general, suppose we have a response y, features x and grouping variables $G_1, G_2, \ldots G_g$. The grouping variables can stratify the inputs or the target (either multinomial or multi-response). Introduction of a grouping

variable G_j corresponds to the addition of an interaction term between x and G_j. Thus one could imagine a more general forward stepwise pretraining process as follows:

1. Start with an overall model, predicting y from x, without any consideration of the grouping variables. Let O_1 and pf_1 be the offset and penalty factor from the chosen model.

2. Introduce the grouping variable G_1 by fitting individual models to the levels of G_1, with the offset and penalty factor O_1 and pf_1. From these models extract O_{2k} and pf_{2k} for the K_1 levels of G_1: $k = 1, 2, \ldots K_1$.

3. Introduce the grouping variable G_2, either as an interaction $x \times G_2$ or an interaction $x \times G_1 \times G_2$, and so on.

4.4 Different groupings in the train and test data

In our initial "input grouped" setting, our training data are partitioned into groups, and we observe the same groups at test time. Now, we consider the setting where the test groups were not observed at train time. For example, we may have a training set of *people*, each of whom has many observations, and at test time we wish to make predictions for observations from new people.

To address this, we use pretraining as described. Now, however, we fit an extra classifier to predict the *training group* for each observation. We then train a final model that predicts the response using the output from the pretraining models and the group classifier. The procedure is illustrated in Figure 9, and applied to real data in Section 4.4.1.

Although one could use the estimated posterior class probabilities from this construction, we have found better empirical results by training a supervised learning algorithm to predict $r(x_i)$ from $\hat{p}_k(x_i), \hat{q}_k(x_i)$ and $\hat{p}_k(x_i) \cdot \hat{q}_k(x_i), k = 1, 2, \ldots K$.

4.4.1 Different train/test grouping example: mass spectrometry cancer data

This data, from a melanoma proteomics study [Margulis et al., 2018], has 2094 peak heights measured via DESI mass spectrometry per pixel, with about 1,000 pixels from each patient. There are 28 training patients, 15 test patients; a total of 29,107 training pixels and 20,607 test pixels. The output target is binary (healthy vs disease). All error rates quoted are per pixel rates.

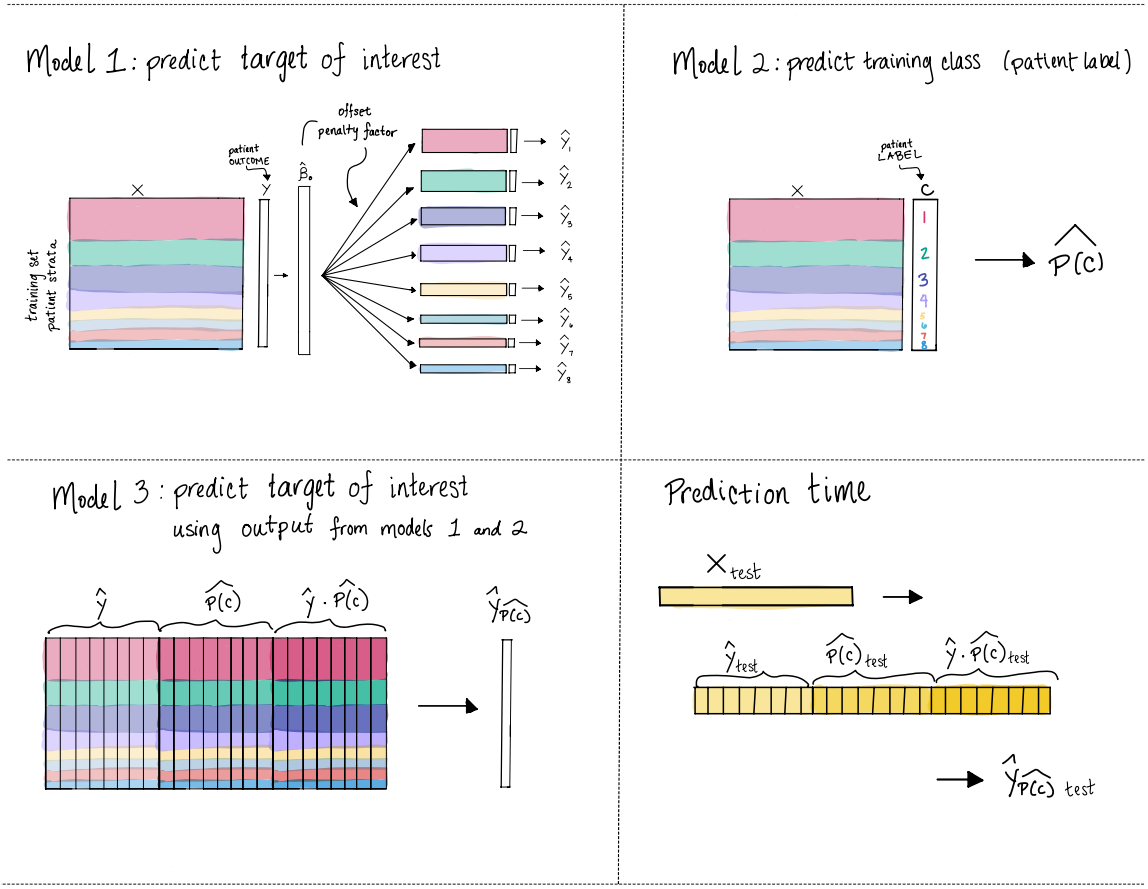

Figure 9: *How to use pretraining when the train and test strata are different (Section 4.4).*

We clustered the training patients using K-means into 4 groups, and then applied pretraining. The pretrained lasso provides an advantage in AUC relative to the overall model: 0.96 for pretraining vs 0.94 without.

4.5 Learning the input groups

Here we consider the setting where there are no fixed input groups, but instead we learn potentially useful input groups from a CART decision tree, and then apply pretraining with these groups. Typically, the features that we use for splitting are not the full set of features x but instead a small set of clinical variables that are meaningful to the scientist.

4.5.1 Learning the input groups example: predicting myocardial infarction

We illustrate this on the U.K. Biobank data, where we have derived 299 features on 64,722 white British individuals, and we wish to predict myocardial infarction. There are 249 metabolites from nuclear magnetic resonance and 50 genomic PCs. The features available for splitting were age, PRS (polygenic risk score) and sex (0=female, 1=male).

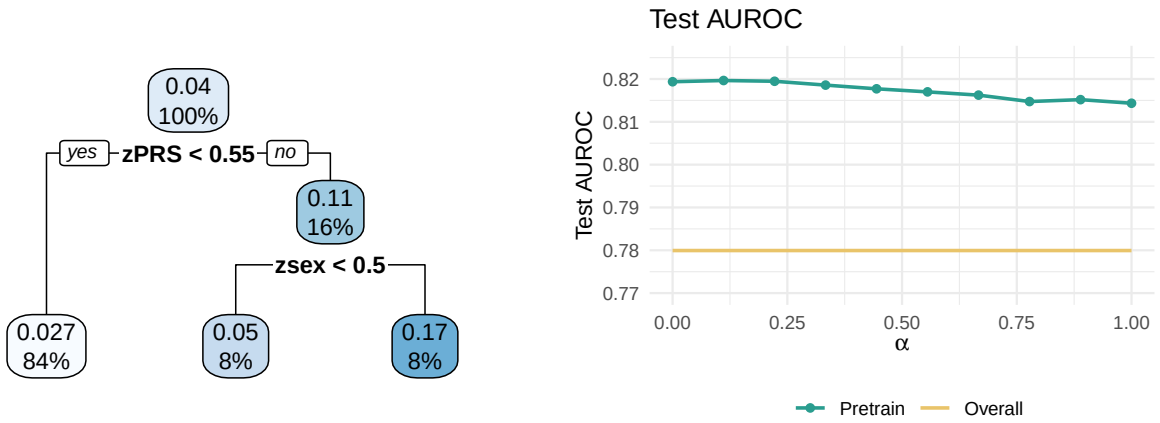

Figure 10: *Left: CART tree learned from the features PRS, Sex and Age. Right: test set AUC for overall (no groups) model and pretrained lasso applied to the 3 groups.*

We used a 50/50 train/test split and built a CART tree using the R package `rpart`, limiting the depth of the tree to be 3 (for illustration)[1]. The left panel of Figure 10 shows the resulting tree. The right-most terminal node contains men with high PRS scores: their risk of MI is much higher than the other two groups (0.17 versus 0.027 and 0.05). The predictions using just this CART tree had a test AUC of 0.49.

We then applied the pretrained lasso for fixed input groups (Algorithm 1) to the three groups defined by the terminal nodes of the tree. The resulting test AUCs for the pretrained lasso and the overall model (an ℓ_1-regularized logistic regression) are shown in the right panel. The pretrained lasso delivered about a 4-5% AUC advantage for all values of α, and it identified a strong interaction of one feature with PRS and sex.

4.6 Conditional average treatment effect estimation

An important problem in causal inference is estimation of the conditional average treatment effect (CATE). The data is of the form $(X_i, W_i, y_i), i = 1, 2, \ldots n$ where X_i is a vector of covariates, Y_i is a quantitative outcome and W_i is a binary treatment indicator. We denote by $(Y_i(0), Y_i(1))$ the corresponding outcomes we would have observed given the treatment assignment $W_i = 0$ or 1 respectively. The goal is to estimate the CATE: $\tau(x) \equiv E(Y(1)|x) - E(Y(0)|x)$. We make the usual assumption that the treatment

[1]Another way to grow the decision tree in this procedure would be to use "Oblique Decision Trees", implemented in the ODRF R language package. These trees fit linear combinations of the features at each split. Since the pretrained lasso fits a linear model (rather than a constant) in each terminal node, this seems natural here. We tried ODT in this example: it produced a similar tree to that from CART, and hence we omit the details. We thank Yu Liu and Yingcun Xia for implementing changes to their R package ODRF so that we could use it in our setting.

assignment is unconfounded.

One popular approach is the "R-learner" of Nie and Wager [2021]. It is based on the objective function

$$\hat{\tau} = \arg\min_{\tau} \frac{1}{n} \sum \Big[(Y_i - m(X_i)) - (W_i - e(X_i)) \cdot \tau(X_i) \Big]^2 \tag{8}$$

where $m(x)$ is the overall mean function and $e(x)$ is the treatment propensity $\Pr(W = 1|X = x)$. In the simplest case (which we focus on here), a linear model is used for $m(x)$ and $\tau(x)$. The lasso version of the R-learner adds an ℓ_1 penalty to the objective function.

The steps of the R-learner are as follows: (1) estimate $m(\cdot), e(\cdot)$ by fitting Y on X, W on X, using cross-fitting; (2) estimate $\tau(\cdot)$ by solving (8) above. For simplicity we assume here that the treatment is randomized so that we can set $e(x) = 0.5$.

Now we can combine the R-learner with the pretrained lasso as follows. We assume the shared support model

$$\hat{Y} = \hat{\beta}_0 + X\hat{\beta} + W \cdot \hat{\tau}(X); \quad \hat{\tau}(X) = X\hat{\theta}_0 + X\hat{\theta} \tag{9}$$

where θ has the same support and signs as β. To fit this, we use the R-learner procedure above, but include in the model for $\tau(X)$ the penalty factor computed from the model for Y [we do not include the offset, since the target in the two models are different]. If this shared support assumption is true or approximately true, we can potentially do a better job at estimating $\tau(x)$. This assumption seems reasonable: it says that the predictive features are likely to overlap with the features that modify the treatment effect.

Figure 11 shows an example with $n = 300$, $p = 20$ and an SNR of about 2. The first 10 components of β are positive, while the second 10 components are zero. In left panel the treatment effect θ has the same support and signs as β, while in the right panel, its support is in the second 10 features, with no overlap with the support of β. The figure shows boxplots of the absolute estimation error in $\tau(x)$ over 20 realizations.

In the left panel we see that the pretrained R-learner outperforms the R-learner for all α, while in the right panel, they behave very similarly. It seems that there is little downside in assuming the shared support model. Upon closer examination, the reason becomes clear: under Model (9) with disjoint support, all 20 features are predictive of the

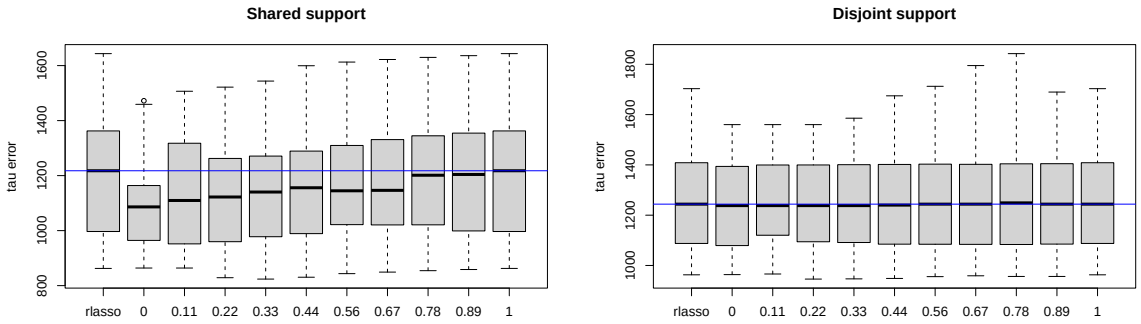

Figure 11: *Results for R-learner experiment. Horizontal blue line shows the median error for the rlasso R-learner.*

outcome, and hence there is no support restriction resulting from the outcome model.

4.7 Beyond linear models: gradient boosting

Here we explore the use of basis functions beyond the linear functions used throughout the paper. Suppose we run gradient boosting [Friedman, 2000] for M steps, giving M trees. Then we can consider the evaluated trees as our new variables, yielding a new set of features. We then apply the pretrained lasso to these new features. Here is the procedure in a little more detail: (1) run M iterations of xgboost to get M trees (basis functions) B $(n \times M)$ and (2) run the pretrained lasso on B.

Consider this procedure in the fixed input groupings use case. We use the lasso to estimate optimal weights for each of the trees, both for an overall model, and for individual group models. For the usual lasso, this kind of "post-fitting" is not new (see e.g. RuleFit [Friedman and Popescu, 2008], [Hastie et al., 2009] page 622).

It is easy to implement this procedure using the `xgboost` library in R [Chen et al., 2023]. Figure 12 shows the results from a simulated example. We first used *xgboost* to generate 50 trees of depth 1 (stumps). Then we simulated data using these trees as features, with a strong common weight vector $\hat{\beta}_0$. The first method in our example, `xgboost`, is boosting applied to the raw features, while the other three methods use the 50 trees generated by `xgboost`. We see that lasso pretraining can help boosting as well.

5 Theoretical results on support recovery

In this section, we prove that the pretraining process recovers the true support and characterize the structure of the learned parameters under suitable assumptions on the

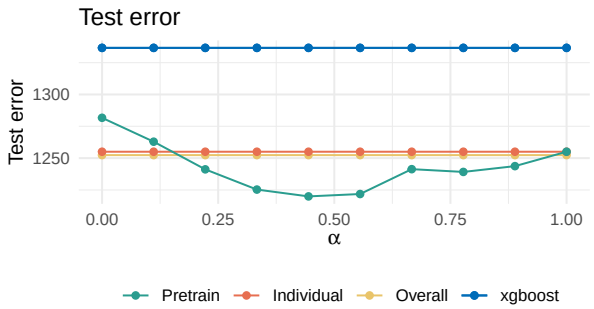

Figure 12: *Results for the pretrained lasso applied to boosted trees. The first method* `xgboost` *uses raw features. The remaining use evaluated trees from* `xgboost` *as features.*

training data.

5.1 Preliminaries

We call a random variable Y sub-Gaussian if it is centered, i.e., $\mathbb{E}[Y] = 0$ and $\mathbb{E}\left[e^{sY}\right] \leq e^{\frac{\sigma^2 s^2}{2}}$ $\forall s \in \mathbb{R}$, where σ^2 is called the variance proxy of Y. For such a random variable, the following tail bound holds: $\mathbb{P}\left[|Y| > t\right] \leq 2e^{-t^2/(2\sigma^2)}$ for all $t > 0$. The random variable Y has bounded variance when σ^2 is bounded by a constant. Examples of such random variables include the standard Gaussian and any random variable that is zero-mean and bounded by a constant.

5.2 An overview of the theoretical results

5.2.1 Conditions for deterministic designs

The recovery of the support of coefficients in lasso models is traditionally guaranteed by conditions like the irrepresentability condition. In this paper, we extend these conditions to the pretrained lasso. Specifically, we introduce a set of deterministic conditions that ensure the recovery of the true support in the shared support model, even when observations are mixed. These conditions, referred to as Pretraining Irrepresentability Conditions, are necessary and sufficient for the pretraining estimator to discard irrelevant variables and recover the true support. Although these conditions are slightly more complex than the classical irrepresentability condition due to the mixed observation model, they are easily interpretable. In summary, there are three key requirements: (1) the off-support features need to be incoherent with the features in the support, (2) the empirical covariance of the features in the support need to be well conditioned, (3) the individual

parameters need to be bounded in magnitude.

5.2.2 Conditions for random designs

Two key aspects are studied when the design matrix is random:

- Pretraining under isometric features: We introduce the subgroup isometry condition (15) to capture how representative the empirical covariance of a subgroup is in relation to the full dataset. This condition holds when the features are independent sub-Gaussian random variables, and helps in analyzing the behavior of the pretraining estimator.

- Recovery under sub-Gaussian covariates: We prove in Theorems 1 and 2 that under certain conditions on the sample size, variable bounds, and noise levels, the pretraining estimator can recover the true support with high probability under the shared support model (10) where the supports are common. These results are similar in spirit to the existing recovery results for lasso [Wainwright, 2009], with a few crucial differences. In particular, it is known in the classical setting that $\mathcal{O}(s\log(p-s))$ measurements are necessary for support recovery with high probability. However, in our setting, our result given in Theorem 1 show that the number of measurements needs to scale as $\mathcal{O}(\max(1,\gamma^2)s\log(p-s))$, where γ is an upper-bound on the magnitude of the weights $|\beta_k|\,\forall k$. The extra $\max(1,\gamma^2)$ factor is due to the mixture observation model (10) instead of a simple linear relation studied in earlier literature. It is an open question to verify that this factor is unavoidable, which we leave for future work.

In addition, we extend the shared support model (10) to lift the assumption that the supports are common, and consider the common and individual support model (20). In this model, there is a shared support between the groups, as well as additional individual supports. We show that support recovery results given in Theorems 1 and 2 still hold under the assumption that the magnitudes β_k^* that belong to the individual support are sufficiently small for each k. This is a necessary condition to ensure that the pretraining estimator only recovers the common support and discards individual supports for each group.

5.3 Shared support model

Consider K sets of observations

$$y_k = X_k \beta_k^* + \varepsilon_k \in \mathbb{R}^{n/K}, \tag{10}$$

where β_k^* are unknown vectors which share a support S of size s. More precisely, we have $(\beta_k^*)_{S^c} = 0 \ \forall k \in [K]$ where S^c is the complement of the subset S. Here, $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_K]$ is a noise vector to account for the measurement errors, which are initially assumed to be deterministic. We assume that $\frac{n}{K}$ is an integer and each subgroup has at least s samples, i.e., $\frac{n}{K} \geq s$. We let $X := \begin{bmatrix} X_1^T & \ldots & X_K^T \end{bmatrix}^T \in \mathbb{R}^{n \times p}$ to denote the full dataset and observations $y := \begin{bmatrix} y_1 & \ldots & y_K \end{bmatrix} \in \mathbb{R}^n$.

We define the pretraining estimator as

$$\hat{\beta}_{\text{pre}} = \arg\min_{\beta} \frac{1}{n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1. \tag{11}$$

Next, we provide an analysis of the pretraining estimator under the shared support model.

5.3.1 Pretraining irrepresentability conditions

We now provide a set of deterministic conditions that guarantee that the pretraining estimator $\hat{\beta}_{\text{pre}}$ recovers the support of $\beta_1, \ldots, \beta_K$. Note that existing results on support recovery for lasso including the irrepresentability condition [Zhao and Yu, 2006], restricted isometry property [Van De Geer and Bühlmann, 2009] or random designs [Wainwright, 2009] are not applicable due to the mixed observation model in (10).

Recall that in the shared support model, the vectors $\beta_1^*, \ldots, \beta_K^*$ have the same support S of size s. Let us use $X_S \in \mathbb{R}^{n \times s}$ to denote the submatrix of the data matrix X restricted to the support S.

Lemma 1 (Pretraining irrepresentability)**.** *Suppose that the conditions*

$$\|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S^*)\|_\infty < \frac{1}{2} \tag{12}$$

$$\|X_{S^c}^T P_S^\perp \sum_{k=1}^{K} D_k X \beta_k^*\|_\infty \leq \frac{\lambda}{4}\,, \tag{13}$$

hold. Then, the pretraining estimator discards the complement of the true support S, i.e., $j \notin S \implies (\hat{\beta}_{\text{pre}})_j = 0 \,\forall j$ in the noiseless case, i.e., $n = 0$. In the noisy case, if the condition

$$\|X_{S^c}^T P_S^\perp \varepsilon\|_\infty \leq \frac{\lambda}{4} \tag{14}$$

holds, the same result holds for an arbitrary noise vector n. Here, $\mathbf{sign}(\beta_S^*) = \mathbf{sign}((\beta_k^*)_S \,\forall k \in [K]$ is the sign of the vectors $\beta_1^*, \ldots, \beta_K^*$ constrained to their support S, $P_S^\perp := I - X_S(X_S^T X_S)^{-1} X_S^T$ and D_k is the diagonal selector matrix for the k-th set of samples, i.e., $D_k X \in \mathbb{R}^{n \times p}$ is $X_k \in \mathbb{R}^{\frac{n}{K} \times p}$ padded with zeros.

Remark 5. When $K = 1$ and $D_1 = I$, the conditions (12) and (13) simplify to the well-known strong irrepresentability condition [Zhao and Yu, 2006], noting that $P_S^\perp D_k X = P_S^\perp X = 0$ which shows (13) always holds.

5.3.2 Pretraining under isometric features

We now analyze the behaviour of the solution $\hat{\beta}_{\text{pre}}$ under the assumptions that the samples from the subgroups follow an isometric distribution relative to the entire dataset.

We introduce the subgroup isometry condition:

$$\text{(subgroup isometry)} \qquad \left\| (X_S^T X_S)^{-1} (X_S^T D_k X_S) - \frac{1}{K} I \right\|_2 \leq \delta \;\forall k \in [K], \tag{15}$$

for some $\delta \in (0, 1)$. The above quantity represents the ratio of empirical covariances of features restricted to the subset S, comparing the entire dataset with the subgroup defined by the k-th group of samples. It quantifies how representative the empirical covariance of the subgroup is in relation to the full dataset.

Remark 6. Note that we have $(X_S^T X_S)^{-1}(X_S^T D_k X_S) = \big(\sum_{i \in [n]} \tilde{x}_i \tilde{x}_i^T\big)^{-1} \big(\sum_{i \in \mathcal{G}_k} \tilde{x}_i \tilde{x}_i^T\big)$, where $\mathcal{G}_k$ is the subset of samples that belong to the group k and $\{\tilde{x}_i\}_{i=1}^n \in \mathbb{R}^s$ are features restricted

to the true support S.

Lemma 2. *Suppose that the samples $x_1, \ldots, x_n \in \mathbb{R}^p$ are i.i.d. sub-Gaussian variables with bounded variance and let $n \geq C\delta^{-2} s \log K$, where C is a constant. Then, the subgroup isometry condition in* (15) *hold with probability at least $1 - C' e^{-C'' n}$, where C' and C'' are constants.*

Lemma 3. *[Pretraining approximates the average of individual parameters] Under the subgroup isometry condition* (15) *and the conditions of Lemma 1, the pretraining estimator satisfies*

$$\left\| \hat{\beta}_{\text{pre}} - \frac{1}{K} \sum_{k=1}^{K} \beta_k^* + \lambda (X_S^T X_S)^{-1} \mathbf{sign}(\beta_S^*) \right\|_2 \leq \delta \sum_{k=1}^{K} \|\beta_k^*\|_2 + \|X_S^\dagger n\|_2. \tag{16}$$

Remark 7. The above result shows that the pretraining estimator approximates the average of the individual models $\frac{1}{K}\sum_{k=1}^{K} \beta_k^*$, in addition to a shrinkage term proportional to λ.

The above result is derived from the optimality conditions for the lasso model (see Appendix).

5.3.3 Recovery under random design

Next, we prove that the Pretraining Irrepresentability condition holds with high probability when the features are generated from a random ensemble.

Theorem 1. *Suppose that the samples $x_1, \ldots, x_n \in \mathbb{R}^p$ are i.i.d. sub-Gaussian variables with bounded variance and the noise vectors $\varepsilon_1, \ldots, \varepsilon_K$ are sub-Gaussian with variance proxy σ^2. In addition, assume that $|\beta_k^*| \leq \gamma \, \forall k \in [K]$ for some $\gamma > 0$, and the number of samples satisfy*

$$n \geq C_1 \max(1, \gamma^2)\, s \log(p - s), \tag{17}$$

for some constant $C_1 > 0$. Then, the conditions (12) *and* (13) *when $\lambda = C_\lambda \sigma \sqrt{\frac{\log(p-s)}{n}}$ hold with probability at least $1 - C_3 e^{-C_4 n/(s\gamma^2)}$ where C_2, C_3, C_4 are constants. Therefore, $\hat{\beta}_{\text{pre}}$ discards the complement of the true support S with the same probability.*

Remark 8. It is instructive to compare the condition (17) with the known results on recovery with lasso under the classical linear observation setting [Wainwright, 2009] that require

$\mathcal{O}(s\log(p-s))$ observations. Therefore, using only the individual observations without pretraining via the ordinary lasso, we need $n/K > s\log(p-s)$ to achieve the same support recovery. Comparing this with Theorem 1, we observe that the pretraining procedure gives a factor K improvement in the required sample size.

Remark 9. We note that the factor $\max(1,\gamma)$ is the extra cost on the number of samples induced by the mixture observation model, which is due to the second condition (13). In order the pretraining estimator to discard irrelevant variables, the magnitude of each linear model weight β_k^* is required to be small. Furthermore, the pretraining estimator can discard variables from the true support S.

Theorem 2. *Suppose that the samples $x_1,\dots,x_n \in \mathbb{R}^p$ are i.i.d. sub-Gaussian variables with bounded variance and the noise vectors $\varepsilon_1,\dots,\varepsilon_K$ are sub-Gaussian with variance proxy σ^2. Set $\lambda = C_\lambda\sigma\sqrt{\frac{\log(p-s)}{n}}$. In addition, assume that $|(\beta_k^*)_j| \le \gamma\,\forall k \in [K]\,\forall j \in [p]$ for some $\gamma > 0$, and the number of samples satisfy*

$$n \ge C_1 \max(1,\gamma^2)\, s\log\big(\max(p-s,K)\big), \tag{18}$$

and

$$\min_j |\big(\frac{1}{K}\sum_{k=1}^{K}\beta_k^*\big)_j| \ge C_1'\sigma\sqrt{\frac{\log(p-s)}{n}}, \tag{19}$$

for some constants C_1, C_1'. Then, the pretraining estimator $\hat{\beta}_{\text{pre}}$ exactly recovers the ground truth support with probability at least $1 - C_3e^{-C_4 n}$ where C_λ, C_2, C_3, C_4 are constants.

Remark 10. The condition (19) is similar to the $\beta_{\min}$ conditions used in the classical analysis of lasso under the linear setting [Wainwright, 2009]. This condition is unavoidable to make sure the pretraining estimator does not discard variables in the ground truth support.

Remark 11. Note that there are pathological cases for which the pretraining estimator fails to recover the support. A simple example is where $\beta_2^* = -\beta_1^*$ and $K = 2$. In this case, the condition (19) can not hold since $\frac{1}{2}(\beta_1^* + \beta_2^*) = 0$. However, we are guaranteed that the pretraining estimator discards all the variables not in the ground truth support under the remaining assumptions.

5.4 Common and individual support model

We now consider K sets of observations

$$y_k = X_k\beta_0^* + X_k\beta_k^* + \varepsilon_k \in \mathbb{R}^{n/K}, \tag{20}$$

where β_0^* is an s sparse vector to account for shared features and β_k^* are s sparse unknown vectors modeling individual features. We assume that the support of β_0^* and β_k^* are not-overlapping.

5.4.1 Recovery under random design

Next, we prove that the pretraining estimator discards the complement of the true support with high probability when the features are generated from a random ensemble and the magnitude of the individual coefficients are sufficiently small.

Theorem 3. *Suppose that the samples $x_1, \ldots, x_n \in \mathbb{R}^p$ are i.i.d. sub-Gaussian variables with bounded variance and the noise vectors $\varepsilon_1, \ldots, \varepsilon_K$ are sub-Gaussian with variance proxy σ^2. Set $\lambda = C_\lambda \sigma \sqrt{\frac{\log(p-s)}{n}}$. In addition, assume that $|(\beta_0^*)_j| \leq \gamma_1 \, \forall j \in [p]$ and $|(\beta_k^*)_j| \leq \gamma_2 \, \forall k \in [K] \, \forall j \in [p]$ for some $\gamma_1 \in (0, \infty)$ and $\gamma_2 \in (0, \frac{\lambda}{4})$, and the number of samples satisfy*

$$n \geq C_5 \max(1, \gamma_1^2) \, s \log\big(\max(p-s, K)\big), \tag{21}$$

and

$$\min_j |\big(\beta_0^*\big)_j| \geq C_5' \sigma \sqrt{\frac{\log(p-s)}{n}}. \tag{22}$$

Then, the pretraining estimator $\hat{\beta}_{\text{pre}}$ exactly recovers the support of the common parameter β_0^* with probability at least $1 - C_6 e^{-C_7 n}$. Here, $C_\lambda, C_5, C_5', C_6$ are constants.

Remark 12. We note that the above theorem imposes the condition $|\beta_k^*| \leq \gamma_2 \, \forall k \in [K]$ for some $\gamma_2 \leq C_\lambda \sigma \sqrt{\frac{\log(p-s)}{n}}$ on the individual coefficients. This is a more stringent requirement compared to Theorem 1 where γ is unrestricted.

5.4.2 Error bounds for the two-stage procedure

We now present an analysis of a simplified form of our pretraining strategy followed by the fitting of individual models. Consider the common and individual support model

$$y_k = X_k\beta_0^* + X_k\beta_k^* \in \mathbb{R}^{n/K} + \varepsilon_k \quad \text{for } k \in [K]. \tag{23}$$

Let us denote the full feature matrix $X = [X_1^T, ..., X_k^T]^T \in \mathbb{R}^{n\times p}$. First, we fit our pretraining estimator

$$\hat{\beta}_0 \in \arg\min_{\beta_0\in\mathbb{R}^p} \sum_{k=1}^{K} \|X_k\beta_0 - y_k\|_2^2 + \lambda\|\beta_0\|_1 \,.$$

Consequently, we fit the individual models using $\hat{\beta}_0$ as an offset term

$$\hat{\beta}_k \in \arg\min_{\beta_k\in\mathbb{R}^p} \|X_k\hat{\beta}_0 + X_k\beta_k - y_k\|_2^2 + \lambda'\|\beta_k\|_1 \,,$$

for $k = 1, ..., K$. Note that we omit the penalty factors and apply ℓ_1 regularization without weighting. In addition, we assume that the tuning parameter α is fixed and known, and ignore it since it can be absorbed into the parameters. The following result presents the prediction error for this two-step procedure.

Theorem 4. *Suppose that $X_k \in \mathbb{R}^{n/K\times p}$ for $k \in [K]$ are fixed matrices and the noise vectors $\varepsilon_1, \ldots, \varepsilon_K$ are sub-Gaussian with variance proxy σ^2. Suppose that there exists constants C, C' such that the columns obey the average magnitude constraint $\frac{1}{n}\sum_{i=1}^{n} X_{ij}^2 \leq C$ for all $j \in [p]$, the average correlation constraint $\max_{j,j'\in[p]} |\frac{1}{n/K}\sum_{i=1}^{n/K}(X_k)_{ij}(X_k)_{ij'}| \leq C'$ for all $k \in [K]$. Let the regularization parameters satisfy $\lambda \geq \sqrt{n}2R(2C\sigma\sqrt{\log(p)} + C'\frac{n}{K}R')$ and $\lambda' \geq 4\sigma\sqrt{n/K}\sqrt{\log(pK)}$ where $R := \|\beta_0^*\|_1$ and $R' := \sum_{k=1}^{K}\|\beta_k^*\|_1$. Then, we have the following in-sample prediction error bound*

$$\frac{1}{n}\sum_{k=1}^{K}\|X_k(\hat{\beta}_0 - \beta_0^* + \hat{\beta}_k - \beta_k^*)\|_2^2 \leq \frac{\lambda R + \lambda' R'}{n} + \frac{\sigma(CR\sqrt{\log(p)} + R'\sqrt{\log(pK)/K})}{\sqrt{n}} + \frac{C'R'}{K},$$

with probability at least $1 - C_3/n$ for a certain constant C_3.

Remark 13. *The prediction error is composed of three parts: the first part is proportional to the regularization parameters λ and λ', which can be minimized by selecting the smallest permissible*

values, $\lambda = \sqrt{n}2R(2C\sigma\sqrt{\log(p)} + C'\frac{n}{K}R')$ *and* $\lambda' = 4\sigma\sqrt{n/K}\sqrt{\log(pK)}$*. The second part decreases to zero as the total number of samples* n *grows to infinity, and the third part vanishes as the number of groups* K *increases. It is important to note that* $R' = \sum_{k=1}^{K} \|\beta_k^*\|_1$*, the sum of the* ℓ_1 *norms of the individual parameters* β_k^**, should grow more slowly than* K*. For example, a scaling of* $\|\beta_k^*\|_1 = \mathcal{O}\left(\frac{1}{K}\right)$ *for all* $k \in [K]$ *satisfies this condition. Such a scaling assumption is unavoidable since we need to ensure that the pretraining stage estimates the common parameter* β_0 *in the presence of individual parameters which effectively act as a disturbance term.*

Remark 14. *The above prediction error can be compared with the individual ordinary lasso estimators* $\tilde{\beta}_k$ *fitted to the data* X_k, y_k *for each* $k \in [K]$*. A standard upper-bound for the average in-sample prediction error for this scheme under the same assumptions as in Theorem 4 is (e.g., see Theorem 11.2. in Hastie et al. [2015])*

$$\frac{1}{n}\sum_{k=1}^{K} \|X_k(\tilde{\beta}_k - \beta_0^* - \beta_k^*)\|_2^2 \lesssim \frac{(\sqrt{K}R + C''/\sqrt{K})\sqrt{\log p}}{\sqrt{n}}, \tag{24}$$

which holds with the probability at least $1 - C_4/n$ *for some constant* C_4*. We emphasize that the term* $\sqrt{K}R$ *is due to ignoring the common component* β_0 *across all groups, and leads to a factor of* $\sqrt{K}$ *larger prediction error compared to our bound provided in Theorem 4.*

The proof of Theorem 4 is provided in the Appendix.

6 Discussion

In this paper we have developed a framework that enables the power of ML pretraining — designed for neural nets — to be applied in a simpler statistical setting (the lasso). We discuss many diverse applications of this paradigm, including stratified models, multinomial responses, multi-response models, conditional average treatment estimation and gradient boosting. Pretraining can be extended to incorporate unlabeled data alongside labeled data using sparse principal component analysis for feature selection. There are likely to be other interesting applications of these ideas. An open source R language package implementing this work is available at github.com/erincr/ptLasso and will be on CRAN in the near future.

Acknowledgments. The authors would like to thank Emmanuel Candes, Daisy Ding, Trevor Hastie, Sarah McGough, Vishnu Shankar, Lu Tian, Ryan Tibshirani, and

Stefan Wager for helpful discussions. We thank Yu Liu and Yingcun Xia for implementing changes to their R package ODRF. B.N. was supported by National Center For Advancing Translational Sciences of the National Institutes of Health under Award Number UL1TR003142. M.A.R. is in part supported by NHGRI under award R01HG010140, and by NIMH under award R01MH124244. R.T. was supported by the NIH (5R01EB001988-16) and the NSF (19DMS1208164). M.P. was supported in part by NSF under Grant DMS-2134248, in part by the NSF CAREER Award under Grant CCF-2236829, in part by the U.S. Army Research Office Early Career Award under Grant W911NF-21-1-0242, and in part by the Stanford Precourt Institute. E.C. was supported by the Stanford Data Science Scholars Program and the Stanford Graduate Fellowship. J.S. was supported by the NIGMS, Stanford University Discovery Innovation Award, and the Chan Zuckerberg Data Insights. S.G. was supported by the Postdoctoral fellowship granted by the Western Norway Regional Health Authority.

References

Kaitlin Chaung, Tavor Z. Baharav, George Henderson, Ivan N. Zheludev, Peter L. Wang, and Julia Salzman. Splash: A statistical, reference-free genomic algorithm unifies biological discovery. *Cell*, 186(25):5440–5456.e26, 2023. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2023.10.028. URL `https://www.sciencedirect.com/science/article/pii/S0092867423011790`.

Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li, and Jiaming Yuan. *xgboost: Extreme Gradient Boosting*, 2023. URL `https://CRAN.R-project.org/package=xgboost`. R package version 1.7.6.1.

The Tabula Sapiens Consortium, Robert C Jones, Jim Karkanias, Mark A Krasnow, Angela Oliveira Pisco, Stephen R Quake, Julia Salzman, Nir Yosef, Bryan Bulthaup, Phillip Brown, et al. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376(6594):eabl4896, 2022.

Jerome Friedman, Robert Tibshirani, and Trevor Hastie. Regularization paths for gen-

eralized linear models via coordinate descent. *Journal of Statistical Software*, 33(1): 1–22, 2010. doi: 10.18637/jss.v033.i01.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. 2008.

Mary J Goldman, Brian Craft, Mim Hastie, Kristupas Repečka, Fran McDade, Akhil Kamath, Ayan Banerjee, Yunhai Luo, Dave Rogers, Angela N Brooks, et al. Visualizing and interpreting cancer genomics data via the xena platform. *Nature biotechnology*, 38 (6):675–678, 2020.

Samuel M Gross and Robert Tibshirani. Data shared lasso: A novel tool to discover uplift. *Computational statistics & data analysis*, 101:226–235, 2016.

Stein-Erik Gullaksen, Jørn Skavland, Sonia Gavasso, Vinko Tosevski, Krzysztof Warzocha, Claudia Dumrese, Augustin Ferrant, Tobias Gedde-Dahl, Andrzej Hellmann, Jeroen Janssen, et al. Single cell immune profiling by mass cytometry of newly diagnosed chronic phase chronic myeloid leukemia treated with nilotinib. *Haematologica*, 102(8):1361, 2017.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015. ISBN 1498712169.

Marek Kokot, Roozbeh Dehghannasiri, Tavor Baharav, Julia Salzman, and Sebastian Deorowicz. Splash2 provides ultra-efficient, scalable, and unsupervised discovery on raw sequencing reads. *BioRxiv*, 2023.

Katherine Margulis, Albert S Chiou, Sumaira Z Aasi, Robert J Tibshirani, Jean Y Tang, and Richard N Zare. Distinguishing malignant from benign microscopic skin lesions using desorption electrospray ionization mass spectrometry imaging. *Proceedings of the National Academy of Sciences*, 115(25):6347–6352, 2018.

Sarah F. McGough, Svetlana Lyalina, Devin Incerti, Yunru Huang, Stefka Tyanova, Kieran Mace, Chris Harbron, Ryan Copping, Balasubramanian Narasimhan, and Robert Tibshirani. Prognostic pan-cancer and single-cancer models: A large-scale analysis using a real-world clinico-genomic database. *medRxiv*, 2023. doi: 10.1101/2023.12.18.23300166. URL `https://www.medrxiv.org/content/early/2023/12/19/2023.12.18.23300166`.

Xinkun Nie and Stefan Wager. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 108(2):299–319, 2021.

Julia Eve Olivieri, Roozbeh Dehghannasiri, Peter L Wang, SoRi Jang, Antoine De Morree, Serena Y Tan, Jingsi Ming, Angela Ruohao Wu, Stephen R Quake, Mark A Krasnow, et al. Rna splicing programs define tissue compartments and cell types at single-cell resolution. *Elife*, 10:e70692, 2021.

Rudolf J Schilder, Scot R Kimball, and Leonard S Jefferson. Cell-autonomous regulation of fast troponin t pre-mrna alternative splicing in response to mechanical stretch. *American Journal of Physiology-Cell Physiology*, 303(3):C298–C307, 2012.

Bert Skagerberg, John F MacGregor, and Costas Kiparissides. Multivariate data analysis applied to low-density polyethylene reactors. *Chemometrics and intelligent laboratory systems*, 14(1-3):341–356, 1992.

Jonathan Tuck and Stephen Boyd. Fitting laplacian regularized stratified gaussian models. *Optimization and Engineering*, pages 1–21, 2021.

Sara A Van De Geer and Peter Bühlmann. On the conditions used to prove oracle results for the lasso. 2009.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Martin J Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using l1-constrained quadratic programming (lasso). *IEEE transactions on information theory*, 55(5):2183–2202, 2009.

Guo Yu, Jacob Bien, and Ryan Tibshirani. Reluctant interaction modeling. *arXiv preprint arXiv:1907.08414*, 2019.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

A Simulation study results

A.1 Input grouped data

Here, we consider the input grouped case, and we compare pretraining to the overall model and individual models in terms of (1) predictive performance on test data, (2) their F1 scores for feature selection and (3) F1 scores for feature selection among the common features only. Our simulations cover grouped data with a continuous response (Table 2), grouped data with a binomial response (Table 3), and data with a multinomial response (Table 4).

SNR	PSE relative to Bayes error			Feature F1			Common feature F1	
	Overall	Pretrain	Indiv.	Overall	Pretrain	Indiv.	Pretrain	Indiv.
Common support with same magnitude, individual features Features: 10 common (coefficient values: 5), 10 per group (coefficient values: 3), 120 total								
13.6	4.3 ± 0.3	$\mathbf{1.4 \pm 0.1}$	1.4 ± 0.1	$\mathbf{75 \pm 3}$	72 ± 2	70 ± 1	31 ± 6	$\mathbf{39 \pm 5}$
3.4	1.9 ± 0.1	$\mathbf{1.4 \pm 0.1}$	$\mathbf{1.4 \pm 0.1}$	70 ± 5	$\mathbf{73 \pm 3}$	70 ± 1	$\mathbf{40 \pm 8}$	39 ± 5
0.4	$\mathbf{1.1 \pm 0.0}$	1.2 ± 0.0	1.3 ± 0.1	23 ± 4	50 ± 6	$\mathbf{64 \pm 5}$	$\mathbf{86 \pm 10}$	73 ± 14
As above, but now $p > n$ Features: 10 common (coefficient values: 5), 10 per group (coefficient values: 2), 2040 total								
11.6	2.8 ± 0.1	$\mathbf{1.9 \pm 0.2}$	2.7 ± 0.2	$\mathbf{37 \pm 5}$	33 ± 5	21 ± 2	72 ± 17	$\mathbf{91 \pm 6}$
2.9	$\mathbf{1.5 \pm 0.1}$	1.5 ± 0.1	2.1 ± 0.1	$\mathbf{33 \pm 4}$	31 ± 5	18 ± 2	81 ± 15	$\mathbf{97 \pm 4}$
0.3	$\mathbf{1.1 \pm 0.0}$	1.2 ± 0.0	1.3 ± 0.0	$\mathbf{29 \pm 4}$	23 ± 5	13 ± 4	$\mathbf{84 \pm 12}$	5 ± 9
Common support with same magnitude, no individual features Features: 10 common (coefficient values: 5), 0 per group, 120 total								
10.0	$\mathbf{1.0 \pm 0.0}$	1.1 ± 0.0	1.2 ± 0.1	50 ± 11	$\mathbf{88 \pm 14}$	21 ± 2	$\mathbf{92 \pm 7}$	76 ± 9
2.5	$\mathbf{1.1 \pm 0.0}$	$\mathbf{1.1 \pm 0.0}$	1.2 ± 0.0	52 ± 11	$\mathbf{85 \pm 16}$	21 ± 2	$\mathbf{91 \pm 9}$	74 ± 9
0.3	$\mathbf{1.1 \pm 0.0}$	$\mathbf{1.1 \pm 0.0}$	1.2 ± 0.0	51 ± 11	$\mathbf{74 \pm 19}$	30 ± 7	$\mathbf{92 \pm 8}$	76 ± 15
Common support with different magnitudes, no individual features Features: 10 common (coefficient values: 1 in group 1, 2 in group 2, etc.), 0 per group, 120 total								
4.4	1.9 ± 0.1	$\mathbf{1.1 \pm 0.0}$	1.2 ± 0.1	51 ± 10	$\mathbf{62 \pm 20}$	22 ± 2	$\mathbf{93 \pm 8}$	79 ± 9
1.1	1.3 ± 0.0	$\mathbf{1.1 \pm 0.0}$	1.2 ± 0.0	52 ± 10	$\mathbf{73 \pm 22}$	23 ± 3	$\mathbf{93 \pm 9}$	86 ± 10
0.1	$\mathbf{1.1 \pm 0.0}$	$\mathbf{1.1 \pm 0.0}$	$\mathbf{1.1 \pm 0.0}$	$\mathbf{53 \pm 12}$	52 ± 18	38 ± 11	$\mathbf{64 \pm 21}$	16 ± 19
As above, but now with individual features Features: 10 common (as above), 10 per group (coefficient values: 3), 120 total								
10.8	3.8 ± 0.2	$\mathbf{1.4 \pm 0.1}$	$\mathbf{1.4 \pm 0.1}$	64 ± 5	$\mathbf{73 \pm 3}$	70 ± 1	$\mathbf{49 \pm 9}$	40 ± 5
4.8	2.3 ± 0.1	$\mathbf{1.3 \pm 0.1}$	1.4 ± 0.1	61 ± 5	$\mathbf{73 \pm 3}$	70 ± 2	$\mathbf{64 \pm 13}$	47 ± 7
1.2	1.4 ± 0.1	$\mathbf{1.2 \pm 0.1}$	1.3 ± 0.1	55 ± 5	51 ± 10	$\mathbf{65 \pm 3}$	$\mathbf{86 \pm 11}$	76 ± 12
Individual features only Features: 0 common, 10 per group (coefficient values: 5), 120 total								
10.1	9.8 ± 0.6	$\mathbf{1.2 \pm 0.0}$	$\mathbf{1.2 \pm 0.0}$	65 ± 3	$\mathbf{67 \pm 3}$	$\mathbf{67 \pm 4}$	—	—
2.5	3.3 ± 0.2	$\mathbf{1.2 \pm 0.1}$	$\mathbf{1.2 \pm 0.1}$	63 ± 6	$\mathbf{68 \pm 3}$	$\mathbf{68 \pm 2}$	—	—
0.6	1.6 ± 0.1	$\mathbf{1.3 \pm 0.1}$	$\mathbf{1.3 \pm 0.1}$	52 ± 5	$\mathbf{69 \pm 2}$	68 ± 3	—	—

Table 2: *Gaussian response, 5 input groups. Each input group has 100 observations; in total there are* $n = 500$ *observations. Each row represents* 100 *simulations, and each result shows the mean and one standard deviation.*

Test AUC (x 100)				Feature F1			Common feature F1	
Bayes	Overall	Pretrain	Indiv.	Overall	Pretrain	Indiv.	Pretrain	Indiv.
Common support with same magnitude, individual features Features: 5 common (-.5, .5, .3, -.9, .1), 5 per group (-.45, .45, .27, -.81, .09), 40 total								
82 ± 2	71 ± 3	$\mathbf{72 \pm 4}$	70 ± 4	62 ± 6	65 ± 8	$\mathbf{66 \pm 6}$	$\mathbf{44 \pm 15}$	39 ± 13
0.7 ± 3	$\mathbf{60 \pm 4}$	59 ± 4	58 ± 3	57 ± 8	60 ± 11	$\mathbf{63 \pm 8}$	$\mathbf{35 \pm 12}$	28 ± 12
61 ± 3	$\mathbf{54 \pm 3}$	53 ± 3	53 ± 3	57 ± 9	$\mathbf{60 \pm 13}$	$\mathbf{60 \pm 13}$	$\mathbf{24 \pm 13}$	18 ± 11
As above, but now $p > n$ Features: 5 common (-.5, .5, .3, -.9, .1), 5 per group (-.45, .45, .27, -.81, .09), 320 total								
82 ± 2	$\mathbf{68 \pm 4}$	67 ± 4	63 ± 4	$\mathbf{42 \pm 9}$	24 ± 7	23 ± 6	$\mathbf{40 \pm 16}$	32 ± 13
70 ± 3	$\mathbf{56 \pm 4}$	55 ± 4	53 ± 4	$\mathbf{29 \pm 11}$	16 ± 6	15 ± 5	$\mathbf{24 \pm 14}$	14 ± 13
61 ± 3	$\mathbf{51 \pm 3}$	$\mathbf{51 \pm 3}$	$\mathbf{51 \pm 4}$	$\mathbf{21 \pm 9}$	10 ± 6	12 ± 4	$\mathbf{8 \pm 8}$	3 ± 6
Common support with same magnitude, no individual features Features: 5 common (-.5, .5, .3, -.9, .1), 0 per group, 40 total								
77 ± 2	$\mathbf{73 \pm 3}$	70 ± 4	66 ± 4	$\mathbf{56 \pm 7}$	47 ± 16	$\mathbf{56 \pm 14}$	$\mathbf{63 \pm 16}$	50 ± 17
65 ± 3	$\mathbf{60 \pm 4}$	57 ± 4	55 ± 4	54 ± 9	54 ± 16	$\mathbf{62 \pm 11}$	$\mathbf{39 \pm 15}$	30 ± 13
58 ± 4	$\mathbf{53 \pm 4}$	51 ± 4	51 ± 3	55 ± 9	59 ± 14	$\mathbf{63 \pm 9}$	$\mathbf{27 \pm 11}$	23 ± 10
Common support with different magnitudes, no individual features Features: 5 common, 0 per group, 40 total Group 1 coefficient values: (-.5, .5, .3, -.9, .1) Group 2 coefficient values: (-.3, .9, .1, -.1, .2) Group 3 coefficient values: (.1, .2, -.1, .2, .3)								
71 ± 3	61 ± 4	$\mathbf{62 \pm 4}$	61 ± 4	53 ± 9	$\mathbf{56 \pm 14}$	$\mathbf{56 \pm 15}$	$\mathbf{44 \pm 16}$	36 ± 17
62 ± 3	$\mathbf{54 \pm 4}$	$\mathbf{54 \pm 4}$	53 ± 4	56 ± 9	60 ± 15	$\mathbf{62 \pm 11}$	$\mathbf{29 \pm 11}$	27 ± 10
56 ± 4	51 ± 3	$\mathbf{52 \pm 3}$	51 ± 3	56 ± 10	61 ± 13	$\mathbf{64 \pm 5}$	$\mathbf{22 \pm 11}$	$\mathbf{22 \pm 10}$
As above, but now with individual features Features: 5 common, 5 per group, 40 total Common support coefficients are as above. Individual support coefficient values are 0.9 × common support coefficient values.								
76 ± 3	61 ± 4	$\mathbf{65 \pm 4}$	64 ± 4	58 ± 9	63 ± 9	$\mathbf{64 \pm 8}$	$\mathbf{37 \pm 14}$	30 ± 15
70 ± 3	56 ± 4	$\mathbf{58 \pm 4}$	$\mathbf{58 \pm 4}$	56 ± 10	61 ± 8	$\mathbf{63 \pm 7}$	$\mathbf{28 \pm 13}$	26 ± 13
64 ± 3	$\mathbf{54 \pm 3}$	$\mathbf{54 \pm 4}$	$\mathbf{54 \pm 3}$	55 ± 9	58 ± 14	$\mathbf{61 \pm 12}$	$\mathbf{24 \pm 13}$	21 ± 12
Individual features only Group 1 coefficient values: (-.5, .5, .3, -.9, .1) Group 2 coefficient values: (-.3, .9, .1, -.1, .2) Group 3 coefficient values: (.1, .2, -.1, .2, .3)								
72 ± 3	56 ± 3	$\mathbf{62 \pm 4}$	$\mathbf{62 \pm 4}$	$\mathbf{57 \pm 9}$	59 ± 10	55 ± 14	—	—
66 ± 3	53 ± 4	$\mathbf{56 \pm 3}$	$\mathbf{56 \pm 3}$	58 ± 9	$\mathbf{60 \pm 12}$	59 ± 13	—	—
60 ± 3	52 ± 3	$\mathbf{53 \pm 3}$	$\mathbf{53 \pm 3}$	57 ± 8	$\mathbf{63 \pm 10}$	59 ± 14	—	—

Table 3: *Binomial response, 3 input groups. Each input group has 100 observations; in total there are* $n = 300$ *training observations. Each row represents* 100 *simulations, and each result shows the mean and one standard deviation.*

Test misclassification rate (x 100)				Feature F1			Common feature F1	
Bayes rule	Overall	Pretrain	Indiv.	Overall	Pretrain	Indiv.	Pretrain	Indiv.
Common support, individual features Features: 3 common (simulation 1: group 1 - group 5 values: -.5, -0.25, 0, 0.25, 0.5) (simulation 2: group 1 - group 5 values: -1, -.5, 0, 0.5, 1) 10 per group (same values as above) 159 total								
49 ± 1	59 ± 2	$\mathbf{58 \pm 2}$	60 ± 2	55 ± 8	$\mathbf{57 \pm 5}$	55 ± 5	26 ± 19	$\mathbf{28 \pm 31}$
22 ± 1	33 ± 2	$\mathbf{32 \pm 2}$	$\mathbf{32 \pm 3}$	$\mathbf{71 \pm 6}$	65 ± 6	65 ± 6	20 ± 15	$\mathbf{55 \pm 20}$
As above, but now $p > n$ Features: 3 common, 10 per group, 640 total Values as above, with extra noise features.								
56 ± 1	62 ± 2	$\mathbf{61 \pm 2}$	62 ± 3	34 ± 7	$\mathbf{36 \pm 6}$	$\mathbf{36 \pm 5}$	$\mathbf{26 \pm 20}$	21 ± 30
28 ± 1	35 ± 3	$\mathbf{34 \pm 2}$	36 ± 4	$\mathbf{56 \pm 11}$	48 ± 8	49 ± 7	24 ± 22	$\mathbf{73 \pm 21}$
Common support, no individual features Features: 3 common, 0 per group, 159 total Common feature values as above.								
70 ± 1	71 ± 3	$\mathbf{70 \pm 2}$	75 ± 4	$\mathbf{32 \pm 13}$	28 ± 10	21 ± 11	$\mathbf{36 \pm 22}$	7 ± 21
58 ± 1	$\mathbf{57 \pm 2}$	58 ± 1	60 ± 2	24 ± 10	29 ± 10	$\mathbf{31 \pm 10}$	$\mathbf{49 \pm 24}$	36 ± 45
Individual features only Features: 0 common, 10 per group, 159 total Individual feature values as above.								
54 ± 1	64 ± 2	$\mathbf{63 \pm 2}$	64 ± 3	$\mathbf{54 \pm 7}$	$\mathbf{54 \pm 4}$	50 ± 7	—	—
27 ± 1	38 ± 2	$\mathbf{36 \pm 2}$	37 ± 3	$\mathbf{69 \pm 6}$	63 ± 6	64 ± 6	—	—

Table 4: *Multinomial response, 5 input groups. Each input group has 50 observations; in total there are* $n = 250$ *observations. Each row represents* 100 *simulations, and each result shows the mean and one standard deviation.*

A.2 External dataset

We repeat the experiment in Section 2.4, now studying the setting where the 10 groups in D_1 share support, but the coefficients on that support have different values (same sign). As before, the group specific coefficients β_g were a mixture of shared and group-specific coefficients, controlled by a parameter ρ:

$$\beta_g = \rho \times \beta_{g,\text{ shared}} + (1 - \rho) \times \beta_{g,\text{ indiv}},$$

where $\beta_{g,\text{ shared}}$ is now a vector with the same support (non-zero entries), but different values for each group, and $\beta_{g,\text{ indiv}}$ is group specific. Also as before, the supports for $\beta_{g,\text{ indiv}}$ are non-overlapping across groups. Performance for $\rho = 0, 0.5, 1$ is shown in Figure 13. Using all of D_1 hurts performance enough that the plot becomes difficult to read; we show the plot again with this case omitted. Pretraining always matched or

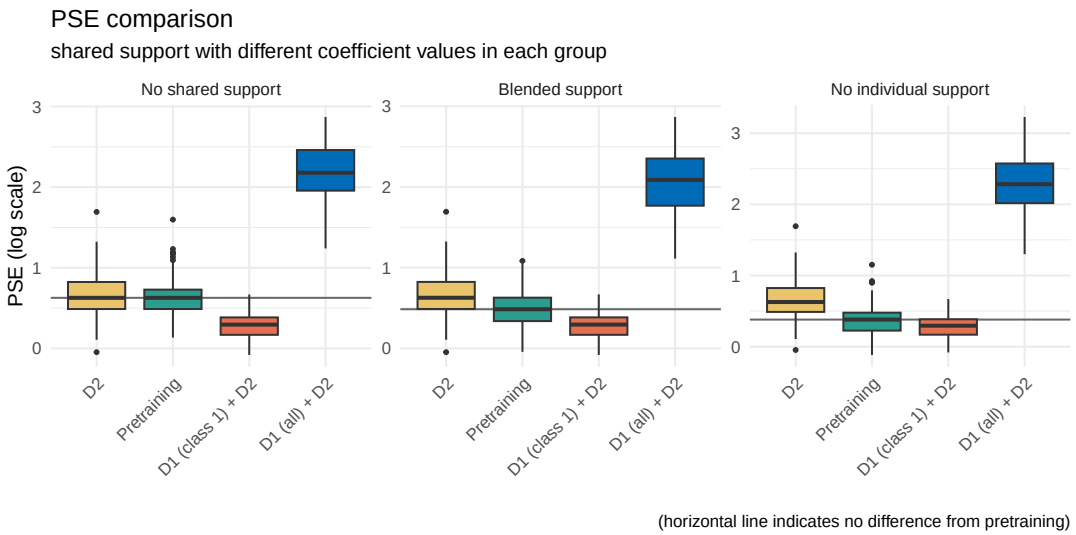

Figure 13: *Comparison of approaches for modeling with a large (usually inaccessible) dataset D_1 and a smaller dataset D_2. When D_1 is informative, pretraining using only D_2 and the coefficients from a model trained with D_1 outperforms using D_2 alone. When D_1 is not informative, pretraining with D_1 does not hurt.*

outperformed using D_2 alone; access to class 1 from D_1 performed better than or roughly the same as pretraining.

B Mathematical Proofs

B.1 Proof of Lemma 1

We analyze the conditions for optimality of the pretraining estimator given in (11). We apply the scaling $X \leftarrow \frac{1}{\sqrt{n}}X$ and $y \leftarrow \frac{1}{\sqrt{n}}y$ to absorb the $\frac{1}{n}$ factor and simplify our notation. Suppose that the support of the optimal solution β is S, which is assumed to contain the support of β_k^* $\forall k$. The optimality conditions that ensure β is the unique solution with support S are as follows

$$X_S^T(X_S\beta_S - y) + \lambda \mathbf{sign}(\beta_S) = 0 \tag{25}$$

$$\|X_{S^c}^T(X_S\beta_S - y)\|_\infty < \lambda, \tag{26}$$

where $\beta = \hat{\beta}_{\text{pre}}$ is the optimal solution. When the matrix $X_S \in \mathbb{R}^{n\times s}$ is full column-rank, the matrix $X_S^T X_S$ is invertible and we can solve for β_S as follows

$$\beta_S = (X_S^T X_S)^{-1}(X_S^T y - \lambda \mathbf{sign}(\beta_S)). \tag{27}$$

Plugging in the observation model $y = \sum_{k=1}^{K} D_k X w_k^* + \varepsilon$, we obtain

$$\beta_S = (X_S^T X_S)^{-1}(X_S^T \sum_{k=1}^{K} D_k X w_k^*) + X_S^\dagger \varepsilon - \lambda (X_S^T X_S)^{-1} \mathbf{sign}(\beta_S). \tag{28}$$

Plugging in the above expression into the condition (26), and dividing both sides by λ, we obtain

$$\|X_{S^c}^T(\lambda^{-1} P_S^\perp \sum_{k=1}^{K} D_k X w_k^* + \lambda^{-1} P_S^\perp \varepsilon + X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty < 1. \tag{29}$$

Using triangle inequality, we upper-bound the left-hand-side to arrive the sufficient condition

$$\lambda^{-1}\|X_{S^c}^T P_S^\perp \sum_{k=1}^{K} D_k X w_k^*\|_\infty + \lambda^{-1}\|X_{S^c}^T P_S^\perp \varepsilon\|_\infty + \|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty < 1. \tag{30}$$

Therefore by imposing the conditions

$$\|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty < \frac{1}{2} \tag{31}$$

$$\|X_{S^c}^T P_S^\perp \sum_{k=1}^{K} D_k X w_k^*\|_\infty \leq \frac{\lambda}{4} \tag{32}$$

$$\|X_{S^c}^T P_S^\perp \varepsilon\|_\infty \leq \frac{\lambda}{4}, \tag{33}$$

we observe that the optimality conditions for β with the support S are satisfied.

B.2 Proof of Theorem 1

First condition

We consider the first condition of pretraining irrepresentability given by

$$\|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty = \max_{j \in S^c} |x_j^T X_S^\dagger \mathbf{sign}(\beta_S)|. \tag{34}$$

Note that x_j^T and $X_S^\dagger \mathbf{sign}(\beta_S)$ are independent for $j \in S^c$. Therefore, $X_S^\dagger \mathbf{sign}(\beta_S)$ is sub-Gaussian with variance proportional to $\frac{1}{n}\|X_S^\dagger \mathbf{sign}(\beta_S)\|_2^2$.

When $n \geq Cs$ for some constant C, the matrix $X_S^T X_S$ is a near-isometry in spectral norm, i.e.,

$$\|X_S^T X_S - I\|_2 \leq \delta, \tag{35}$$

with probability at least $1 - C_1 e^{-C_2 n}$ where C_1, C_2 are constants. Therefore for $\delta < 1$, we have $\|X_S^\dagger\|_2 \leq (1-\delta)^{-1}$ and $\|X_S^\dagger \mathbf{sign}(\beta_S)\|_2^2 \lesssim \|\mathbf{sign}(\beta_S)\|_2^2 = s$.

Applying union bound, we obtain

$$\mathbb{P}\left[\max_{j \in S^c} |x_j^T X_S^\dagger \mathbf{sign}(\beta_S)| \leq \delta\right] \leq (p-s)\mathbb{P}\left[|x_1^T X_S^\dagger \mathbf{sign}(\beta_S)| \leq \delta\right] \tag{36}$$

$$\leq (p-s)e^{-\delta^2 C' n/s} \tag{37}$$

$$= e^{-C'\delta^2 n/s + \log(p-s)}, \tag{38}$$

for some constant C'.

Consequently, for $n \gtrsim \delta^{-2} s \log(p-s)$ we have $\max_{j \in S^c} |x_j^T X_S^\dagger \mathbf{sign}(\beta_S)| \leq \delta$ with probability at least $1 - C_3 e^{-C_4 \delta^2 n/s}$ where C_3, C_4 are constants.

Second condition

We proceed bounding the second irrepresentability condition involving the matrix $X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X w_k^*$ using the same strategy used above. Note the critical fact that the shared support model implies the matrices X_{S^c} and

$$P_S^\perp \sum_{k=1}^K D_k X w_k^* = P_S^\perp \sum_{k=1}^K D_k X_S (w_k^*)_S,$$

are independent since the latter matrix only depends on the features X_S.

Note that

$$\|P_S^{\perp}\sum_{k=1}^{K} D_k X_S(w_k^*)_S\|_2 \leq \|\sum_{k=1}^{K} D_k X_S(w_k^*)_S\|_2 \tag{39}$$

$$= \left(\sum_{k=1}^{K} \|D_k X_S(w_k^*)_S\|_2^2\right)^{1/2}. \tag{40}$$

Recalling the scaling of the X by $\frac{1}{n}$, we note that $D_k X_S$ is an $n \times s$ formed by the concatenation of an an $\frac{n}{K} \times s$ matrix of i.i.d. sub-Gaussian variables with variance $\mathcal{O}(\frac{1}{n})$ with an $(n-k) \times s$ matrix of zeros. From standard results on the singular values of sub-Gaussian matrices Vershynin [2018], we have $\|D_k X_S\|_2 \lesssim \frac{\sqrt{n/K}+\sqrt{s}}{\sqrt{n}} = \sqrt{\frac{1}{K}} + \sqrt{\frac{s}{n}}$ with probability at least $1 - C_5 e^{-C_6 n}$. Using the fact that w_k^* has entries bounded in $[-\gamma, +\gamma]$, we obtain the upper-bound

$$\|P_S^{\perp}\sum_{k=1}^{K} D_k X_S(w_k^*)_S\|_2 \leq \left(\sum_{k=1}^{K} \|D_k X_S(w_k^*)_S\|_2^2 \|(w_k^*)_S\|_2^2\right)^{1/2} \tag{41}$$

$$\lesssim \left(\sum_{k=1}^{K} (\frac{1}{K} + \frac{s}{n}) s\gamma^2\right)^{1/2} \tag{42}$$

$$= \left((1 + \frac{Ks}{n}) s\gamma^2\right)^{1/2} \tag{43}$$

$$\leq \sqrt{s}\gamma + \frac{\sqrt{K}}{\sqrt{n}} s\gamma \tag{44}$$

$$\leq 2\sqrt{s}\gamma, \tag{45}$$

where we used the fact that $n/K \geq s$, i.e., each subgroup has at least s samples, in the final inequality. Repeating the same argument involving sub-Gaussian variables and the union bound used for the first condition above, we obtain that for $n \gtrsim \delta^{-2}\gamma^2 s \log(p-s)$ we have $X_{S^c}^T P_S^{\perp} \sum_{k=1}^{K} D_k X w^* \leq \delta$ with probability at least $1 - C_7 e^{-C_8 n\delta^2/(s\gamma^2)}$ where C_7, C_8 are constants.

Third condition

Using standard results on Gaussian vectors, and repeating the union bound argument used in analyzing the first condition, we obtain that $\|X_{S^c}^T P_S^{\perp} \varepsilon\|_{\infty} \leq \delta\lambda$ when we set

$\lambda = C_\lambda \sigma \sqrt{\frac{\log p - s}{n}}$ and $n \gtrsim \delta^{-2}\sigma^2 \log(p-s)$ with probability at least $1 - C_9 e^{-C_{10} n\delta^2/\sigma^2}$ where C_λ, C_9, C_{10} are constants.

Applying union bound to bound the probability that all of the three conditions hold simultaneously, we complete the proof of the theorem.

B.3 Proof of Lemma 2

We apply well-known concentration bounds for the extreme singular values of i.i.d. Gaussian matrices (see e.g. Vershynin [2018]). These bounds $\|X_S^T X_S - KI\| \leq c_1 \delta$ and $\|X_S^T D_k X_S - I\| \leq c_2 \delta$ for each fixed $k \in [K]$ with high probability when $n \succeq \delta^{-2} s$. Applying union bound over $k \in [K]$, we obtain the claimed result.

B.4 Proof of Lemma 3

We consider the expression for β_S given in (28) in the proof of Lemma 1. Applying triangle inequality to control the terms on the right-hand-side, we obtain the claimed result.

B.5 Proof of Theorem 2

Note that we only need to control the signs of β_S given in (27), in addition to the guarantees of Theorem 1. Our strategy is to bound the ℓ_∞ norm of $\beta_S - \frac{1}{K}\sum_{k=1}^K \beta_k^*$ via its ℓ_2 norm and establishing entrywise control on β_S by the assumption on the minimum value of the average $\frac{1}{K}\sum_{k=1}^K \beta_k^*$. We combine Lemma 2 and Lemma 3 with the expression (27) to obtain

$$\|\beta_S - \frac{1}{K}\sum_{k=1}^K \beta_k^*\|_\infty \leq \lambda \|(X_S^T X_S)^{-1} \mathbf{sign}(\beta_S^*)\|_2 + \delta \sum_{k=1}^K \|\beta_k^*\|_2 + \|X_S^\dagger \varepsilon\|, \tag{46}$$

with high probability. Noting that $\|(X_S^T X_S)^{-1}\|_2 \lesssim K(1+\delta)$ with high probability, and $\sum_{k=1}^K \|\beta_k^*\|_2 \leq K\gamma$ by our assumption on the magnitude of β_k^*, we obtain the claimed result.

B.6 Proof of Theorem 3

The main difference of this result compared to the proof of Theorem 1 is in the analysis of the quantity $\|X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X w_k^*\|_\infty$. Unfortunately, X_{S^c} and $P_S^\perp \sum_{k=1}^K D_k X w_k^*$ are no longer independent. We proceed as follows

$$\|X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X w_k^*\|_\infty = \max_{j \in S^c} |x_j^T \sum_{k=1}^K D_k X w_k^*| \tag{47}$$

$$= |x_j^T \sum_{r \neq j} \sum_{k=1}^K D_k X (w_k^*)_r + x_j^T D_k X (w_k^*)_j| \tag{48}$$

$$\leq |x_j^T \sum_{r \neq j} \sum_{k=1}^K D_k X (w_k^*)_r| + |x_j^T x_j (w_k^*)_j| \tag{49}$$

we bound the last term via $|x_j^T x_j (w_k^*)_j| \leq \|x_j\|_2^2 \gamma_2^2$ and impose $\gamma_2 \in (0, \frac{\lambda}{4})$. Note that $\|x_j\|_2^2 \lesssim 1$ with high probability due to the rescaling by $\frac{1}{n}$. The rest of the proof is identical to the proof of Theorem 1.

B.7 Proof of Theorem 4

We first derive an error bound for the pretraining stage using the basic inequality

$$\sum_{k=1}^K \|X_k \hat{\beta}_0 - y_k\|_2^2 + \lambda \|\hat{\beta}_0\|_1 \leq \sum_{k=1}^K \|X_k \beta_0^* - y_k\|_2^2 + \lambda \|\beta_0^*\|_1, \tag{50}$$

which follows from the optimality of $\hat{\beta}_0$ in the pretraining lasso objective.

Plugging in the model for y, we obtain

$$\sum_{k=1}^K \|X_k(\hat{\beta}_0 - \beta_0^*) - X_k \beta_k^* - \varepsilon_k\|_2^2 \leq \sum_{k=1}^K \|X_k \beta_0^* + \varepsilon_k\|_2^2 + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1). \tag{51}$$

Expanding the square and cancelling common terms we get

$$\sum_k \|X_k \Delta_0\|_2^2 \leq 2 \sum_k (X_k \beta_k^* + \varepsilon_k)^T X_k \Delta_0 + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1), \tag{52}$$

where we defined $\Delta_0 := \hat{\beta}_0 - \beta_0^*$. We apply Cauchy–Schwarz inequality and triangle

inequality to obtain

$$\sum_{k=1}^{K} \|X_k\Delta_0\|_2^2 \leq 2\|\Delta_0\|_1\Big(\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \sum_k \|X_k^T X_k\beta_k^*\|_\infty\Big) + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1) \tag{53}$$

$$\leq 2\|\Delta_0\|_1\Big(\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \sum_k \|X_k^T X_k\beta_k^*\|_\infty\Big) + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1) \tag{54}$$

$$\leq 2\|\Delta_0\|_1\Big(\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \sum_k \|X_k^T X_k\|_\infty\|\beta_k^*\|_1\Big) + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1) \tag{55}$$

$$\leq 2\|\Delta_0\|_1\Big(\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \max_{k\in[K]}\max_{i,j\in[p]}|(X_k^T X_k)_{ij}|\sum_k \|\beta_k^*\|_1\Big) + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1) \tag{56}$$

$$\leq 2\|\Delta_0\|_1\Big(\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \frac{C'n}{K}\sum_k \|\beta_k^*\|_1\Big) + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1) \tag{57}$$

$$\leq 2(\|\beta_0^*\|_1 + \|\hat{\beta}_0\|_1)\Big(\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \frac{C'n}{K}\sum_k \|\beta_k^*\|_1\Big) + \lambda(\|\beta_0^*\|_1 - \|\hat{\beta}_0\|_1). \tag{58}$$

Next, we pick $\lambda \geq 2\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + 2\frac{C'n}{K}\sum_k \|\beta_k^*\|_1$ to obtain the the upper bound on the normalized prediction error

$$\frac{1}{n}\sum_{k=1}^{K} \|X_k\Delta_0\|_2^2 \leq 2\|\beta_0^*\|_1\Big(\frac{1}{n}\big\|\sum_k X_k^T\varepsilon_k\big\|_\infty + \frac{C'}{K}\sum_k \|\beta_k^*\|_1\Big) + \frac{\lambda}{n}\|\beta_0^*\|_1.$$

We apply standard concentration results for the maximum of independent sub-Gaussian variables Vershynin [2018] to control the term $\|\sum_k X_k^T\varepsilon_k\|_\infty$. After simplification we obtain

$$\frac{1}{n}\sum_k \|X_k\Delta_0\|_2^2 \leq \frac{4\|\beta_0^*\|_1 C\sigma\sqrt{\log(p)}}{\sqrt{n}} + \frac{2\|\beta_0^*\|_1 C'\sum_k\|\beta_k^*\|_1}{K} + \frac{\lambda}{n}\|\beta_0^*\|_1 \tag{59}$$

$$\leq \|\beta_0^*\|_1\left(\frac{4C\sigma\sqrt{\log(p)}}{\sqrt{n}} + \frac{2C'\sum_k\|\beta_k^*\|_1}{K} + \frac{\lambda}{n}\right), \tag{60}$$

with probability at least $1-C_3/n$ when we set $\lambda \geq \sqrt{n}2\|\beta_0^*\|_1(2C\sigma\sqrt{\log(p)}+C'\frac{n}{K}\sum_k\|\beta_k^*\|_1)$. The above inequality shows that the prediction error of the pretraining stage, $X\hat{\beta}_0 - X\beta_0^*$ is controlled with high probability.

Next, we analyze the second stage using the same basic inequality argument used

above. We have

$$\|X_k\hat{\beta}_0 + X_k\hat{\beta}_k - y_k\|_2^2 \leq \|X_k\hat{\beta}_0 + X_k\beta_k^* - y_k\|_2^2 + \lambda'(\|\beta_k^*\|_1 - \|\hat{\beta}_k\|_1). \tag{61}$$

Defining $\Delta_k := \hat{\beta}_k - \beta_k^*$ for $k \in [K]$, we simplify the above expression to

$$\|X_k(\Delta_0 + \Delta_k) - \varepsilon_k\|_2^2 \leq \|X_k\Delta_0 - \varepsilon_k\|_2^2 + \lambda'(\|\beta_k^*\|_1 - \|\hat{\beta}_k\|_1). \tag{62}$$

Note that this expression depends on the error of the pretraining stage $X_k\Delta_0$, for which we have established bounds. Expanding the square and simplifying the terms, we obtain

$$\begin{aligned}\|X_k(\Delta_0 + \Delta_k)\|_2^2 &\leq \|X_k\Delta_0\|_2^2 - 2\Delta_0^T X_k^T\varepsilon_k + 2(\Delta_0 + \Delta_k)^T X_k^T\varepsilon_k \\ &\quad + \lambda'(\|\beta_k^*\|_1 - \|\hat{\beta}_k\|_1) \\ &= \|X_k\Delta_0\|_2^2 + 2\Delta_k^T X_k^T\varepsilon_k + \lambda'(\|\beta_k^*\|_1 - \|\hat{\beta}_k\|_1). \end{aligned} \tag{63}$$

We sum the left-hand-side for $k \in [K]$ and obtain.

$$\sum_{k=1}^K \|X_k(\Delta_0 + \Delta_k)\|_2^2 = \sum_{k=1}^K \left(\|X_k\Delta_0\|_2^2 + 2\Delta_k^T X_k^T\varepsilon_k + \lambda'(\|\beta_k^*\|_1 - \|\hat{\beta}_k\|_1)\right).$$

We use the bound $2\sum_k \Delta_k^T X_k^T\varepsilon_k \leq 2\sum_k \|\Delta_k\|_1\|X_k^T\varepsilon_k\|_\infty \leq 2\sum_k(\|\beta^*\|_1 + \|\hat{\beta}\|_1)\max_{k\in[K]}\|X_k^T\varepsilon_k\|_\infty$, where we applied the Cauchy Schwarz inequality twice. Next, we let $\lambda' \geq 2\max_{k\in[K]}\|X_k^T\varepsilon_k\|_\infty$ in order to cancel terms involving $\|\hat{\beta}_k\|_1$, we obtain

$$\sum_{k=1}^K \|X_k(\Delta_0 + \Delta_k)\|_2^2 = \sum_{k=1}^K \|X_k\Delta_0\|_2^2 + 2\sum_k \|\beta_k^*\|_1 \max_{k\in[K]}\|X_k^T\varepsilon_k\|_\infty + \lambda'\sum_k \|\beta_k^*\|_1.$$

Following the concentration bound for the maximum of sub-Gaussian variables as in the analysis of the pretraining stage, $\|X_k^T\varepsilon_k\|_\infty$ is bounded by $2\sigma\sqrt{n/K}\sqrt{\log(pK)}$ with high probability for all $k \in [K]$. We combine the above bound with the error on the pretraining

stage in (59), and obtain

$$\frac{1}{n}\sum_{k=1}^{K}\|X_k(\Delta_0+\Delta_k)\|_2^2 \le \|\beta_0^*\|_1\left(\frac{4C\sigma\sqrt{\log(p)}}{\sqrt{n}}+\frac{2C'\sum_k\|\beta_k^*\|_1}{K}+\frac{\lambda}{n}\right) + \sum_k\|\beta_k^*\|_1\left(\frac{4\sigma\sqrt{\log(pK)}}{\sqrt{nK}}+\frac{\lambda'}{n}\right), \quad (64)$$

with probability at least $1 - C_3/n$ when the regularization parameters satisfy $\lambda \ge \sqrt{n}2\|\beta_0^*\|_1(2C\sigma\sqrt{\log(p)} + C'\frac{n}{K}\sum_k\|\beta_k^*\|_1)$ and $\lambda' \ge 4\sigma\sqrt{n/K}\sqrt{\log(pK)}$. □

B.8 Does cross-validation work here?

In lasso pretraining, "final" cross-validated error that we use for the estimation of both λ and α is the error reported in the last application of `cv.glmnet`. There are many reasons why this estimate might be biased for the test-error. As with usual cross-validation with k folds, each training set has $n - n/k$ observations (rather than n and hence the CV estimate will be biased upwards. On the other hand, in the pretrained lasso, we re-used the data in the applications of `cv.glmnet`, and this should cause a downward bias. Note that we could instead do proper cross-validation— leaving out data and running the entire pipeline for each fold. But this would be prohibitively slow.

We ran a simulation experiment to examine this bias. The results are in Figure 14. The y-axis shows the relative error in the CV estimate as a function of the true test error. The boxplot on the left corresponds to the overall model fit via the lasso: as expected, the estimate is a little biased upwards. The other boxplots show that the final reported CV error is on the order of 5 or 10% too small as an estimate of the test error, Hence this bias does not seem like a major practical problem, but should be kept in mind.

The data were simulated with $n = 500, p = 1000, \text{SNR} = 2.28$, $K = 9$ groups. The class sizes are as follows: there is one group each with $n = 100$ and $n = 80$; two groups with $n = 60$, and 5 groups with $n = 40$. The shared component, β_0, is 1 for the first 10 coefficients and 0 otherwise. In each of the 9 groups, the individual coefficients modify these coefficients: $\{\beta_{1j}, \beta_{2j}, \ldots, \beta_{9j}\} = \{19, 16, 14, 11, 9, 7, 4, 2, 0\}$ for j in 1 to 10. Additionally, the remaining values β_{kj} for $j > 10$ are 0 or 1 with 9 features having the value 1, and the nonzero entries of the β_ks are non-overlapping. A test set of size 5000 was also generated in the same way.

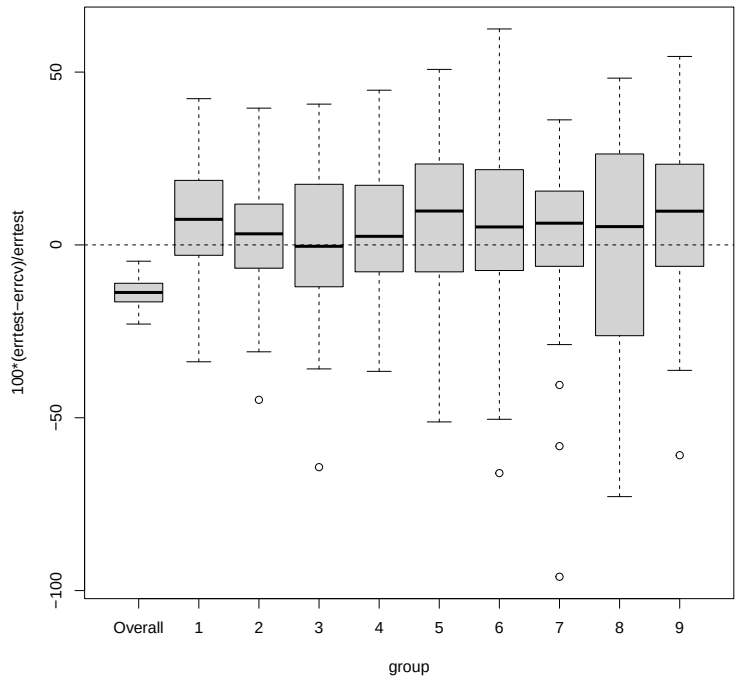

Figure 14: *Relative error of 5-fold CV error, as an estimate of test error (50 simulations). Left boxplot is for the overall model; other boxplots show results for the 9 groups in pretrained lasso. Cross-validation overestimates the test error in the overall model, but underestimates it each of the 9 groups at the end of the two-step pretraining. However, the bias is relatively small in each case.*