Pretraining and the Lasso

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

[1]Department of Biomedical Data Science
[2]Department of Electrical Engineering
[3]Department of Statistics
[4] Department of Biochemistry
[5] Institute for Computational and Mathematical Engineering
Stanford University

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 (which makes the final classification) and train (or "fine tune") those weights on our dataset.[1] 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 an overall model is fit to a large set of data, and then fine-tuned to a specific task on a smaller dataset. This latter dataset can be a subset of the original dataset, but does not need to be. We find that this framework has a wide variety of applications, including stratified models, multinomial targets, 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: Supervised Learning, Pretraining, Lasso, Transfer learning

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 which contains 1000 object classes and 1,281,167 training images. The weights in this fitted network are then fixed, except for the top layer which makes the final classification of dogs vs cats; finally, the weights in this top layer are refitted using our training set of 500 images. This approach is effective because the initial network, learned 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.

[1]Typically only the top-layer is fine-tuned, but more layers can be fine-tuned, if computationally feasible. This is an area of active research.

Our motivating example came from a study carried out in collaboration with Genentech (McGough et al. 2023). The authors curated a large pancancer dataset, consisting of 10 groups of patients with different cancers, approximately 30,000 patients in all. 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 50,000 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 are trained for each class and used to make predictions for that 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 (such as 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 overall (pancancer) model can be blended with individual models in an adaptive way, a paradigm that 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 the result on the TCGA pancancer dataset. Section 3 discusses related work. In section 4 we demonstrate the generality of the idea, detailing a number of different "use cases". We discuss a method for learning the input groups from the data itself in Section 5. In Section 6 we study the performance of the pretrained lasso in different use cases on simulated data. Real data examples are shown throughout the paper, including application to cancer, genomics, and chemometrics. Section 7 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 move beyond linear models in Section 10, illustrating an application of the pretrained lasso to gradient boosting. We examine our use of cross-validation in Section 11 and end with a discussion in Section 12.

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 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 to a weighted norm, taking the form

$$\sum_j \text{pf}_j |\beta_j| \tag{2}$$

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 (i.e., never entered into the model).

2.2 Underlying model and intuition

Suppose we express our data as a feature matrix X and a target 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 $y_1, \ldots, y_K$.

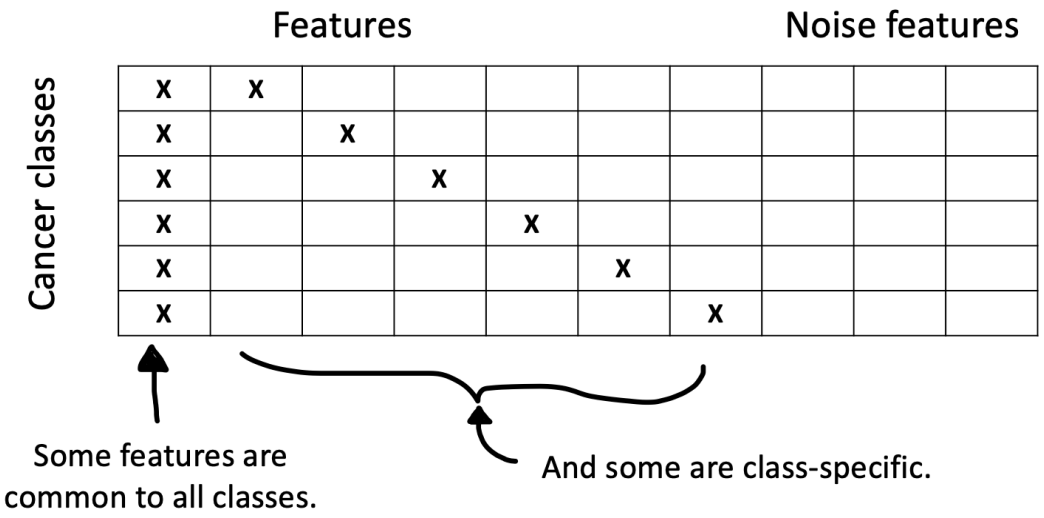

Figure 1: *Conceptual model: some features are predictive for all or most classes, some are specific to each class, and some are noise.*

As shown in Figure 1, we imagine that the features are roughly divided into two types: common features that are predictive in most or all classes, and individual features, predictive in one particular class. Finally there are noise features, with little or no predictive power. 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 class-specific 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, a kind of data-shared lasso (Gross & Tibshirani 2016), has the form:

$$y_k = (\mu_0 + \mu_k) + X_k(\beta_0 + \beta_k) + \varepsilon_k \text{ for } k = 1, 2, \dots K, \tag{3}$$

where y_k is the vector of responses for data in group k. Note that β_0 is shared across all classes k; this is intended to capture the common features. The class specific β_k captures features that are unique to each class, and may additionally adjust the coefficient values in β_0.

We fit this model in two steps. First, we train a model using all the data. We fit an *overall* model:

$$\hat{\mu}_0, \hat{\beta}_0 = \underset{\mu,\beta}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{K} \| y_k - (\mu \mathbf{1} + X_k \beta) \|_2^2 + \lambda ||\beta||_1, \tag{4}$$

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: we find $\hat{\beta}_k$ and $\hat{\mu}_k$ such that

$$\begin{aligned} \hat{\mu}_k, \hat{\beta}_k = \underset{\mu,\beta}{\operatorname{argmin}} \frac{1}{2} \| y_k - (1-\alpha) \left(\hat{\mu}_0 \mathbf{1} + X_k \hat{\beta}_0 \right) - (\mu \mathbf{1} + X_k \beta) \|_2^2 + \\ \lambda \sum_{j=1}^{p} \Big[I(j \in S(\hat{\beta}_0)) + \frac{1}{\alpha} I(j \notin S(\hat{\beta}_0)) \Big] |\beta_j|. \end{aligned} \tag{5}$$

We choose λ through cross validation, and $\alpha \in [0, 1]$ is a hyperparameter. Notice that 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 (illustrated in Figure 4).

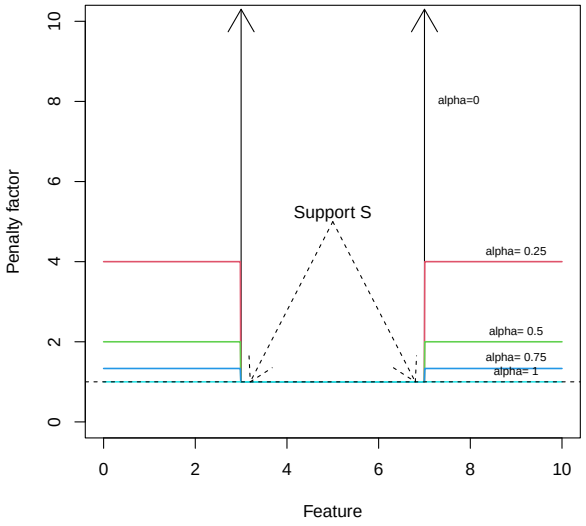

Figure 2: *Features in the support S always have penalty factor 1; off the support, the penalty factor grows as α approaches 0.*

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 class specific model uses the target $y_k - (1-\alpha)\left(\hat{\mu}_0\mathbf{1} + X_k\hat{\beta}_0\right)$. 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 the overall model, and when $\alpha = 1$, the overall model is ignored.

Second, the usual lasso penalty is modified by the penalty factor $I(j \in S(\hat{\beta}_0)) + \frac{1}{\alpha}I(j \notin S(\hat{\beta}_0))$ for each coefficient β_j. This is a function that is 1 on the support of the overall model $S(\hat{\beta}_0)$ and $\frac{1}{\alpha}$ off the support (illustrated in Figure 2). When $\alpha = 0$, the penalty factor is ∞ off the support $S(\hat{\beta}_0)$, and so the class specific model is only able to use features on the support of the overall model. 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 7), 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 model fit at the first step, while the penalty factor captures its support.

2.3 The algorithm

We now summarize the pretrained lasso algorithm discussed above. 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 shown in Figure 3. As described in Section 4, our proposed paradigm is far more general: we first describe the simplest case for ease of exposition.

The procedure is a two step process: first, an *overall* model is fit using all the data. An offset and penalty factor are computed from this model, and these two ingredients are passed on to Step 2, where a *class specific* model is fit to each class. The class specific model is used for prediction in each class. The two steps are given in detail in Algorithm 1.

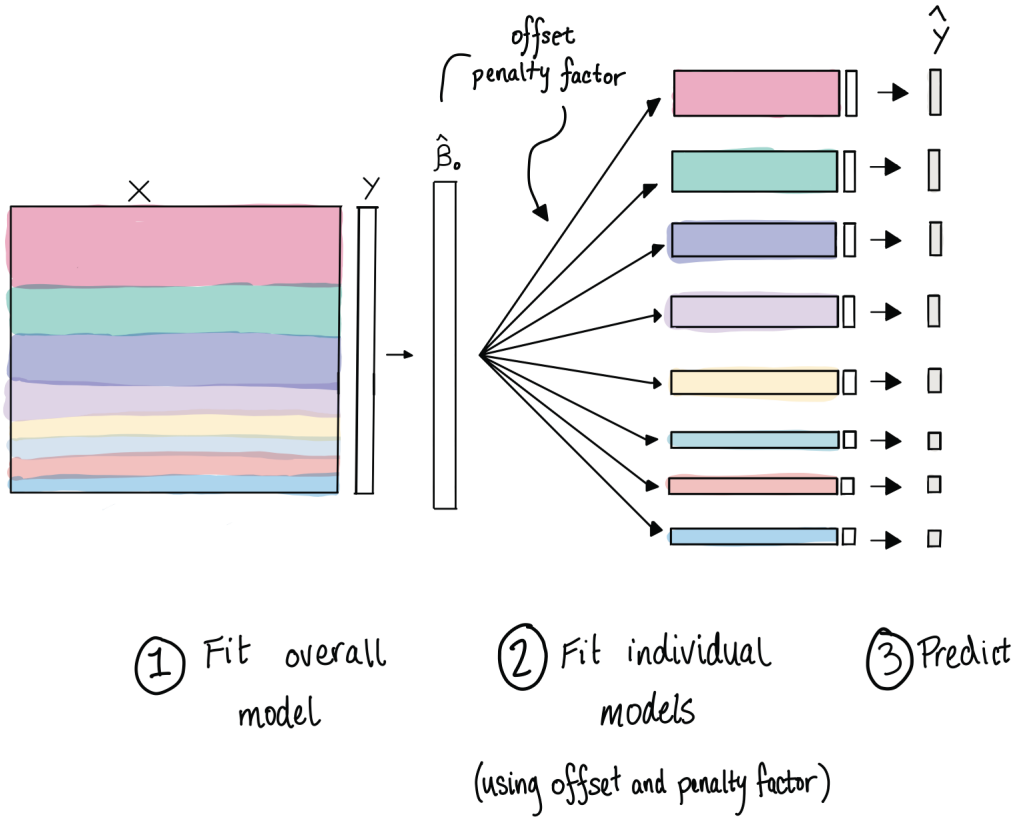

Figure 3: *Workflow for the pretrained lasso.*

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:
   - Compute the linear predictor $X_k\hat{\beta}_0 + \hat{\mu}_0$, and 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` by $\texttt{pf}_j = (1-\alpha) \cdot [I(j \notin S) \cdot \frac{1}{\alpha} + I(j \in S)]$.

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

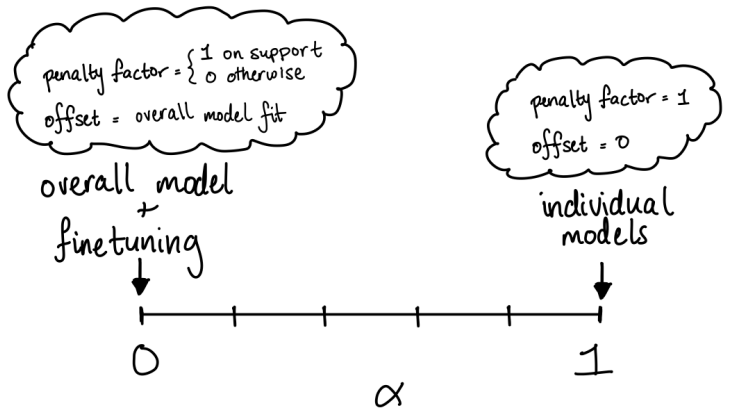

Figure 4: *Spectrum of pretrained lasso models, indexed by the hyperparameter α.*

We again note that when $\alpha = 0$, this 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 specific group. When $\alpha = 1$ the method corresponds to fitting k separate class-specific models. See Figure 4.

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 extremes. We have not proven that this particular 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.

2.4 Simulated example

Figure 5 shows an example with $n = 500, p = 1000, SNR = 2.33$, $K = 9$ groups, and the common coefficients β_0 have different magnitudes in each group (ranging from 20 to 0), and the individual coefficients β_{kj} are 0 or 1 for all k, such that the nonzero entries of the β_ks are non-overlapping. A test set of size 5000 was

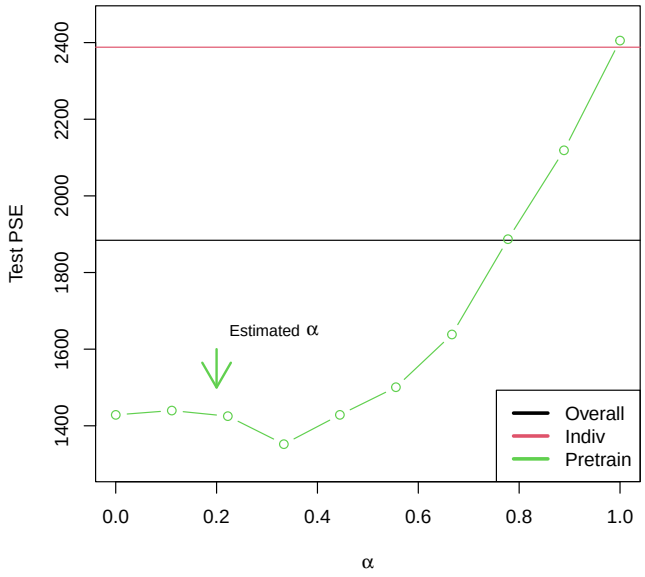

Figure 5: *Results for the pretrained lasso applied to a simulated data set. The arrow indicates the cross-validated choice of α.*

also generated. Shown are the test set prediction squared error for the overall, individual and pretrained lasso models. The arrow indicates the cross-validated choice of α for the pretrained lasso, which achieves the lowest PSE among the 3 methods.

2.5 Pretraining using an external dataset

Here we examine the setting where there is a large external dataset with multiple classes (denoted by D1), and we have a smaller training set (D2) from just one class (say class 1). Our goal is to make accurate predictions for class 1. This follows our analogy to using ImageNet (D1) to train a large neural network, and then to fine tune using a smaller dataset of cats and dogs (D2). We consider four different approaches to this problem: (1) Fit the lasso with cross-validation (`cv.glmnet`) to D2 and use the estimated model to make predictions for class 1; (2) Run the pretrained lasso, using D1, D2 for the two stages; (3) Combine the data from D1 and D2 for class 1, run `cv.glmnet` on this class 1 data, and make predictions; (4) Combine all of D1 with D2, run `cv.glmnet` and make predictions. This is illustrated in Figure 6.

Note that (1) does not require access to D1, while (2) requires just the offset and penalty factor from the lasso model fit to D1. On the other hand, (3) uses class 1 data from D1, while (4) requires *all* of the data D1. We wish to compare approaches (1) and (2), which do not require access to D1, with the other two approaches.

We generated a dataset D1 in exactly the same manner as in the previous example, and in addition, a dataset D2 in class 1 of the same size as class 1 in D1. The MSE results over 50 simulations are shown in the left panel of Figure 7. We find that the pretrained lasso (2) outperforms (1) and (4), and nearly does as well (3), which requires access to D1.

In this example, the data were generated with strong shared effects and weaker individual effects. In the right panel of Figure 7 we have made the problem harder for the pretrained lasso. The individual effects are now stronger than the shared effects, and the overlap in the supports of the coefficients in the overall

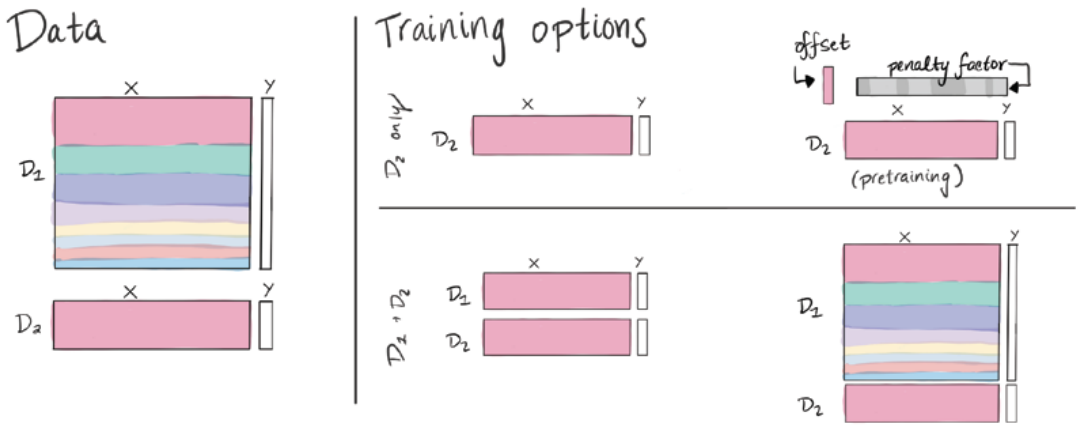

Figure 6: *Options for modeling with an external dataset D1 and a smaller training dataset D2. (The bottom row depicts options when D1 is available at train time.)*

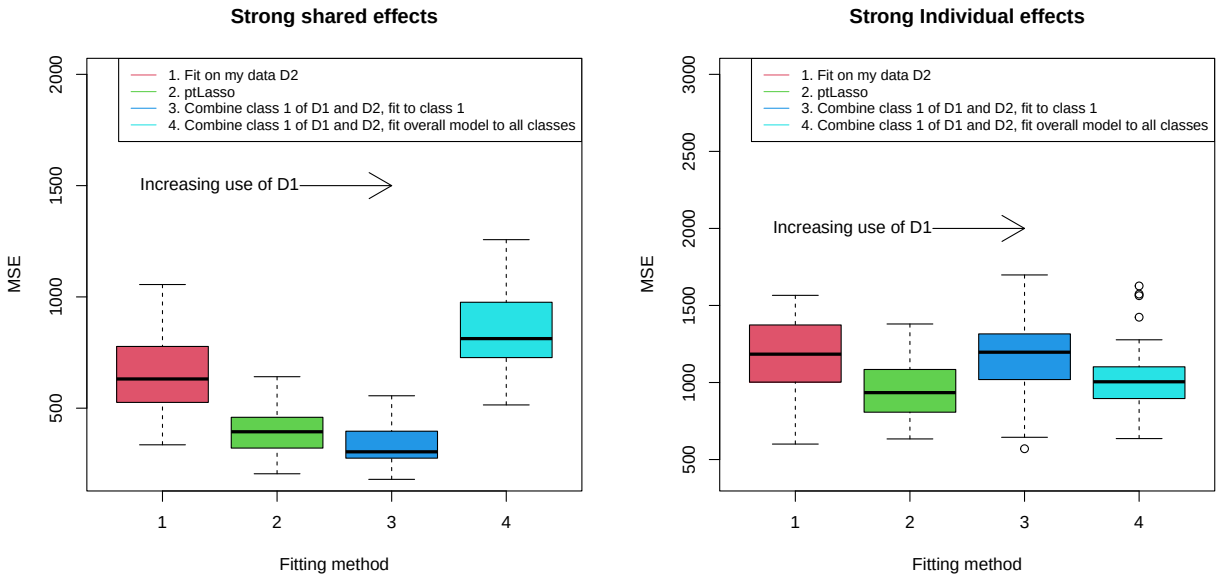

Figure 7: *Comparison of approaches for modeling with a large (usually inaccessible) dataset D1 and a smaller dataset D2. Pretraining using only D2 and the coefficients from a model trained with D1 performs nearly as well as having access to D1 directly.*

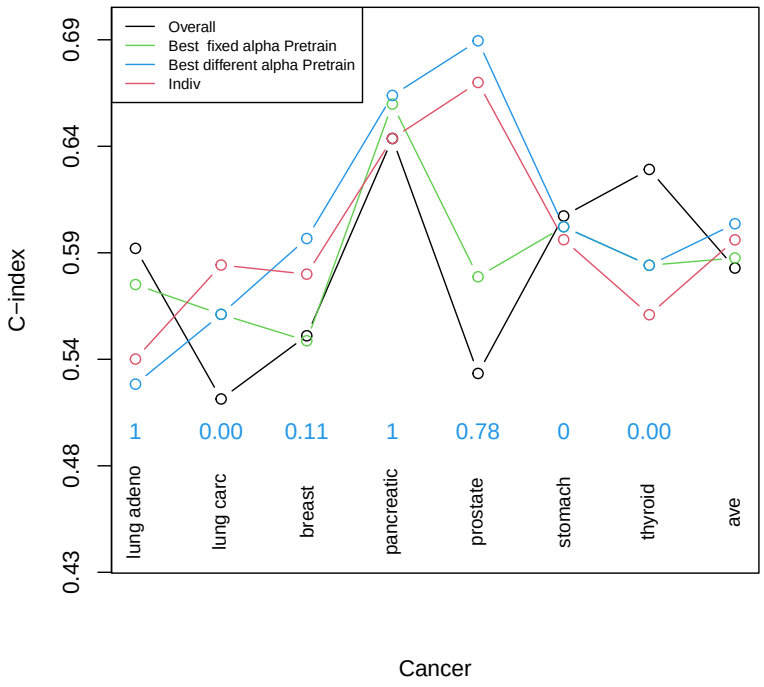

Figure 8: *TCGA dataset: C-index values for different models, and 7 cancer classes; the rightmost points show the average C-index over the 7 classes. The numbers in blue above the cancer class labels indicate the values of α chosen for each class.*

and individual models is only 50%. In addition, the overlapping coefficients are not equal but only agree in sign[2]. We see that the pretrained lasso again outperforms approach 1 and loses narrowly to approach 4, which requires full access to the external data D1.

2.6 Example: TCGA PanCancer dataset

At the time of this writing, for logistical reasons, we have not yet applied lasso pretraining to the Genentech pancancer dataset discussed earlier. Instead we applied it to the public domain TCGA pancancer dataset (Goldman et al. 2020). After cleaning and collating the data, we were left with 4037 patients, and 20,531 gene expression values. The patients fell into one of 7 cancer classes as detailed in Table 1. We used a CART survival tree to pre-cluster the 7 classes into 3 classes: ("breast invasive carcinoma", "prostate adenocarcinoma" , "thyroid carcinoma"), ("lung adenocarcinoma", "lung squamous cell carcinoma", "stomach adenocarcinoma") and "pancreatic adenocarcinoma"

The outcome was PFS (progression-free survival): there were 973 events. For computational ease, we filtered the genes down to the 500 genes having the largest absolute Cox PH score: all methods used this filtered dataset. The data was divided into a training (50%), validation set (25%) and a test set (25%). The validation set was used to select the best values for α.

Figure 8 shows the test set C-index values for a number of methods. Table 1 shows the number of non-zero genes from each cancer class, for each model.

We see that the pretrained lasso provides a clear improvement as compared to the overall model, and a small improvement over the individual models. From a biological point of view, the separation of genes into common and cancer-specific types could aid in the understanding of the underlying diseases.

[2]This scenario is discussed in Section 7

	lung adeno	lung carc	breast	pancreatic	prostate	stomach	thyroid	Total
Sample size	283	281	593	98	279	202	282	2018
Overall								25
PreTrain	50	15	15	1	3	1	8	93
Indiv	5	7	1	1	1	19	1	35

Table 1: *Sample size (number of patients) and number of non-zero genes in each fitted model. There was no overlap between the genes selected by the overall model and the 2nd stage of the pretrained lasso. The pretrained lasso with α chosen separately for each class, does the best by a small margin.*

Remark 4. Suppose we had more than one pancancer dataset (say 10), and as above, we want to predict the outcome in one particular cancer. Emmanuel Candès suggested that one might repeatedly sample one of the 10 datasets, and apply the pretrained lasso to each realization. In this way, one could obtain posterior distributions of model parameters and predictions, to account for the variability in pancancer datasets.

Remark 5. Pretraining may be useful when the input groups share overlapping — but not identical — features. For example, some features may be measured for just one cancer; this feature may then be used only in the individual model for that cancer (stage 2 of pretraining), while the overall model (stage 1) uses features shared by all groups.

3 Related work

3.1 Data Shared Lasso

Data Shared Lasso (Gross & Tibshirani 2016) (DSL) is a closely related approach for modeling data with a natural group structure. It solves the problem

$$\left(\hat{\beta}, \hat{\Delta}_1, \ldots, \hat{\Delta}_K\right) = \operatorname{argmin} \frac{1}{2} \sum_i \left(y_i - x_i^T(\beta + \Delta_{k_i})\right)^2 + \lambda \left(\|\beta\|_1 + \sum_{k=1}^{K} r_k \|\Delta_k\|_1 \right). \tag{6}$$

That is, 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 an "overall" model 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 more flexible: for example, 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 & 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 features because they are both blood cancers, and we could prespecify this when fitting a laplacian regularized stratified model. So, while pretrained lasso uses information from an *overall* model, laplacian regularized stratified modeling uses known similarities between individual *groups*.

3.3 Reluctant Interaction Modeling

Reluctant Interaction Modeling Yu et al. (2019) is a method to train 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 uses a similar algorithm: train an initial model, and then train a second model to the residual from the first, using a subset of features.

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

$$y = X\beta + Z\theta + \varepsilon, \tag{7}$$

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 a dataset with fixed input groupings: 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, with 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.

The common feature of these different "use cases" is the passing of an offset and penalty factor from one model to the next.

Here is a (non-exhaustive) list of potential use cases:

1. **Input grouping:**
   The rows of X are partitioned into groups. These groups may be:
   (a) Pre-specified (Section 2.3), 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 4.3), e.g. different train and test patients.
   (c) Learned from the data via a decision tree (Section 5).
2. **Target grouping:**
   Here, there is a natural grouping on the target y, and y may be:
   (a) binomial or multinomial, and there is one group for each response class.
   (b) multi-response: y is a *matrix*, and there is one group for each column of y (Section 4.1). Two special cases: time-ordered columns, where the same target variable is measured at different points in time, and mixed targets, where the different target columns are of different types, e.g. quantitative, survival, or binary/multinomial. In both cases, the pretrained lasso is applied to each target column in sequence. This is illustrated in Section 8.

3. **Both input and target groupings:** Suppose for example the target y is 0-1, multinomial or multi-response, *and* there is a separate grouping on the rows of X, e.g. the rows of X are stratified into age groups, and we want to predict cancer class.

4. **Conditional average treatment effect estimation**. This is similar to the input grouping case (#1 above). Here the groups are defined by the levels of a treatment variable (Section 9).

5. **Unlabelled pretraining:** Given unlabelled pretraining data, we can use sparse PCA to estimate the support, and use the first principal component as the offset.

Of course, other scenarios are possible.

4.1 Target grouping: binomial, multinomial or multiresponse target

We begin by describing the multinomial (or binomial) setting, and then we extend this to the multiresponse case. 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. Figure 9 shows an overview of our two-stage procedure. It is much the same as our earlier algorithm, the only difference being the way in which the models are combined at the end. Here is the procedure in detail:

Algorithm 2 Pretrained Lasso with target groups

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. use the penalty $\sum_{j=1}^{p} \|\beta_{j,\cdot}\|_2$).

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 $I(j \in S_k) + (1/\alpha)I(j \notin S_k)$. 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.

This is applied to real data in Section 4.2 next.

When the target is multi-response, the procedure is nearly identical. At the first stage, we again fit a grouped multinomial model to all classes. Then at the second stage, we fit a separate model for each *column* of y, using the corresponding offset and support from the first stage as described earlier.

4.2 Example: classifying cell types with features derived from the SPLASH algorithm

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 to analyzing genomic sequencing. SPLASH is directly run on 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 train and test, and we used cross validation to select the pretraining hyperparameter α. Results across a range of α values are shown in Figure 10. We find that, for most values of α, pretraining outperforms the overall and individual models.

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

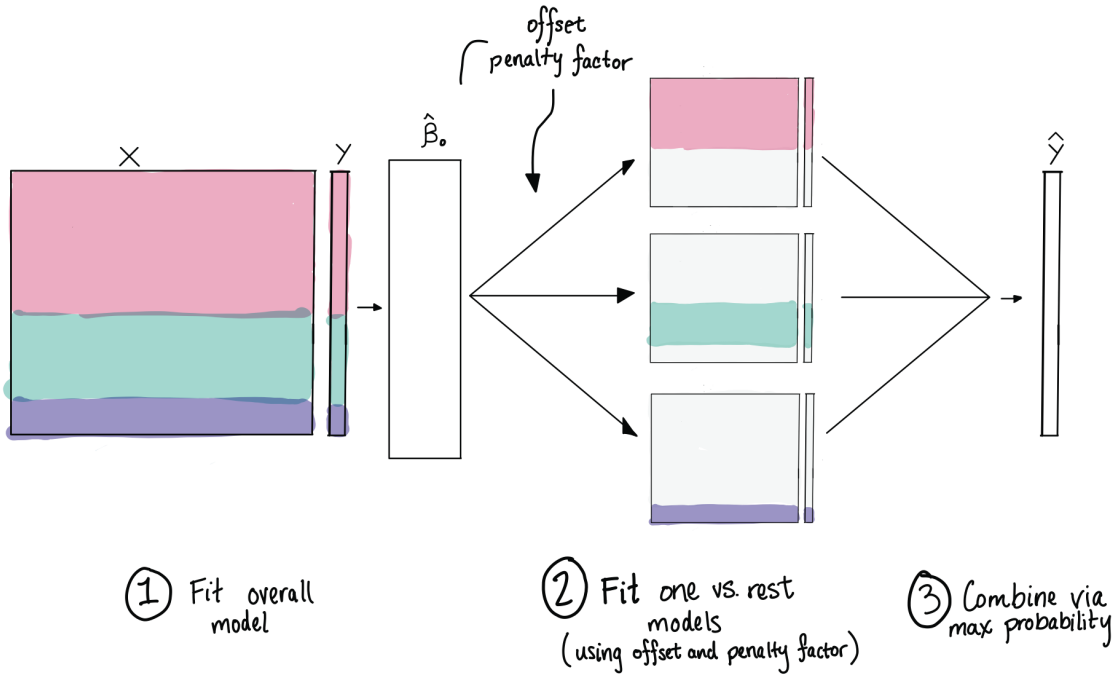

Figure 9: *Workflow for pretrained lasso applied to multinomial data.*

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.

4.3 Different groupings in the train and test data

In the settings described above, our training data are naturally 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 previously described. Now, however, we fit an extra model to predict the *training group* for each observation. This is a multinomial classifier, and for each new observation, it returns a vector of probabilities describing how similar the observation is to each training group. Now, at prediction time for a new observation, we first make a prediction using each of the K pretrained models to obtain a prediction vector $\hat{y}$. Then, we use the multinomial classifier to predict similarity to each of the K training groups; this results in a K-vector $\hat{p}$. Our final prediction is $\hat{y} \cdot \hat{p}$, a weighted combination of $\hat{y}$ and $\hat{p}$. This procedure is illustrated in Figure 11, described in detail in Algorithm 3 and applied to real data in Section 4.4 below.

Although expression (8) makes sense mathematically, we have often found better empirical results if we instead train a supervised learning algorithm to predict $r(x_i)$ from $\hat{p}_k(x_i)$ and $\hat{q}(x_i), k = 1, 2, \ldots K$.

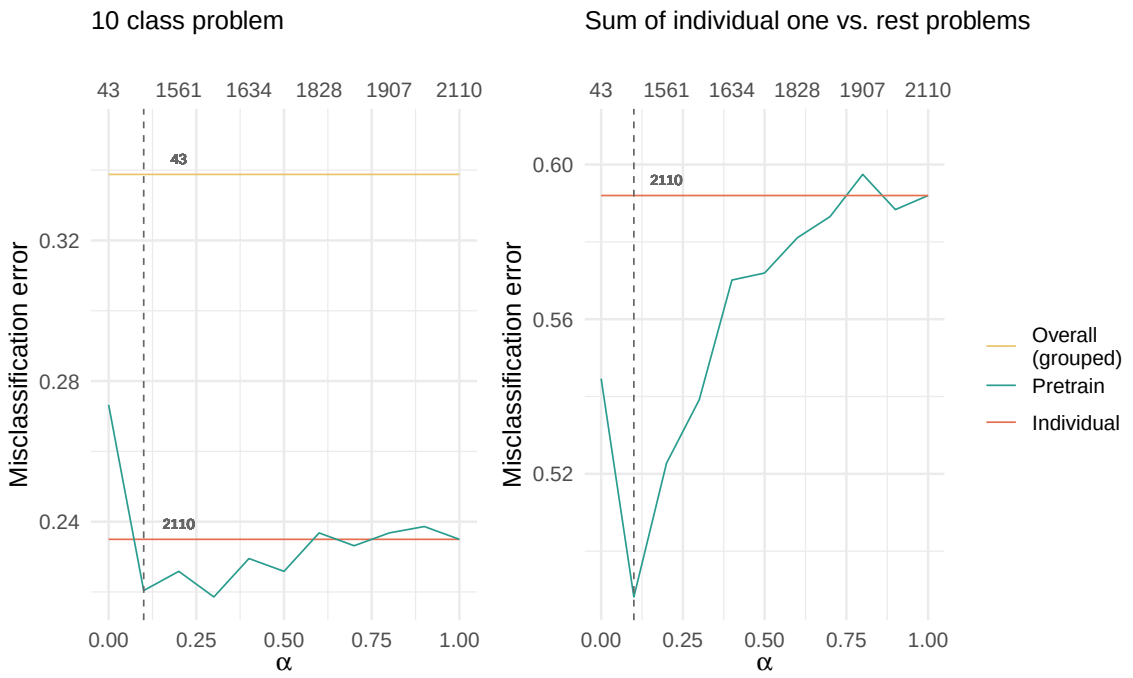

Figure 10: *Performance of cell type classification on held-out data. The vertical dashed line shows the value of the hyperparameter α chosen by cross validation*

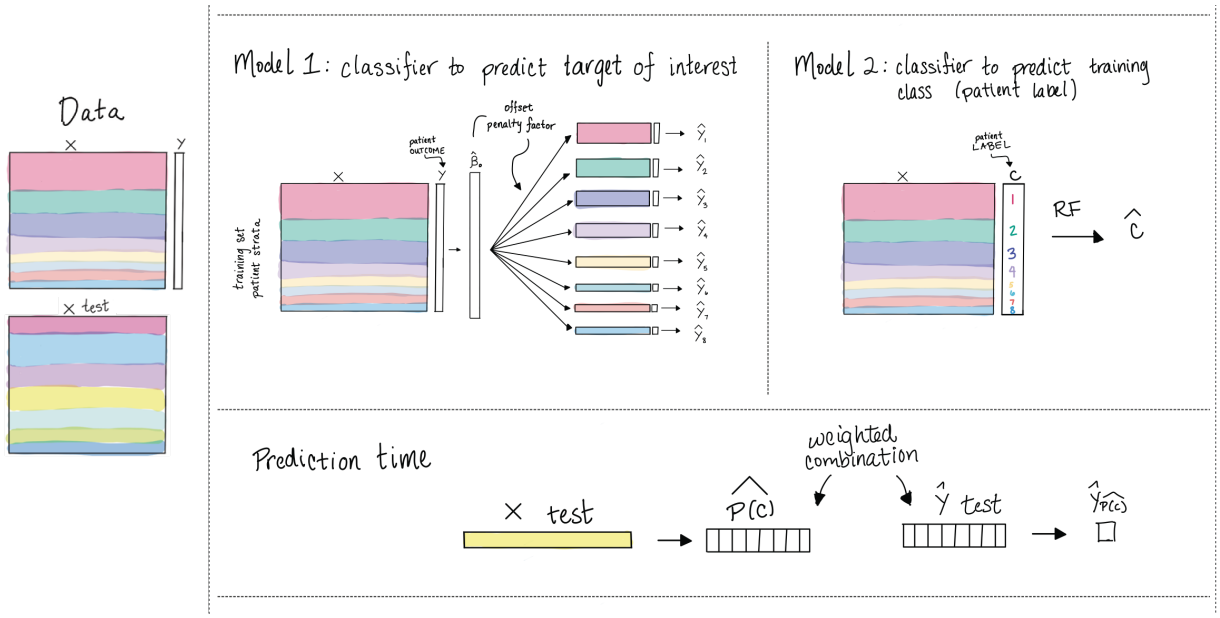

Figure 11: *How to use pretraining when the train and test strata are different (Section 4.3).*

Algorithm 3 Pretrained Lasso: different input groupings in the training and test data

For simplicity assume that the target y is binary, there are $k = 1, 2, \ldots K$ groups in the training set and $g = 1, 2, \ldots G$ (different) groups in the test set.

1. Apply Algorithm 1 to yield individual models for each input group. Let the estimated probabilities $P(Y = 1 \mid x)$ in group k be $\hat{p}_k(x)$.

2. Fit a separate model to predict the training group from the features, using for example a random forest. Let $\hat{q}_k(x)$ be the resulting estimated class probabilities for $Y = 1|x$ for classes $k = 1, 2, \ldots K$,

3. Given x_i, the feature vector for a test observation, compute

$$\hat{r}(x_i) \sim \sum_k \hat{p}_k(x_i)\hat{q}_k(x_i), \tag{8}$$

the estimated probability that $Y = 1|x_i$ for test observation x_i.

4.4 Mass spectrometry cancer data

This data comes from a proteomics study of melanoma (Margulis et al. 2018). A total of 2094 peak heights from DESI mass spectometry were measured for each image pixel, with about a thousand pixels measured for each patient. There are 28 training patients, 15 test patients; a total of 29,107 training pixels, 20,607 test pixels. There is an average of about 1000 pixels per patient. The output target is binary (healthy vs disease). All error rates quoted are per pixel rates.

We clustered the training patients using K-means into 4 groups (see Table (2).

Cluster	Members
1	3 4 7 10 11 12 16 17 19
2	1 2 20 23 24 25 26 27 28
3	15 21 22
4	5 6 8 9 13 14 18

Table 2: *Melanoma data: Clusters of training patients*

Tables 3, 4 and Figure 12 show the test error and AUC results. We see that the pretrained lasso provides a small advantage in AUC as compared to the overall model.

Cluster	CV-AUC Pretrain	AUC Pretrain
1	0.932	0.945
2	0.973	0.887
3	0.938	0.929
4	0.955	0.930

Table 3: *Melanoma data: pretrained lasso CV and test set AUCs for each cluster.*

Remark 6. *Pretrained lasso fits an interaction model.* In general, suppose we have a target variable y, features x and grouping variables $G_1, G_2, \ldots G_g$, As illustrated above, 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 the O_1 and pf_1 be the offset and penalty factor from the chosen model.

Method	Test AUC
Overall model	0.940
Pretrained lasso using (8)	0.935
Pretrained lasso using supervised learner to predict $r(x_i)$	0.960

Table 4: *Melanoma data: test AUCs. In the third line, we used the lasso trained on* $\hat{p}_k(x_i), k = 1, 2, \ldots K$ *and* $\hat{q}_k(x_i), g = 1, 2, \ldots K$ *and their products to predict* $r(x_i)$

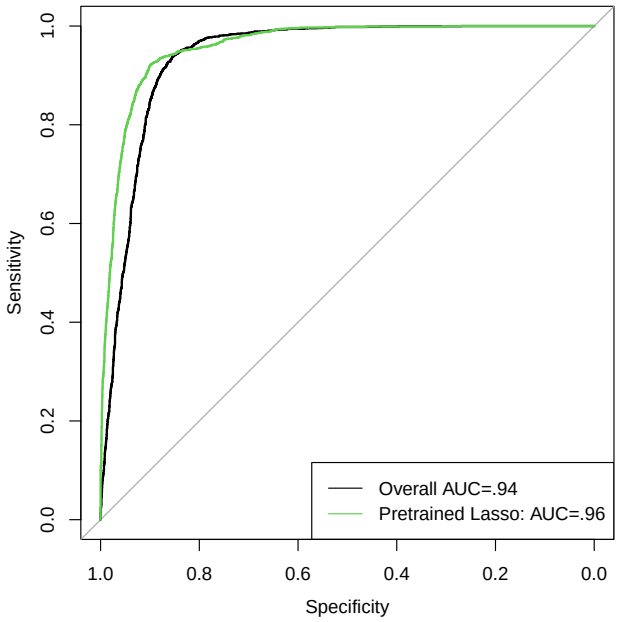

Figure 12: *Melanoma data: ROC curves*

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 levels $k = 1, 2, \ldots K_2$.

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.

Remark 7. *Input "grouping" with a continuous variable:* Instead of a discrete grouping variable G, suppose that we have a continuous modifying variable such as age. Here, we can use the pretrained lasso idea as follows:

1. At the first stage, train a model using all rows of the X, and without use of G.

2. At the second stage: fit a model again using all rows of X, but now multiply each column, and the offset, by G. Use the offset and penalty factor from the first model as defined in Section 2.3.

In the second stage, we force an interaction between age and the other features. This mirrors the case where the grouping variable is discrete; fitting separate models for each group is an interaction between the grouping variable and all other variables.

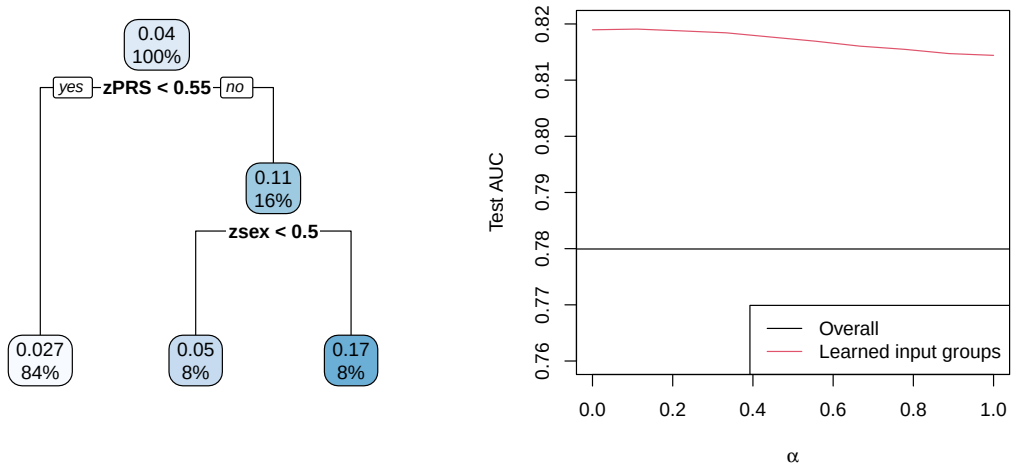

Figure 13: *Left: CART tree learned from the partitioning features PRS, Sex and Age. Right: test set AUC for overall (no groups) model and pretrained lasso applied to the 3 groups (terminal nodes in left panel).*

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 tree. Typically, the features that we make available for splitting are not the full set of features x but instead a small set of clinical variables that are meaningful to the scientist.

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

We split the data into two equal parts at random (train/test) and built a CART tree using the R package `rpart`, limiting the depth of the tree to be 3 (for illustration). The left panel of Figure 13 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 tests AUC for the pretrained lasso and the overall model (an ℓ_1-regularized logistic regression) is shown in the right panel. We see that the pretrained lasso delivers about a 4-5% AUC advantage, for all values of α.

Figure 14 displays a heatmap of the non-zero coefficients within each of the three groups, and overall.

Another way to grow the a decision tree in this procedure would be to use "Oblique Decision Trees", implemented in the ODRF R language package.[3] 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 very similar tree to that from CART, and hence we omit the details.

[3]We thank Yu Liu and Yingcun Xia for implementing changes to their R package ODRF so that we could use it in our setting.

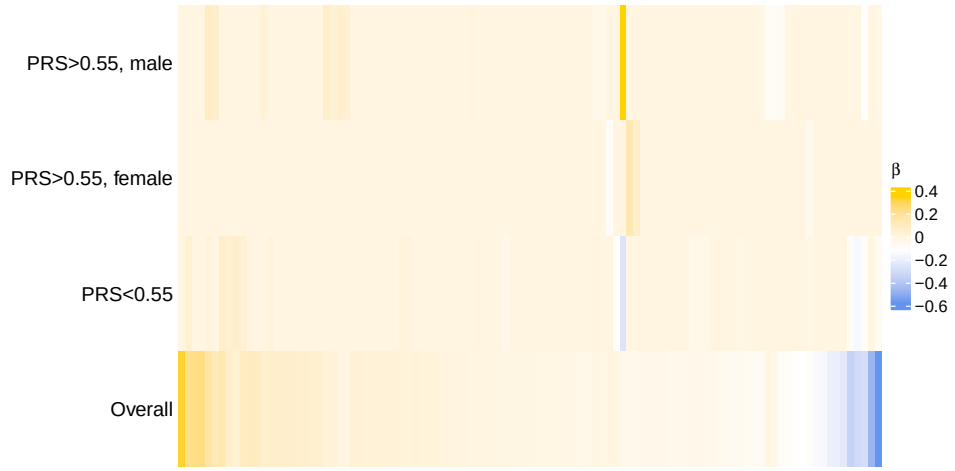

Figure 14: *Coefficients of pretraining features learned from U.K. Biobank example. The vertical yellow-blue stripe shows a strong interaction of one feature with PRS and sex.*

6 Simulation studies

Here, we use simulations to compare pretraining with the overall model and individual models. The three approaches are compared in terms of (1) their predictive performance on test data and (2) their F1 scores for feature selection. Additionally, we compare pretraining to the individual models in terms of their ability to recover the *common* features; the features shared across all groups. Pretraining naturally identifies common features (using those identified in the first stage of training). For the individual model, we define common features as those which are selected for at least 51% of the group-specific models.

We focus here on data with a continuous response (Figures 15, 16 and 17) and five input groups (with $n = 100$ observations each), across a range of signal-to-noise ratios (SNRs). In the first simulation, we create data with a common support and group-specific features, where the *magnitude* of the coefficients in the common support differs across groups. Then, we simulate data with a common support only (same magnitude), and finally we simulate data with individual features only. In the latter two cases, we expect the overall model and the individual models respectively to have the best performance. In general, we find that pretraining outperforms the overall and individual models when our assumptions are met: when there are features shared across all groups and features specific to each individual group. Further, pretraining has a particular advantage when the shared features have different *magnitudes* in each group.

We share the results from a more complete simulation study in Appendix A. There, the simulations cover three settings: grouped data with a continuous response (Table 5), grouped data with a binomial response (Table 6), and data with a multinomial response (Table 7).

7 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 training data.

7.1 Preliminaries

We call a random variable Y sub-Gaussian if it is centered, i.e., $\mathbb{E}[Y] = 0$ and

$$\mathbb{E}\left[e^{sY}\right] \leq e^{\frac{\sigma^2 s^2}{2}}, \quad \forall s \in \mathbb{R}, \tag{9}$$

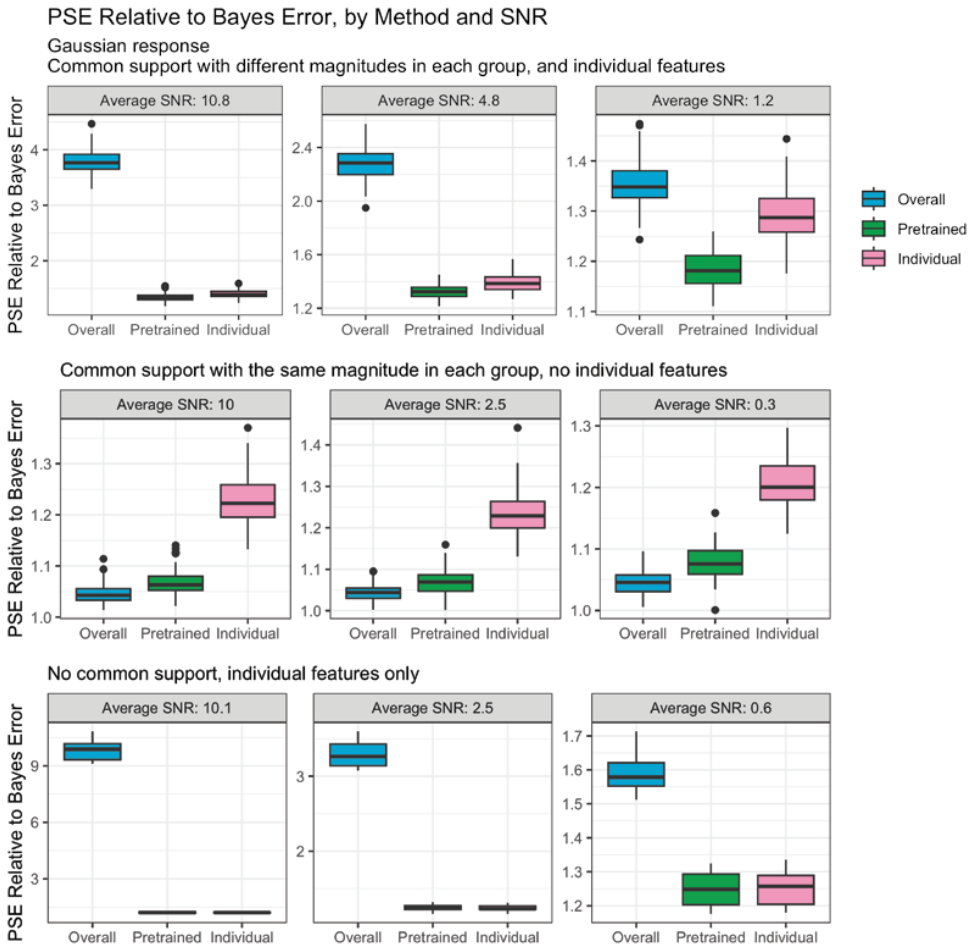

Figure 15: *Prediction squared error, relative to the Bayes error, across* 100 *simulations. Data with a continuous response and five input groups, each with* 100 *observations*

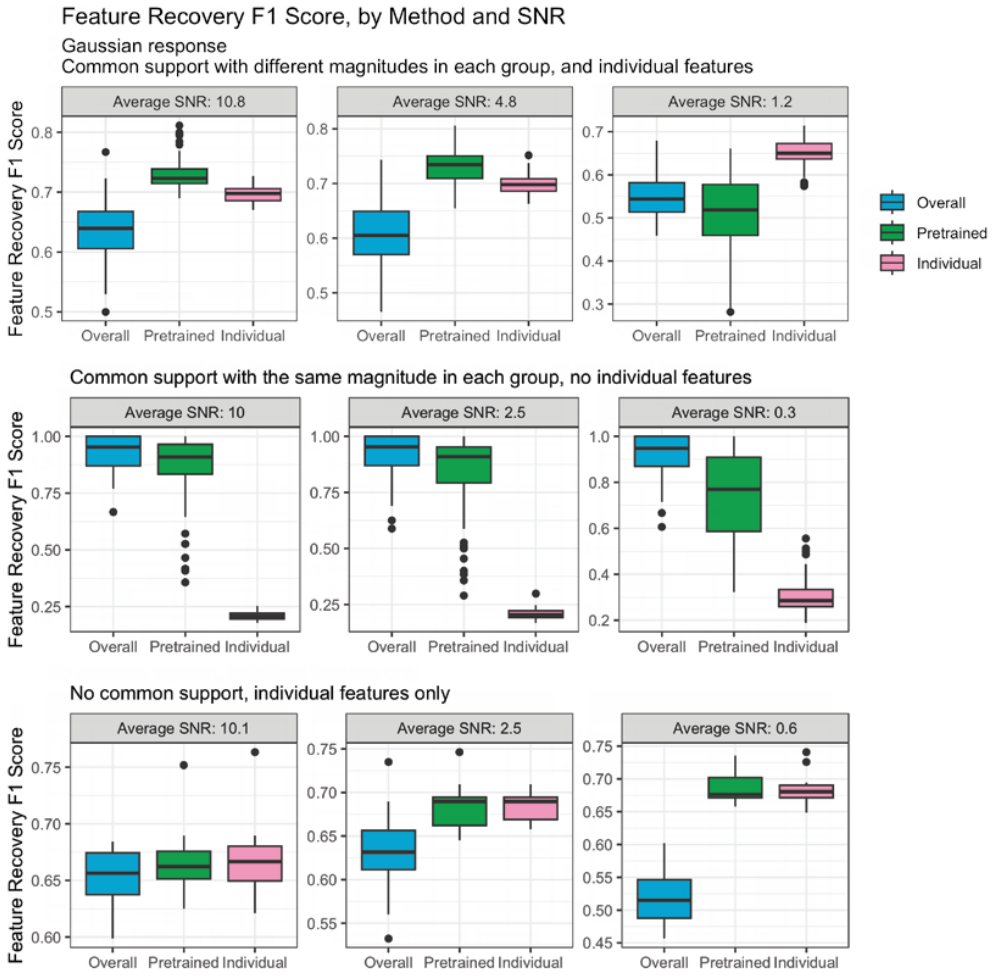

Figure 16: *F1 score for feature recovery: true positives are features selected by the lasso that truly have a nonzero value and true negatives are features that were correctly not selected. Data with a continuous response and five input groups, each with* 100 *observations*

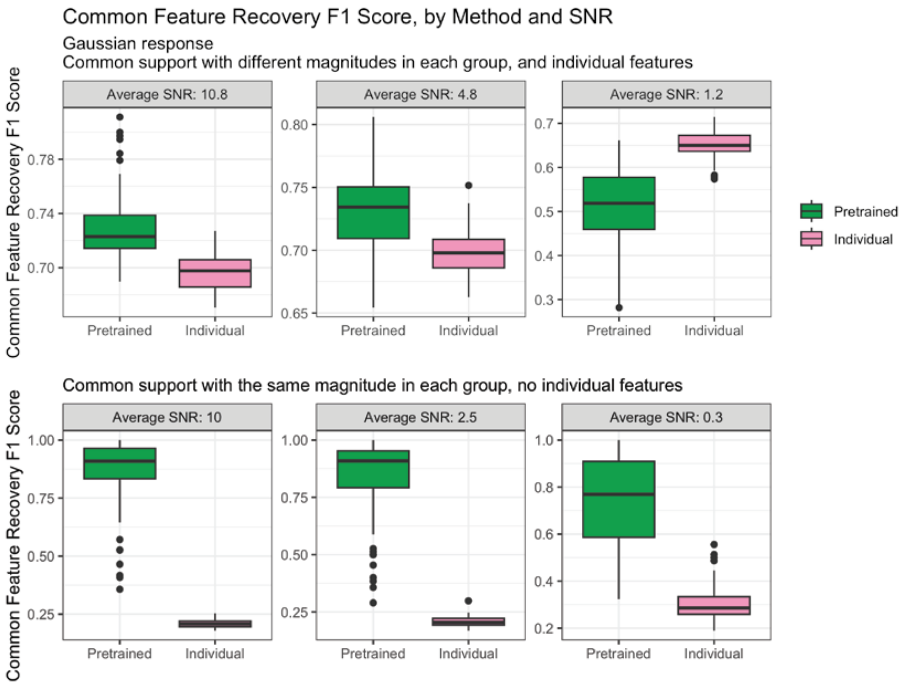

Figure 17: *F1 score for feature recovery on the common features only. Data with a continuous response and five input groups, each with* 100 *observations*

where σ^2 is called the variance proxy of Y. For such a random variable, the following tail bound holds

$$\mathbb{P}\left[|Y| > t\right] \leq 2e^{-t^2/(2\sigma^2)} \qquad \text{for all } t > 0. \tag{10}$$

We call that 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 varible that is zero-mean and bounded by a constant. We use the notation $\lesssim$ and $\gtrsim$ to denote inequality relations up to a constant factor. Specifically, for two quantities a and b, $a \lesssim b$ means that there exists a constant $C > 0$ such that $a \leq Cb$. Similarly, $a \gtrsim b$ means that there exists a constant $C > 0$ such that $a \geq Cb$.

7.2 An overview of the theoretical results

7.2.1 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 pretraining 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.

In Section 7.5, we analyze the in-sample mean-squared error of our two-stage procedure and establish an upper-bound that holds for any deterministic design. This upper-bound is composed of two distinct

components The first component diminishes as the total number of samples increases, while the second component decreases as the number of groups grows. This result highlights the importance of both sample size and group structure in achieving accurate predictions.

7.2.2 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 (17) 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 (11) 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 (11) 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 (11) to lift the assumption that the supports are common, and consider the common and individual support model (22). 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.

7.3 Shared support model

Consider K sets of observations

$$y_k = X_k\beta_k^* + \varepsilon_k \in \mathbb{R}^{n/K}, \tag{11}$$

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

$$\hat{\beta}_{\text{pre}} = \arg\min_{\beta} \frac{1}{n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1. \tag{12}$$

The individual models are defined as

$$\hat{\beta}_i = \arg\min_{\beta} \frac{K}{n}\|y_i - (1-\alpha)X_i\hat{\beta}_{\text{pre}} - X_i\beta\|_2^2 + \sum_{j=1}^{P}\lambda_j|\beta_j|\,, \tag{13}$$

for all i. Here, we require an appropriate choice of the regularization weights $\lambda_1, ..., \lambda_P$.

7.3.1 Pretraining irrepresentability conditions

We now provide a set of deterministic conditions which 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 & Yu 2006), restricted isometry property (Van De Geer & Bühlmann 2009) or random designs (Wainwright 2009) are not applicable due to the mixed observation model in (11).

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*

$$\|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S^*)\|_\infty < \frac{1}{2} \tag{14}$$

$$\|X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X \beta_k^*\|_\infty \leq \frac{\lambda}{4}\,, \tag{15}$$

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

$$\|X_{S^c}^T P_S^\perp \varepsilon\|_\infty \leq \frac{\lambda}{4} \tag{16}$$

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 8. When $K = 1$ and $D_1 = I$, the conditions (14) and (15) simplify to the well-known strong irrepresentability condition (Zhao & Yu 2006), noting that $P_S^\perp D_k X = P_S^\perp X = 0$ which shows (15) always holds.

7.3.2 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:

$$\text{(subgroup isometry)} \qquad \left\| (X_S^T X_S)^{-1} (X_S^T D_k X_S) - \frac{1}{K} I \right\|_2 \leq \delta \; \forall k \in [K], \tag{17}$$

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 9. Note that we have $(X_S^T X_S)^{-1}(X_S^T D_k X_S) = \left(\sum_{i \in [n]} \tilde{x}_i \tilde{x}_i^T \right)^{-1} \left(\sum_{i \in \mathcal{G}_k} \tilde{x}_i \tilde{x}_i^T \right)$, 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* (17) *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* (17) *and the conditions of Lemma 1, the pretraining estimator satisfies*

$$\left\| \hat{\beta}_{\text{pre}} - \frac{1}{K} \sum_{k=1}^K \beta_k^* + \lambda (X_S^T X_S)^{-1} \mathbf{sign}(\beta_S^*) \right\|_2 \leq \delta \sum_{k=1}^K \|\beta_k^*\|_2 + \|X_S^\dagger n\|_2. \tag{18}$$

Remark 10. 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 standard optimality conditions for the Lasso model, as detailed in Wainwright (2009) (see Appendix).

7.3.3 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*

$$n \geq C_1 \max(1, \gamma^2)\, s \log(p - s), \tag{19}$$

for some constant $C_1 > 0$. Then, the conditions (14) and (15) 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 11. It is instructive to compare the condition (19) 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, i.e., taking $\alpha = 1$ in (13), 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 12. 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 (15). 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 $x_1, \ldots, x_n \in \mathbb{R}^p$ 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 $|(\beta_k^*)_j| \leq \gamma \, \forall k \in [K] \, \forall j \in [p]$ for some $\gamma > 0$, and the number of samples satisfy*

$$n \geq C_1 \max(1, \gamma^2)\, s \log\big(\max(p - s, K)\big), \tag{20}$$

and

$$\min_j |\big(\frac{1}{K}\sum_{k=1}^{K}\beta_k^*\big)_j| \geq C_1' \sigma \sqrt{\frac{\log(p-s)}{n}}, \tag{21}$$

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 $1 - C_3 e^{-C_4 n}$ where C_λ, C_2, C_3, C_4 are constants.

Remark 13. The condition (21) 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 14. 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 (21) 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.

7.4 Common and individual support model

We now consider K sets of observations

$$y_k = X_k\beta_0^* + X_k\beta_k^* + \varepsilon_k \in \mathbb{R}^{n/K}, \tag{22}$$

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.

7.4.1 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*

$$n \geq C_5 \max(1, \gamma_1^2)\, s \log\big(\max(p-s, K)\big), \tag{23}$$

and

$$\min_j |\big(\beta_0^*\big)_j| \geq C_5' \sigma \sqrt{\frac{\log(p-s)}{n}}. \tag{24}$$

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 15. 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.

7.5 Prediction 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

$$y_k = (1-\alpha) X_k \beta_0^* + X_k \beta_k^* + \varepsilon_k \in \mathbb{R}^{n/K} \quad \text{for } k \in [K], \tag{25}$$

where $\alpha \in [0, 1]$ is a fixed parameter. Let us denote the full feature matrix $X = [X_1^T, ..., X_k^T]^T \in \mathbb{R}^{n \times p}$. We now switch to the ℓ_1-norm constrained version of Lasso in order to provide a tighter control on the magnitudes of the learned parameters. Define the pretraining estimator as

$$\hat{\beta}_0 \in \arg \min_{\beta_0 :\ \|\beta_0\|_1 \leq R} \sum_{k=1}^{K} \|X_k \beta_0 - y_k\|_2^2,$$

where $R > 0$ is a hyperparameter that controls the ℓ_1 regularization. Consequently, we fit the individual models using $\hat{\beta}_0$ as an offset term

$$\hat{\beta}_k \in \arg \min_{\beta_k :\ \|\beta_k\|_1 \leq R_k} \|(1-\alpha) X_k \hat{\beta}_0 + X_k \beta_k - y_k\|_2^2,$$

for $k = 1, ..., K$ for a fixed value of the offset weight $\alpha \in [0, 1)$ and some $R_k > 0$, $k = 1, ..., K$. We assume that $R, R_1, ..., R_K$ are sufficiently large to ensure that $\|\beta_0^*\|_1 \leq R$ and $\|\beta_k^*\|_1 \leq R_k$ for all $k \in [K]$. Note that exact values of $\{\|\beta_k^*\|_1\}_{k=1}^K$ are not required. An overestimation of their ℓ_1-norms is sufficient.

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', 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 $k \in [K]$ and $\sum_{k=1}^{K} R_k \leq C''$. Set $\alpha = 1/2$. Then, we have the following in-sample prediction error bound*

$$\frac{1}{n}\sum_{k=1}^{K} \|X_k(\frac{\hat{\beta}_0 - \beta_0^*}{2} + \hat{\beta}_k - \beta_k^*)\|_2^2 \leq \frac{\sigma(RC\sqrt{\log p} + 8C''\sqrt{\log(pK)/K})}{\sqrt{n}} + \frac{RC'C''}{2K}, \tag{26}$$

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

Remark 16. *The prediction error consists of two components: one that decreases to zero as the total number of samples (n) increases to infinity, and another that decreases to zero as the number of groups (K) approaches infinity. It is important to observe that the constraint $\sum_{k=1}^{K} R_k \leq C''$ implies that the true individual parameters β_k^* have small ℓ_1 norms. For instance, a scaling of $\|\beta_k^*\|_1 = \mathcal{O}(\frac{1}{K})$ for all $k \in [K]$ is one example that satisfies this condition. Such a scaling assumption is unavoidable 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 17. *The above prediction error can be compared with the individual Lasso estimators $\tilde{\beta}_k$ fitted to the data X_k, y_k for each $k \in [K]$, which corresponds to setting $\alpha = 1$ in our procedure. 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))*

$$\frac{1}{n}\sum_{k=1}^{K} \|X_k(\tilde{\beta}_k - \beta_0^* - \beta_k^*)\|_2^2 \lesssim \frac{(\sqrt{K}R + C''/\sqrt{K})\sqrt{\log p}}{\sqrt{n}}, \tag{27}$$

which holds with the same probability as in (26). 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 in (26).

8 Multi-response models and chaining of the outcomes

Another interesting use-case is the multi-response setting, where the outcome Y has $K > 1$ columns. The data in these columns may be quantitative or integers. In this setting the rows of X are no longer grouped: the "grouping" here is defined by the columns of y (y_1 is group 1, y_2 is group 2 and so on). The multinomial target 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 for example in problems where each observation can fall in more than one class.

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

$$\sum_{k=1}^{K} \|y_k - X\beta_{\cdot,k}\|^2 + \sum_{j=1}^{p} \|\beta_{j,\cdot}\|_2, \tag{28}$$

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 $\beta_{\cdot,k}$.

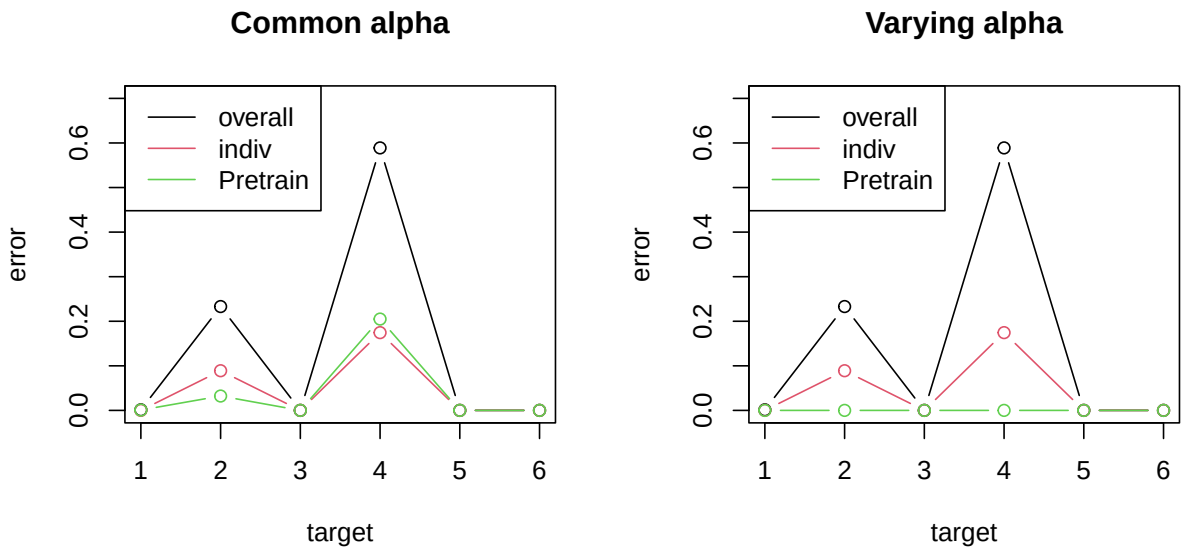

Figure 18: *Results for multi-response chemometrics example.*

Figure 18 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 measurements 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 18 shows the LOOCV squared error over the 6 outcomes, using both the best common α (left plot) and the allowing α to vary over the 6 outcomes. We see that pretrained lasso performs best in both settings.

Other multi-response settings include time-ordered target columns, and outcomes of different types (e.g. quantitative, survival, binary). In both cases we can apply pretrained lasso in a sequential (chained) fashion. We fit a model to the first outcome, compute the offset and penalty factor, and pass these to a model which fits to the second outcome, and so on.

We illustrate the second scenario in Figure 19. We generated three outcome variables— quantitative, censored survival and binary— all functions of a (sparse) linear predictor $x\beta$. The β and noise levels were chosen so that the Gaussian linear model (for the first outcome) had an SNR of about 2. The figure panels show the test set results (in green) for the survival outcome and the binary outcome, as a function of the pretraining hyperparameter α. For comparison, the test set C-index and AUC for the survival and binary outcomes, modelled separately, are shown in red. We see that the pretrained lasso is able to borrow strength from one target to the next, and as a result, yields higher accuracy.

This application of pretrained lasso to mixed outcomes requires a prior ordering of the outcomes. In real applications it might make sense to place the primary outcome measure in the first position, and the rest in decreasing order of importance.

9 Conditional average treatment effect (CATE) estimation

An important problem in causal inference is the estimation of conditional average treatment effects. In the most common setting, 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

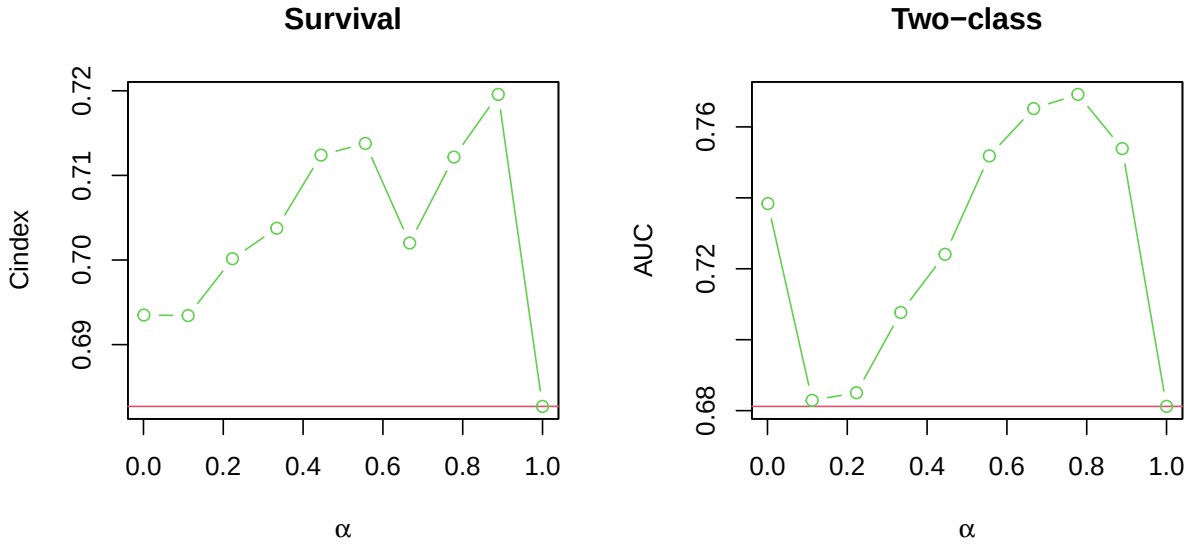

Figure 19: *Results for mixed outcomes example.*

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 assignment is unconfounded.

One popular approach is the "R-learner" of Nie & Wager (2021). It is based on the objective function

$$\hat{\tau} = \text{argmin}_\tau \frac{1}{\text{n}} \sum \Big[(\text{Y}_\text{i} - \text{m}^*(\text{X}_\text{i})) - (\text{W}_\text{i} - \text{e}^*(\text{X}_\text{i})) \cdot \tau(\text{X}_\text{i}) \Big]^2 \tag{29}$$

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 (29) 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

$$\begin{aligned} Y &= \beta_0 + X\beta + W \cdot \tau(X) + \epsilon \\ \tau(X) &= X\theta_0 + X\theta \end{aligned} \tag{30}$$

where θ_0 has the same support and signs as β. To fit this, we use the same 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 also seems reasonable: it says that the predictive features are likely to overlap with the features that modify the treatment effect.

Figure 20 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

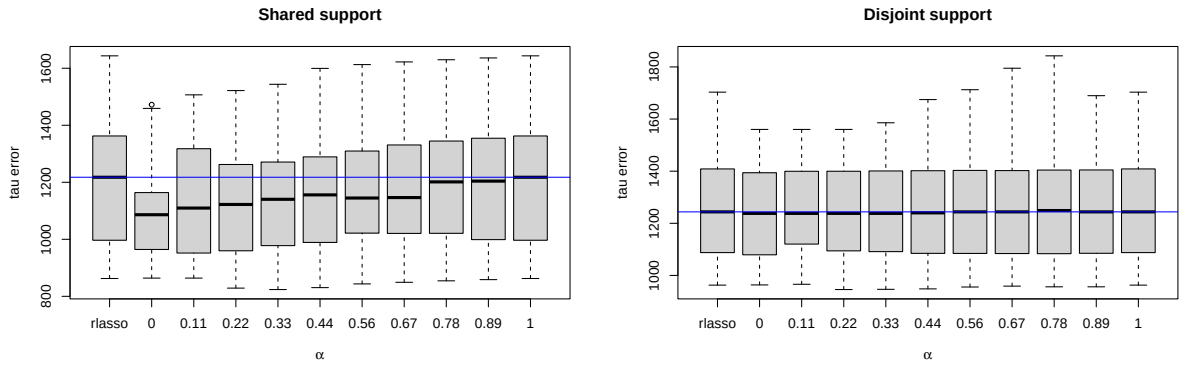

Figure 20: *Results for R-learner experiment. Horizontal blue line is drawn at the median error for the rlasso R-learner.*

support and positive 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 for all values of α the pretrained R-learner outperforms the R-learner, 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 30 with disjoint support, all 20 features are predictive of the outcome, and hence there is no support restriction resulting from the fit of the outcome model.

10 Beyond linear models: an application to gradient boosting

Here we explore the use of basis functions beyond the linear functions used throughout the paper. Suppose we run gradient boosting (Chen & Guestrin 2016) 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:

- run M iterations of xgboost to get M trees (basis functions) B $(n \times M)$
- 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 & Popescu 2008) and ESL (Hastie et al. 2009) page 622).

It is easy to implement this procedure using the `xgboost` library in R (Chen et al. 2023). Figure 21 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 test error results are shown in Figure 21. The first method— `xgboost`— is vanilla boosting applied to the raw features, while the other three methods use the 50 initial trees generated by `xgboost`. We see that lasso pretraining can help boosting as well.

11 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

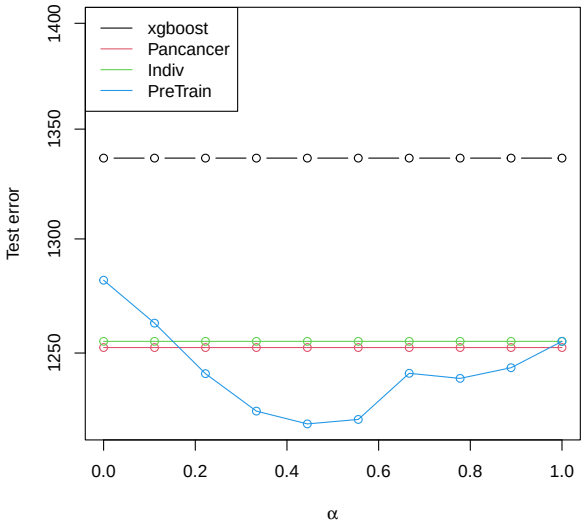

Figure 21: *Results for the pretrained lasso applied to boosted trees. The first method* `xgboost` *uses the raw features. The remaining three methods use the evaluated trees from* `xgboost` *as features.*

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 model was the same as that used in Figure 5, with the results in Figure 22. 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.

12 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 targets, multi-response models, conditional average treatment estimation and gradient boosting. There are likely to be other interesting applications of these ideas, including the transfer of knowledge from a model pretrained on a large corpus, and then fine-tuned on a smaller dataset for the task at hand.

We plan to release an open source R language package that implements these ideas.

Acknowledgements

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

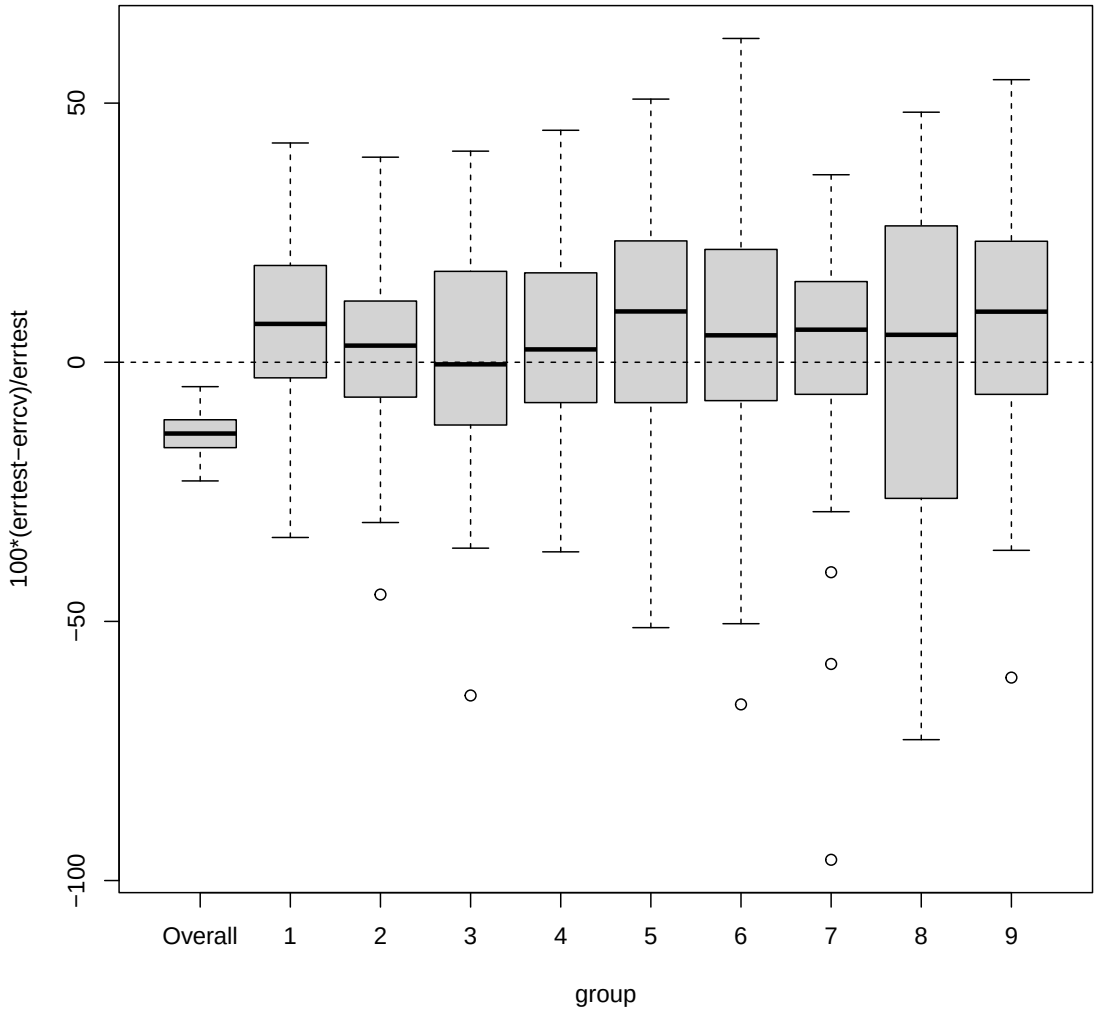

Figure 22: *Relative error of 5-fold CV error, as an estimate of test error (50 simulations). Left boxplot is for the overall model; other boxplots show results for the 9 groups in pretrained lasso. We see that CV overestimates test error in the overall model, but underestimates it each of the 9 groups at the end of the two-step pretraining. However the bias is relatively small in each case.*

Advancing Translational Sciences of the National Institutes of Health under Award Number UL1TR003142. M.A.R. is in part supported by National Human Genome Research Institute (NHGRI) under award R01HG010140, and by the National Institutes of Mental Health (NIMH) under award R01MH124244 both of the National Institutes of Health (NIH). R.T. was supported by the National Institutes of Health(5R01EB001988-16) and the National Science Foundation(19DMS1208164). M.P. was supported in part by the National Science Foundation (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 National Institute of General Medical Sciences grant R35 GM139517, Stanford University Discovery Innovation Award, and the Chan Zuckerberg Data Insights.

A Simulation study results

Here we include more complete results of the simulation study described in Section 6. As before, we compare pretraining with 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 5), grouped data with a binomial response (Table 6), and data with a multinomial response (Table 7).

SNR	PSE relative to Bayes error			Feature F1 score			Common feature F1 score	
	Overall	Pretrain	Indiv.	Overall	Pretrain	Indiv.	Pretrain	Indiv.
Common support with same magnitude, individual features Features: 10 common, 10 per group, 120 total								
13.6 ± 0.8	4.34 ± 0.28	$\mathbf{1.39 \pm 0.07}$	1.40 ± 0.07	$\mathbf{75 \pm 3}$	72 ± 2	70 ± 1	31 ± 6	$\mathbf{39 \pm 5}$
3.4 ± 0.2	1.89 ± 0.09	$\mathbf{1.39 \pm 0.07}$	1.41 ± 0.07	70 ± 5	$\mathbf{73 \pm 3}$	70 ± 1	$\mathbf{40 \pm 8}$	39 ± 5
0.4 ± 0.0	$\mathbf{1.14 \pm 0.03}$	1.19 ± 0.04	1.29 ± 0.05	23 ± 4	50 ± 6	$\mathbf{64 \pm 5}$	$\mathbf{86 \pm 10}$	73 ± 14
As above, but now $p > n$ Features: 10 common, 10 per group, 2040 total								
11.6 ± 0.7	2.79 ± 0.13	$\mathbf{1.91 \pm 0.19}$	2.68 ± 0.24	$\mathbf{37 \pm 5}$	33 ± 5	21 ± 2	72 ± 17	$\mathbf{91 \pm 6}$
2.9 ± 0.2	$\mathbf{1.51 \pm 0.06}$	1.52 ± 0.08	2.07 ± 0.13	$\mathbf{33 \pm 4}$	31 ± 5	18 ± 2	81 ± 15	$\mathbf{97 \pm 4}$
0.3 ± 0.0	$\mathbf{1.14 \pm 0.03}$	1.16 ± 0.04	1.32 ± 0.04	$\mathbf{29 \pm 4}$	23 ± 5	13 ± 4	$\mathbf{84 \pm 12}$	5 ± 9
Common support with same magnitude, no individual features Features: 10 common, 0 per group, 120 total								
10.0 ± 0.6	$\mathbf{1.05 \pm 0.02}$	1.07 ± 0.02	1.23 ± 0.05	50 ± 11	$\mathbf{88 \pm 14}$	21 ± 2	$\mathbf{92 \pm 7}$	76 ± 9
2.5 ± 0.2	$\mathbf{1.05 \pm 0.02}$	1.07 ± 0.03	1.23 ± 0.05	52 ± 11	$\mathbf{85 \pm 16}$	21 ± 2	$\mathbf{91 \pm 9}$	74 ± 9
0.3 ± 0	$\mathbf{1.05 \pm 0.02}$	1.08 ± 0.03	1.21 ± 0.04	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, 0 per group, 120 total								
4.4 ± 0.3	1.87 ± 0.09	$\mathbf{1.10 \pm 0.03}$	1.23 ± 0.05	51 ± 10	$\mathbf{62 \pm 20}$	22 ± 2	$\mathbf{93 \pm 8}$	79 ± 9
1.1 ± 0.1	1.26 ± 0.04	$\mathbf{1.09 \pm 0.03}$	1.20 ± 0.04	52 ± 10	$\mathbf{73 \pm 22}$	23 ± 3	$\mathbf{93 \pm 9}$	86 ± 10
0.1 ± 0.0	$\mathbf{1.06 \pm 0.02}$	1.09 ± 0.03	1.11 ± 0.03	$\mathbf{53 \pm 12}$	52 ± 18	38 ± 11	$\mathbf{64 \pm 21}$	16 ± 19
As above, but now with individual features Features: 10 common, 10 per group, 120 total								
10.8 ± 0.7	3.79 ± 0.21	$\mathbf{1.35 \pm 0.07}$	1.40 ± 0.07	64 ± 5	$\mathbf{73 \pm 3}$	70 ± 1	$\mathbf{49 \pm 9}$	40 ± 5
4.8 ± 0.4	2.27 ± 0.12	$\mathbf{1.32 \pm 0.05}$	1.39 ± 0.06	61 ± 5	$\mathbf{73 \pm 3}$	70 ± 2	$\mathbf{64 \pm 13}$	47 ± 7
1.2 ± 0.1	1.35 ± 0.05	$\mathbf{1.18 \pm 0.04}$	1.29 ± 0.05	55 ± 5	51 ± 10	$\mathbf{65 \pm 3}$	$\mathbf{86 \pm 11}$	76 ± 12
Individual features only Features: 0 common, 10 per group, 120 total								
10.1 ± 0.5	9.84 ± 0.56	1.21 ± 0.05	$\mathbf{1.20 \pm 0.04}$	65 ± 3	$\mathbf{67 \pm 3}$	$\mathbf{67 \pm 4}$	—	—
2.5 ± 0.2	3.3 ± 0.19	$\mathbf{1.24 \pm 0.05}$	$\mathbf{1.24 \pm 0.05}$	63 ± 6	$\mathbf{68 \pm 3}$	$\mathbf{68 \pm 2}$	—	—
0.6 ± 0.0	1.59 ± 0.06	$\mathbf{1.25 \pm 0.05}$	$\mathbf{1.25 \pm 0.05}$	52 ± 5	$\mathbf{69 \pm 2}$	68 ± 3	—	—

Table 5: *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				Feature F1 score			Common feature F1 score	
Bayes	Overall	Pretrain	Indiv.	Overall	Pretrain	Indiv.	Pretrain	Indiv.
Common support with same magnitude, individual features Features: 5 common, 5 per group, 40 total								
0.82 ± 0.02	0.71 ± 0.03	$\mathbf{0.72 \pm 0.04}$	0.70 ± 0.04	62 ± 6	65 ± 8	$\mathbf{66 \pm 6}$	$\mathbf{44 \pm 15}$	39 ± 13
0.70 ± 0.03	$\mathbf{0.60 \pm 0.04}$	0.59 ± 0.04	0.58 ± 0.03	57 ± 8	60 ± 11	$\mathbf{63 \pm 8}$	$\mathbf{35 \pm 12}$	28 ± 12
0.61 ± 0.03	$\mathbf{0.54 \pm 0.03}$	0.53 ± 0.03	0.53 ± 0.03	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 per group, 320 total								
0.82 ± 0.02	$\mathbf{0.68 \pm 0.04}$	0.67 ± 0.04	0.63 ± 0.04	$\mathbf{42 \pm 9}$	24 ± 7	23 ± 6	$\mathbf{40 \pm 16}$	32 ± 13
0.70 ± 0.03	$\mathbf{0.56 \pm 0.04}$	0.55 ± 0.04	0.53 ± 0.04	$\mathbf{29 \pm 11}$	16 ± 6	15 ± 5	$\mathbf{24 \pm 14}$	14 ± 13
0.61 ± 0.03	$\mathbf{0.51 \pm 0.03}$	$\mathbf{0.51 \pm 0.03}$	$\mathbf{0.51 \pm 0.04}$	$\mathbf{21 \pm 9}$	10 ± 6	12 ± 4	$\mathbf{8 \pm 8}$	3 ± 6
Common support with same magnitude, no individual features Features: 5 common, 0 per group, 40 total								
0.77 ± 0.02	$\mathbf{0.73 \pm 0.03}$	0.70 ± 0.04	0.66 ± 0.04	$\mathbf{56 \pm 7}$	47 ± 16	$\mathbf{56 \pm 14}$	$\mathbf{63 \pm 16}$	50 ± 17
0.65 ± 0.03	$\mathbf{0.60 \pm 0.04}$	0.57 ± 0.04	0.55 ± 0.04	54 ± 9	54 ± 16	$\mathbf{62 \pm 11}$	$\mathbf{39 \pm 15}$	30 ± 13
0.58 ± 0.04	$\mathbf{0.53 \pm 0.04}$	0.51 ± 0.04	0.51 ± 0.03	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								
0.71 ± 0.03	0.61 ± 0.04	$\mathbf{0.62 \pm 0.04}$	0.61 ± 0.04	53 ± 9	$\mathbf{56 \pm 14}$	$\mathbf{56 \pm 15}$	$\mathbf{44 \pm 16}$	36 ± 17
0.62 ± 0.03	$\mathbf{0.54 \pm 0.04}$	$\mathbf{0.54 \pm 0.04}$	0.53 ± 0.04	56 ± 9	60 ± 15	$\mathbf{62 \pm 11}$	$\mathbf{29 \pm 11}$	27 ± 10
0.56 ± 0.04	0.51 ± 0.03	$\mathbf{0.52 \pm 0.03}$	0.51 ± 0.03	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								
0.76 ± 0.03	0.61 ± 0.04	$\mathbf{0.65 \pm 0.04}$	0.64 ± 0.04	58 ± 9	63 ± 9	$\mathbf{64 \pm 8}$	$\mathbf{37 \pm 14}$	30 ± 15
0.70 ± 0.03	0.56 ± 0.04	$\mathbf{0.58 \pm 0.04}$	$\mathbf{0.58 \pm 0.04}$	56 ± 10	61 ± 8	$\mathbf{63 \pm 7}$	$\mathbf{28 \pm 13}$	26 ± 13
0.64 ± 0.03	$\mathbf{0.54 \pm 0.03}$	$\mathbf{0.54 \pm 0.04}$	$\mathbf{0.54 \pm 0.03}$	55 ± 9	58 ± 14	$\mathbf{61 \pm 12}$	$\mathbf{24 \pm 13}$	21 ± 12
Individual features only								
0.72 ± 0.03	0.56 ± 0.03	$\mathbf{0.62 \pm 0.04}$	$\mathbf{0.62 \pm 0.04}$	$\mathbf{57 \pm 9}$	59 ± 10	55 ± 14	—	—
0.66 ± 0.03	0.53 ± 0.04	$\mathbf{0.56 \pm 0.03}$	$\mathbf{0.56 \pm 0.03}$	58 ± 9	$\mathbf{60 \pm 12}$	59 ± 13	—	—
0.60 ± 0.03	0.52 ± 0.03	$\mathbf{0.53 \pm 0.03}$	$\mathbf{0.53 \pm 0.03}$	57 ± 8	$\mathbf{63 \pm 10}$	59 ± 14	—	—

Table 6: *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 misclass. rate			Feature F1 score			Common feature F1 score	
Bayes rule	Overall	Pretrain	Indiv.	Overall	Pretrain	Indiv.	Pretrain	Indiv.
Common support, individual features Features: 3 common, 10 per group, 159 total								
0.49 ± 0.01	0.59 ± 0.02	$\mathbf{0.58 \pm 0.02}$	0.60 ± 0.02	55 ± 8	$\mathbf{57 \pm 5}$	55 ± 5	26 ± 19	$\mathbf{28 \pm 31}$
0.22 ± 0.01	0.33 ± 0.02	$\mathbf{0.32 \pm 0.02}$	$\mathbf{0.32 \pm 0.03}$	$\mathbf{71 \pm 6}$	65 ± 6	65 ± 6	20 ± 15	$\mathbf{55 \pm 20}$
As above, but now $p > n$ Features: 3 common, 10 per group, 640 total								
0.56 ± 0.01	0.62 ± 0.02	$\mathbf{0.61 \pm 0.02}$	0.62 ± 0.03	34 ± 7	$\mathbf{36 \pm 6}$	$\mathbf{36 \pm 5}$	$\mathbf{26 \pm 20}$	21 ± 30
0.28 ± 0.01	0.35 ± 0.03	$\mathbf{0.34 \pm 0.02}$	0.36 ± 0.04	$\mathbf{56 \pm 11}$	48 ± 8	49 ± 7	24 ± 22	$\mathbf{73 \pm 21}$
Common support, no individual features Features: 3 common, 0 per group, 159 total								
0.70 ± 0.01	0.71 ± 0.03	$\mathbf{0.70 \pm 0.02}$	0.75 ± 0.04	$\mathbf{32 \pm 13}$	28 ± 10	21 ± 11	$\mathbf{36 \pm 22}$	7 ± 21
0.58 ± 0.01	$\mathbf{0.57 \pm 0.02}$	0.58 ± 0.01	0.60 ± 0.02	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								
0.54 ± 0.01	0.64 ± 0.02	$\mathbf{0.63 \pm 0.02}$	0.64 ± 0.03	$\mathbf{54 \pm 7}$	$\mathbf{54 \pm 4}$	50 ± 7	—	—
0.27 ± 0.01	0.38 ± 0.02	$\mathbf{0.36 \pm 0.02}$	0.37 ± 0.03	$\mathbf{69 \pm 6}$	63 ± 6	64 ± 6	—	—

Table 7: *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.*

B Mathematical Proofs

B.1 Proof of Lemma 1

We analyze the conditions for optimality of the pretraining estimator given in (12). 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

$$X_S^T(X_S\beta_S - y) + \lambda \mathbf{sign}(\beta_S) = 0 \tag{31}$$

$$\|X_{S^c}^T(X_S\beta_S - y)\|_\infty < \lambda, \tag{32}$$

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

$$\beta_S = (X_S^T X_S)^{-1}(X_S^T y - \lambda \mathbf{sign}(\beta_S)). \tag{33}$$

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

$$\beta_S = (X_S^T X_S)^{-1}(X_S^T \sum_{k=1}^K D_k X w_k^*) + X_S^\dagger \varepsilon - \lambda (X_S^T X_S)^{-1}\mathbf{sign}(\beta_S). \tag{34}$$

Plugging in the above expression into the condition (32), and dividing both sides by λ, we obtain

$$\|X_{S^c}^T(\lambda^{-1}P_S^\perp \sum_{k=1}^K D_k X w_k^* + \lambda^{-1}P_S^\perp \varepsilon + X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty < 1. \tag{35}$$

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

$$\lambda^{-1}\|X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X w_k^*\|_\infty + \lambda^{-1}\|X_{S^c}^T P_S^\perp \varepsilon\|_\infty + \|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty < 1. \tag{36}$$

Therefore by imposing the conditions

$$\|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty < \frac{1}{2} \tag{37}$$

$$\|X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X w_k^*\|_\infty \leq \frac{\lambda}{4} \tag{38}$$

$$\|X_{S^c}^T P_S^\perp \varepsilon\|_\infty \leq \frac{\lambda}{4}, \tag{39}$$

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

$$\|X_{S^c}^T X_S^\dagger \mathbf{sign}(\beta_S)\|_\infty = \max_{j\in S^c} |x_j^T X_S^\dagger \mathbf{sign}(\beta_S)|. \tag{40}$$

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.,

$$\|X_S^T X_S - I\|_2 \leq \delta, \tag{41}$$

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

$$\mathbb{P}\left[\max_{j \in S^c} |x_j^T X_S^\dagger \mathbf{sign}(\beta_S)| \leq \delta\right] \leq (p-s)\mathbb{P}\left[|x_1^T X_S^\dagger \mathbf{sign}(\beta_S)| \leq \delta\right] \tag{42}$$

$$\leq (p-s)e^{-\delta^2 C' n/s} \tag{43}$$

$$= e^{-C'\delta^2 n/s + \log(p-s)}, \tag{44}$$

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

$$\|P_S^\perp \sum_{k=1}^K D_k X_S (w_k^*)_S\|_2 \leq \|\sum_{k=1}^K D_k X_S (w_k^*)_S\|_2 \tag{45}$$

$$= \left(\sum_{k=1}^K \|D_k X_S (w_k^*)_S\|_2^2\right)^{1/2}. \tag{46}$$

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

$$\|P_S^{\perp} \sum_{k=1}^{K} D_k X_S (w_k^*)_S\|_2 \leq \left(\sum_{k=1}^{K} \|D_k X_S (w_k^*)_S\|_2^2 \|(w_k^*)_S\|_2^2 \right)^{1/2} \tag{47}$$

$$\lesssim \left(\sum_{k=1}^{K} (\frac{1}{K} + \frac{s}{n}) s\gamma^2 \right)^{1/2} \tag{48}$$

$$= \left((1 + \frac{Ks}{n}) s\gamma^2 \right)^{1/2} \tag{49}$$

$$\leq \sqrt{s}\gamma + \frac{\sqrt{K}}{\sqrt{n}} s\gamma \tag{50}$$

$$\leq 2\sqrt{s}\gamma, \tag{51}$$

where we used $n/K \geq s$, i.e., each subgroup has at least s samples, in the final inequality. Repeating the same sub-Gaussianity argument and 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 (34) 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 (33), 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 (33) to obtain

$$\|\beta_S - \frac{1}{K}\sum_{k=1}^{K} \beta_k^*\|_{\infty} \leq \lambda \|(X_S^T X_S)^{-1} \mathbf{sign}(\beta_S^*)\|_2 + \delta \sum_{k=1}^{K} \|\beta_k^*\|_2 + \|X_S^{\dagger}\varepsilon\|, \tag{52}$$

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

$$\|X_{S^c}^T P_S^\perp \sum_{k=1}^K D_k X w_k^*\|_\infty = \max_{j \in S^c} |x_j^T \sum_{k=1}^K D_k X w_k^*| \tag{53}$$

$$= |x_j^T \sum_{r \neq j} \sum_{k=1}^K D_k X (w_k^*)_r + x_j^T D_k X (w_k^*)_j| \tag{54}$$

$$\leq |x_j^T \sum_{r \neq j} \sum_{k=1}^K D_k X (w_k^*)_r| + |x_j^T x_j (w_k^*)_j| \tag{55}$$

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

$$\sum_{k=1}^K \|X_k \hat{\beta}_0 - y_k\|_2^2 \leq \sum_{k=1}^K \|X_k \beta_0^* - y_k\|_2^2, \tag{56}$$

which follows from the optimality of $\hat{\beta}_0$ and the feasibility of β_0^* in the pretraining Lasso objective due to our assumption that $\|\beta_0^*\|_1 \leq R$. Plugging in the model for y, we obtain

$$\sum_{k=1}^K \|X_k(\hat{\beta}_0 - \beta_0^*) - X_k \beta_k^* - \varepsilon_k\|_2^2 \leq \sum_{k=1}^K \|X_k \beta_0^* + \varepsilon_k\|_2^2. \tag{57}$$

Expanding the square and cancelling common terms we get

$$\sum_k \|X_k \Delta_0\|_2^2 \leq 2 \sum_k (X_k \beta_k^* + \varepsilon_k)^T X_k \Delta_0, \tag{58}$$

where we defined $\Delta_0 := \hat{\beta}_0 - \beta_0^*$. Using the fact that $\|\hat{\beta}_0\|_1 \leq R$, $\|\beta_0^*\| \leq R$, we obtain $\|\Delta_0\|_1 \leq 2R$ and apply Cauchy–Schwarz inequality to reach

$$\sum_{k=1}^K \|X_k \Delta_0\|_2^2 \leq 2R \|\sum_k X_k^T \varepsilon_k\|_\infty + 2R \sum_k \|X_k^T X_k \beta_k^*\|_\infty \tag{59}$$

$$\leq 2R \|\sum_k X_k^T \varepsilon_k\|_\infty + 2R \sum_k \|X_k^T X_k\|_\infty \|\beta_k^*\|_1 \tag{60}$$

$$\leq 2R \|\sum_k X_k^T \varepsilon_k\|_\infty + 2C'' R \max_{k \in [K]} \max_{i,j \in [p]} |(X_k^T X_k)_{ij}| \tag{61}$$

$$\leq 2R \|\sum_k X_k^T \varepsilon_k\|_\infty + \frac{n 2 R C' C''}{K} \tag{62}$$

We apply standard concentration results for the maximum of independent sub-Gaussian variables Vershynin (2018) to control the term $\| \sum_k X_k^T \varepsilon_k \|_\infty$ and establish that

$$\frac{1}{n} \sum_k \|X_k \Delta_0\|_2^2 \leq \frac{4RC\sqrt{n}\sigma\sqrt{\log(p)}}{n} + \frac{2RC'C''}{K}, \tag{63}$$

with probability at least $1 - C_3/n$. 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

$$\|(1-\alpha)X_k\hat{\beta}_0 + X_k\hat{\beta}_k - y_k\|_2^2 \leq \|(1-\alpha)X_k\hat{\beta}_0 + X_k\beta_k^* - y_k\|_2^2. \tag{64}$$

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

$$\|X_k((1-\alpha)\Delta_0 + \Delta_k) - \varepsilon_k\|_2^2 \leq \|(1-\alpha)X_k\Delta_0 - \varepsilon_k\|_2^2. \tag{65}$$

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

$$\begin{aligned} \|X_k((1-\alpha)\Delta_0 + \Delta_k)\|_2^2 &\leq \|(1-\alpha)X_k\Delta_0\|_2^2 - 2(1-\alpha)\Delta_0^T X_k^T\varepsilon_k + 2((1-\alpha)\Delta_0 + \Delta_k)^T X_k^T\varepsilon_k & (66)\\ &= \|(1-\alpha)X_k\Delta_0\|_2^2 + 2\Delta_k^T X_k^T\varepsilon_k. & (67)\end{aligned}$$

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

$$\sum_{k=1}^K \|X_k((1-\alpha)\Delta_0 + \Delta_k)\|_2^2 = \sum_{k=1}^K \|(1-\alpha)X_k\Delta_0\|_2^2 + 2\Delta_k^T X_k^T\varepsilon_k. \tag{68}$$

Since $\sum_{k=1}^K \|\Delta_k\|_1 \leq \sum_{k=1}^K \|\hat{\beta}_k\|_1 + \|\beta_k^*\|_1 \leq 2\sum_{k=1}^K R_k \leq 2C''$, we use the bound $2\sum_k \Delta_k^T X_k^T\varepsilon_k \leq 2\sum_k \|\Delta_k\|_1 \|X_k^T\varepsilon_k\|_\infty \leq 4C'' \max_{k\in[K]} \|X_k^T\varepsilon_k\|_\infty$, where we applied the Cauchy Schwarz inequality twice. Following the concentration bound for the maximum of sub-Gaussian variables as before, $\|X_k^T\varepsilon_k\|_\infty$ is bounded by $2\sigma\sqrt{n/K}\sqrt{\log(pK)}$ with high probability for all $k \in [K]$. We set $\alpha = 1/2$ and combine the above bound with the error on the pretraining stage in (63), and obtain

$$\begin{aligned} \frac{1}{n}\sum_{k=1}^K \|X_k(\Delta_0/2 + \Delta_k)\|_2^2 &\leq \frac{1}{4}\left(\frac{4RC\sigma\sqrt{\log(p)}}{\sqrt{n}} + \frac{2RC'C''}{K}\right) + \frac{8C''\sigma\frac{1}{\sqrt{K}}\sqrt{\log(pK)}}{\sqrt{n}} & (69)\\ &= \frac{\sigma(RC\sqrt{\log p} + 8C''\sqrt{\log(pK)/K})}{\sqrt{n}} + \frac{RC'C''}{2K}, & (70)\end{aligned}$$

with probability at least $1 - C_3/n$, which completes the proof.

References

Chaung, K., Baharav, T. Z., Henderson, G., Zheludev, I. N., Wang, P. L. & Salzman, J. (2023), 'Splash: A statistical, reference-free genomic algorithm unifies biological discovery', *Cell* **186**(25), 5440–5456.e26. **URL:** *https://www.sciencedirect.com/science/article/pii/S0092867423011790*

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining', pp. 785–794.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y. & Yuan, J. (2023), *xgboost: Extreme Gradient Boosting*. R package version 1.7.6.1.
URL: *https://CRAN.R-project.org/package=xgboost*

Consortium, T. T. S., Jones, R. C., Karkanias, J., Krasnow, M. A., Pisco, A. O., Quake, S. R., Salzman, J., Yosef, N., Bulthaup, B., Brown, P. et al. (2022), 'The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans', *Science* **376**(6594), eabl4896.

Friedman, J. H. & Popescu, B. E. (2008), 'Predictive learning via rule ensembles'.

Friedman, J., Tibshirani, R. & Hastie, T. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22.

Goldman, M. J., Craft, B., Hastie, M., Repečka, K., McDade, F., Kamath, A., Banerjee, A., Luo, Y., Rogers, D., Brooks, A. N. et al. (2020), 'Visualizing and interpreting cancer genomics data via the xena platform', *Nature biotechnology* **38**(6), 675–678.

Gross, S. M. & Tibshirani, R. (2016), 'Data shared lasso: A novel tool to discover uplift', *Computational statistics & data analysis* **101**, 226–235.

Hastie, T., Tibshirani, R., Friedman, J. H. & Friedman, J. H. (2009), *The elements of statistical learning: data mining, inference, and prediction*, Vol. 2, Springer.

Hastie, T., Tibshirani, R. & Wainwright, M. (2015), *Statistical learning with sparsity: the lasso and generalizations*, CRC press.

Kokot, M., Dehghannasiri, R., Baharav, T., Salzman, J. & Deorowicz, S. (2023), 'Splash2 provides ultra-efficient, scalable, and unsupervised discovery on raw sequencing reads', *BioRxiv* .

Margulis, K., Chiou, A. S., Aasi, S. Z., Tibshirani, R. J., Tang, J. Y. & Zare, R. N. (2018), '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.

McGough, S. F., Lyalina, S., Incerti, D., Huang, Y., Tyanova, S., Mace, K., Harbron, C., Copping, R., Narasimhan, B. & Tibshirani, R. (2023), 'Prognostic pan-cancer and single-cancer models: A large-scale analysis using a real-world clinico-genomic database', *medRxiv* .
URL: *https://www.medrxiv.org/content/early/2023/12/19/2023.12.18.23300166*

Nie, X. & Wager, S. (2021), 'Quasi-oracle estimation of heterogeneous treatment effects', *Biometrika* **108**(2), 299–319.

Olivieri, J. E., Dehghannasiri, R., Wang, P. L., Jang, S., De Morree, A., Tan, S. Y., Ming, J., Wu, A. R., Quake, S. R., Krasnow, M. A. et al. (2021), 'Rna splicing programs define tissue compartments and cell types at single-cell resolution', *Elife* **10**, e70692.

Schilder, R. J., Kimball, S. R. & Jefferson, L. S. (2012), '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.

Skagerberg, B., MacGregor, J. F. & Kiparissides, C. (1992), 'Multivariate data analysis applied to low-density polyethylene reactors', *Chemometrics and intelligent laboratory systems* **14**(1-3), 341–356.

Tuck, J. & Boyd, S. (2021), 'Fitting laplacian regularized stratified gaussian models', *Optimization and Engineering* pp. 1–21.

Van De Geer, S. A. & Bühlmann, P. (2009), 'On the conditions used to prove oracle results for the lasso'.

Vershynin, R. (2018), *High-dimensional probability: An introduction with applications in data science*, Vol. 47, Cambridge university press.

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

Yu, G., Bien, J. & Tibshirani, R. (2019), 'Reluctant interaction modeling', *arXiv preprint arXiv:1907.08414* .

Zhao, P. & Yu, B. (2006), 'On model selection consistency of lasso', *The Journal of Machine Learning Research* **7**, 2541–2563.