Final Lab - fMRI

Aniket Kesari
25948127
UC Berkeley Law

November 18, 2017

```
## [1] 50000
```

Introduction

In this lab, the objective is to develop an algorithm that uses functional Magnetic Resonance Imaging (fMRI) data to reconstruct still images. The data provided provides several thousand features, as well as multiple target outcomes. Each of the target outcomes represents the intensity of the response in a "voxel" (analagous to a 3D pixel) on the brain, and the features are the fMRI signals. This lab utilizes supervised learning approaches to develop accurate predictive regression models. The report proceeds as follows: Part II discusses the data cleaning procedure, Part III does some exploratory data analysis, Part IV discusses the preparation of the data partition, Part V discusses the various regression methods used, Part VI explores model selection techniques, Part VII presents validation set performance, Part VIII looks at regression diagnostics, Part IX discusses the scientific findings, Part X summarizes out-of-sample prediction issues, Part XI concludes.

Data Cleaning/Processing

The data came unlabeled, and in several pieces, which demanded a few preliminary cleaning steps. I take the following steps:

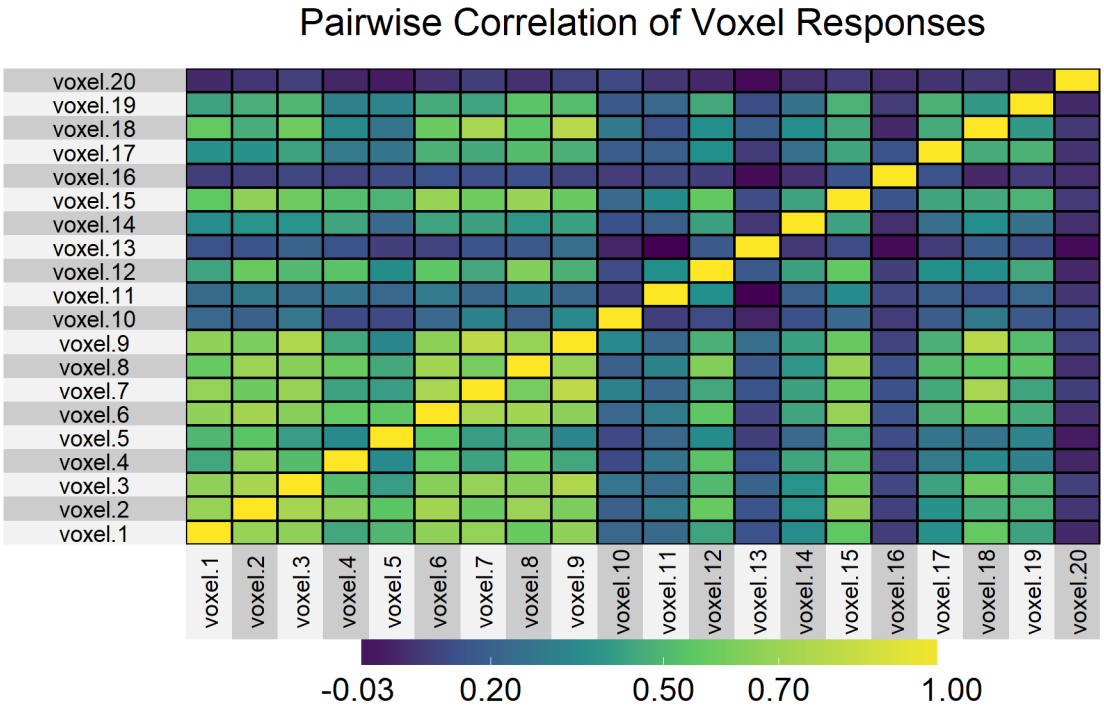
1. Label all of the columns in the "feature" dataset with the prefix of "feature" followed by the column number.
2. Add an ID column to the "voxel location" dataset.
3. Label all of the columns in the "response" dataset with voxel numbers, corresponding to the IDs assigned in the previous step.
4. Join the "feature" and "response" datasets into a master dataset.

Exploratory Data Analysis

Below, I visualize the pairwise Pearson correlations between the voxel intensities. A few interesting patterns emerge:

1. Voxels 1-9 share high correlations with one another
2. Voxels 17-19 are similar to voxel 1-9
3. Voxels 10, 11, 13, 16, and 20 share very little similarity to any other voxel. Voxels 16 and 20 seem especially uncorrelated.

Because the voxels are a spatial concept, it is unsurprising that some of the responses might be highly correlated with one another, as multiple voxels may correspond to a particular region of the brain.



To investigate this further, I plot a 3D scatterplot (Figure 1) to visualize the location of the voxels relative to one another. The voxels are shaded by their "Z" coordinates to highlight which voxels are in the same plane as each other. Looking at this plot, the correlations between voxels 1-9 makes sense as they are spatially proximate (all on the left side of the brain, and touching at least one other voxel in this group). Meanwhile, voxel 20 is on its own plane, and voxel 16 is quite far from any other voxel, which explains why their values were especially uncorrelated.

Partititon the Data

Moving onto model fitting, I partition the master dataset into training, validation, and test sets. Each set uses 60%, 20%, and 20% of the data respectively. These numbers were chosen to strike a balance between providing the algorithm with enough data to learn on, while also saving enough data for model selection and testing purposes.

The basic steps for model fitting are (according to Hastie in Elements of Statistical Learning):

1. Train the models on the training data
2. Predict on the validation set
3. Estimate the Prediction Error and Perform Model Selection
4. Use the best model to predict on the test set to measure Generalization Error

Regression Methods

There are a few difficulties with this particular prediction problem. The usual Ordinary Least Squares (OLS) takes the form:

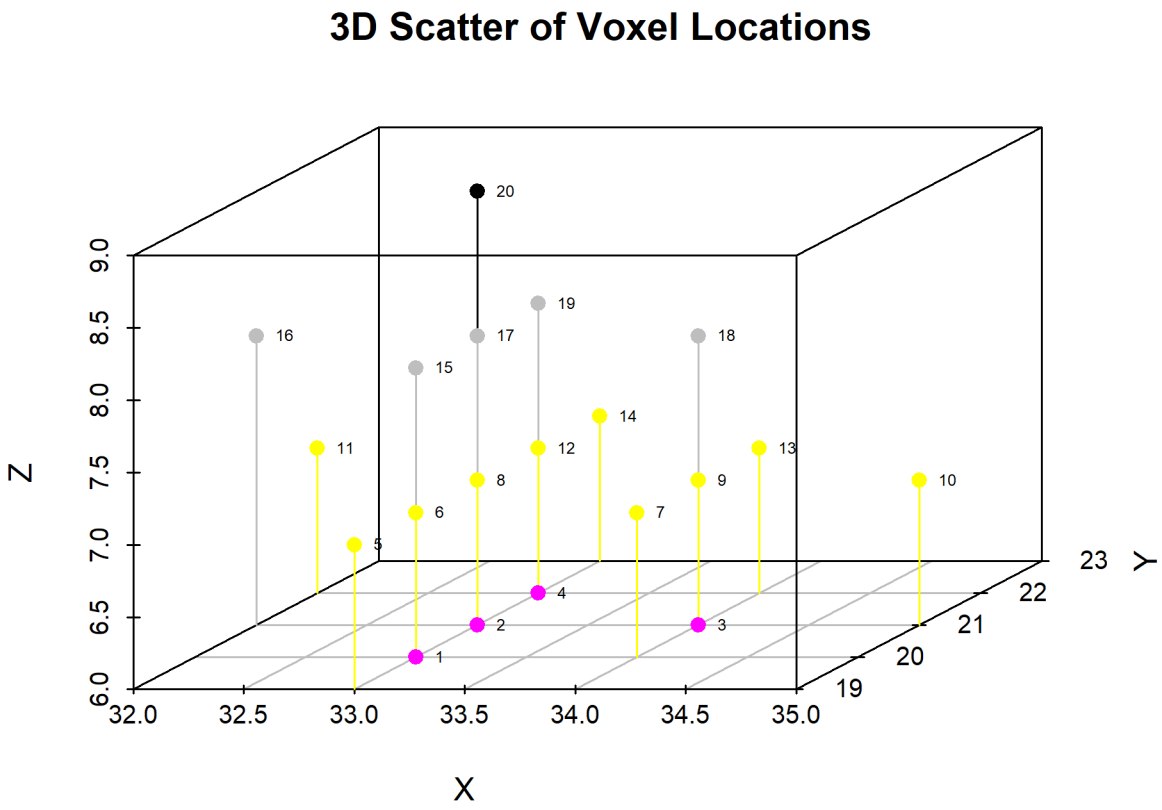


Figure 1: Spatial Position of Voxels

$Y = X\beta + \epsilon$

Where Y is a vector of responses, and X is a matrix of predictors. However, the standard OLS regression can fall apart if certain assumptions are violated. In this case, the predictor data likely suffers from multicollinearity, which can create a range of problems including biased regression estimates, large standard errors, and poor predictive performance. Furthermore, this dataset has only 1750 observations, but over 10,000 features. In general, one should avoid specifying regression models with more features than observations because there is not a lot of information to make accurate predictions. Thus, the main task is to develop algorithms that avoids these problems, but can still achieve good predictive performance.

From a computational perspective, the problem is further complicated because of the high-dimensionality and multiple responses. Models can take a long time to train on high-dimensional datasets, particular when relying on iterative procedures. Furthermore, in this case, the task requires predicting 20 separate models (one for each voxel). All of this is compounded by the additional computation required for bootstrapping and/or cross-validation procedures. Parallel processing and cluster computing aid in this endeavor.

Random Forest

The first method I employ is the random forest. A Random Forest is a non-parametric ensemble method that constructs multiple Classification and Regression Trees (CARTs), and then returns the mean prediction. The main advantage of the random forest over a CART is that it mitigates the tendency of CART to overfit its training data. Moreover, it is well suited to datasets with a large number of predictors because it randomly selects predictors as it grows trees and chooses optimal splits based on these random selections. In particular, I implement the "ranger" package. The primary advantage of "ranger" over other implementations is that its base is written in C, and therefore performs computations quickly.

Least Absolute Shrinkage and Selection Operator (LASSO)

Next, I employ a Least Absolute Shrinkage and Selection Operator (LASSO) method. The LASSO is advantageous because it performs both variable selection and regularization. This means that it culls extaneous predictors (thus simplifying the model), and prevents overfitting the statistical noise, which aids in out-of-sample predictions.

Model Selection for LASSO

After running the above models, to select the best model, I employ a number of selection techniques. These techniques basically evaluate model parameters. The central goals are are to select the simplest model possible that also generalizes well to a new dataset.

Cross-Validation

The basic logic underlying cross-validation techniques is that the data should be split in some fashion, and the model should be trained on a large fraction of the data, then tested on another portion of the data that was left out of the training process.

In this case, I employed a k-fold cross-validation procedure within the LASSO call. In this procedure, the data is split into k-folds, and one fold is left out as the test set, with the other k-1 folds used for training. This procedure is repeated until all folds have been used exactly once for testing, and then the estimate is derived from averaging the results from each of the k results.

The advantage to this procedure is that it directly measures prediction error, and corrects for overfitting to the training data by cycling through validation sets and taking their average. The main disadvantage to this

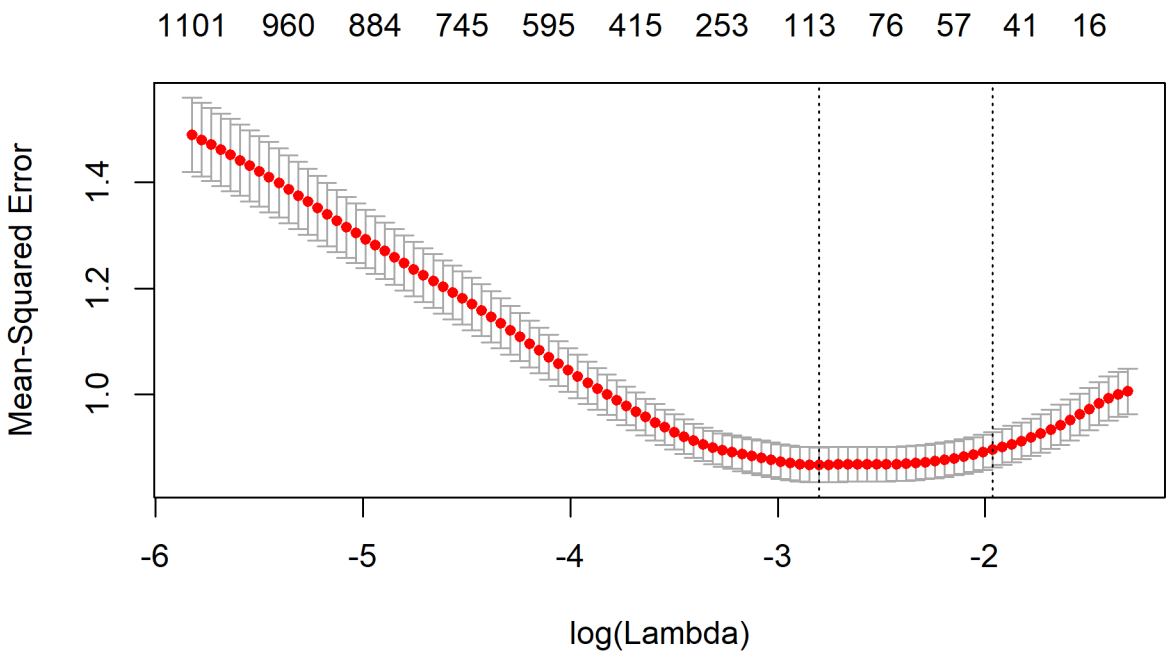


Figure 2: Plot of CV LASSO

approach is that it is computationally expensive (especially as one adds more k's), forcing a tradeoff between predictive accuracy and computation time. Moreover, k-fold cross validation will yield biased estimates relative to exhaustive cross validation procedures (detailed below).

To illustrate the advantage of a cv-based approach, consider the plot below (Figure 2). The points each represent the intersection of the Mean Square Error of an estimate (with confidence intervals), and the corresponding lambda parameter. In a LASSO procedure, lambda is basically a parameter that tells the function how much to penalize the function. In this case, the tuning parameter is minimized at around the log of -2.7.

ES-CV

Estimation Stability Cross Validation (ES-CV) was first proposed by Lim and Yu (2013). The Estimation Stability (ES) portion is basically a test statistic that penalizes the estimated variance by the squared mean size of the estimated solution across a series of tuning parameters, τ. Because a test statistic is not guaranteed to converge to a single local minimum, the ES component is combined with cross-validation to select the best metric.

Akaike Information Criterion

The Akaike Information Criterion (AIC) was derived from information theory. In the context of model selection, it is basically a measure of information loss. The idea is that one should minimize the AIC between several candidate models.

The formula is:

$AIC = 2k - 2ln(\hat{L})$

Where k is the number of parameters, and $\hat{L}$ is the maximum likelihood of that model.

The major benefit from an information criterion approach is that it is not nearly as computationally intensive as cross-validation methods. The major disadvantage is that AIC and AICc (discussed below) rely on the assumption that the model being assessed is univariate, linear in its terms, and has normally distributed residuals. If these assumptions fail, the formulau used to derive the information entropy will not be correct.

AICc

AICc is substantially similar to AIC. It solves the following formula:

$AICc = AIC + \frac{2k(k+1)}{n-k-1}$

Basically, it adds a small sample correction the usual AIC equation. The main advantage of adding this penalty is that AIC may select too many parameters when n is not many times larger than k^2, thus resulting in an overfit model. AICc approaches AIC as n grows larger, which makes it most useful when dealing with small sample sizes and lots of features.

Bayes Information Criterion

Formally, the Bayes Information Criterion (BIC) is defined as:

$BIC = ln(n)k - 2n(\hat{L})$

BIC is similar to AIC in that it is also rooted in information theory. THe main difference is that BIC will tend to penalize extra parameters, and assumes that the "true" model exists and will try to find it. AIC may be a better choice when the goal is simply predictive performance rather than fitting the true causal model.

Comparative Discussion

Overall, I would prefer to use ESCV as much as possible, with BIC being a good choice if computing the ESCV is prohibitively expensive. The table below gives a sample each metric's scores for the number of variables in the model. As expected, CV and ESCV are quite close to one another, as are AIC and AICc. I would be wary of using either AIC measure in this case because of the massive number of features in this dataset, and AIC does not penalize the number of features as much as BIC.

The lowest ESCV measure was a lambda of .002, which corresponded to selecting 1087 variables. This is considerably fewer than the nearly 11,000 that were originally present in the dataset.

Performance on Validation Set

Next, I evaluate the performance of the models on the validation set that was previously set aside. Using the Gallant Lab standard, I calculated this as the Pearson correlation between the predicted value and the actual value. Below, I visualize

Table 1: Preview of Model Selection Metrics

Df	Dev	CVLambda	ESCVLambda	AIC	AICc	BIC
0	0.0000000	0.2845972	0.2845972	0.000000	0.000000	0.000000
2	0.0074680	0.2716618	0.2716618	-4.152736	-4.141297	5.764161
5	0.0185409	0.2593144	0.2593144	-10.240830	-10.183469	14.551412
10	0.0354348	0.2475281	0.2475281	-18.683714	-18.472379	30.900770
15	0.0541366	0.2362776	0.2362776	-29.100146	-28.636826	45.276580
17	0.0718590	0.2255384	0.2255384	-44.447457	-43.855581	39.846165
21	0.0889148	0.2152873	0.2152873	-55.067005	-54.169917	49.060412
24	0.1049911	0.2055022	0.2055022	-66.617247	-65.448795	52.385514
27	0.1203941	0.1961618	0.1961618	-77.432526	-75.955964	56.445580
28	0.1347239	0.1872459	0.1872459	-91.076193	-89.488705	47.760362
29	0.1482037	0.1787353	0.1787353	-103.791883	-102.089339	40.003120
30	0.1606659	0.1706115	0.1706115	-115.396632	-113.574889	33.356820
35	0.1725253	0.1628570	0.1628570	-118.343416	-115.863101	55.202278
37	0.1837217	0.1554549	0.1554549	-126.566394	-123.793218	56.896197
40	0.1939951	0.1483892	0.1483892	-131.781701	-128.537389	66.556235
44	0.2035853	0.1416447	0.1416447	-134.251223	-130.318750	83.920506
47	0.2127146	0.1352067	0.1352067	-138.217528	-133.723504	94.829546
49	0.2214106	0.1290614	0.1290614	-143.710843	-138.820624	99.253128
50	0.2293915	0.1231953	0.1231953	-150.423441	-145.328536	97.498979
53	0.2367391	0.1175959	0.1175959	-152.444656	-146.709185	110.353109

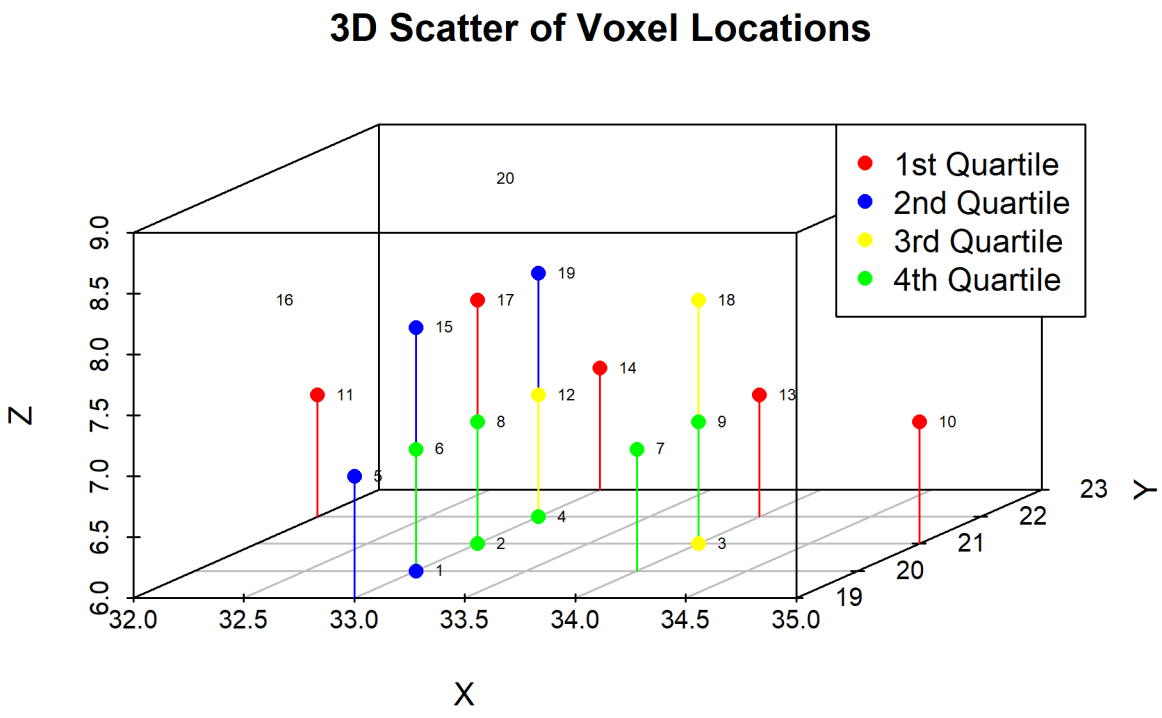
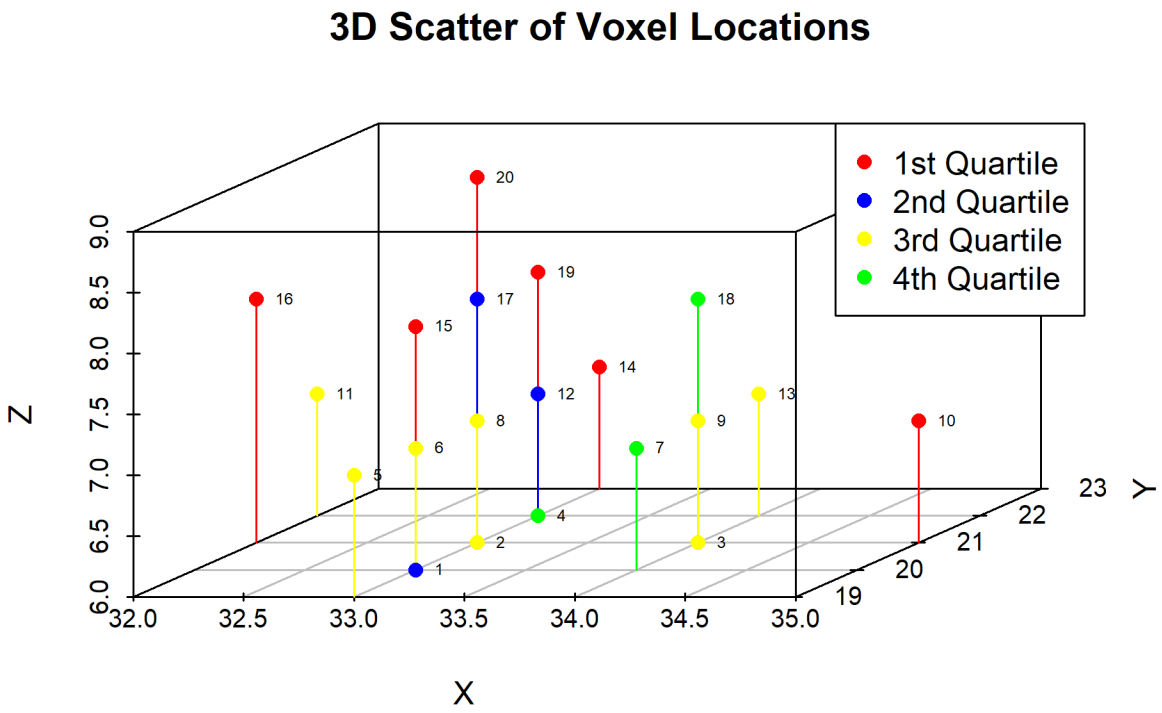


Figure 3: LASSO Prediction Correlation 3D Scatter



The 3D scatterplots shown here are shaded by the relative accuracy models predicting each voxel. Interestingly, both the lasso and random forest models had a similar relative ordering of how accurate their models were on any given model. However, the random forest performed much more admirably in terms of its overall accuracy. The summary statistics for the lasso's accuracy are:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.1978  0.3604  0.4713  0.4445  0.5372  0.5948       2
```

Whereas the summary statistics for the random forest are:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7432  0.7852  0.8125  0.8054  0.8217  0.8510
```

The random forest was much more consistent, being able to achieve predictions with 80% correlation, plus or minus 2% across nearly every voxel. Meanwhile, even the best lasso model only had a correlation of about .5661. This suggests that even though the random forest can be more computationally intensive and has some black-box qualities, its predictive accuracy was hard to beat in this case.

Diagnostics

Next, I explore some standard regression diagnostics for the LASSO model. Regression diagnostics are typically used to assess the overall fit of a model, detect outliers, and check if the modeling assumptions are reasonable. This process is important because poor model fit indicates that there may be a better model choice, and influential points can sometimes severely bias an estimate, thus compromising out-of-sample predictions.

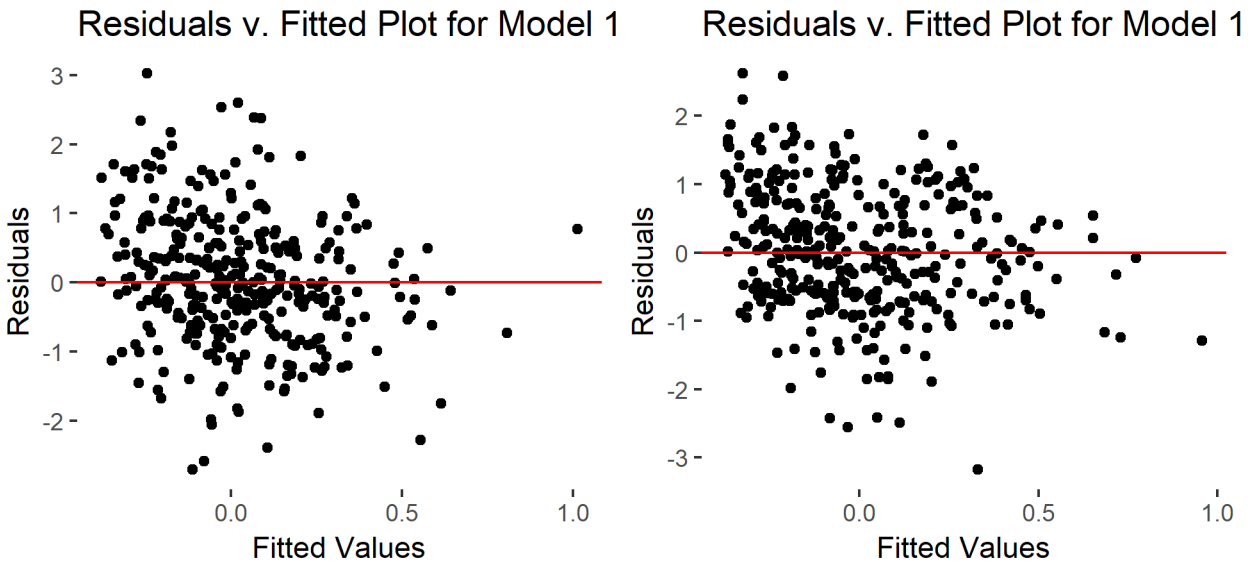


Figure 4: Residuals v. Fitted Plots

Residuals v. Fitted Plot

First, I look as a residuals v. fitted plot. This is a standard way to assess whether the data meet the homoscedasticity assumption, or in other words, that the standard deviations of the features are constant and do not depend on the x-value. Looking at plots of model 1 and 15 (predicting on voxels 1 and 15 respectively), there is some evidence that this assumptions has been violated. For a homoscedastic plot, the variances would be uniformly distributed across the window, but in this case there is a downward fan shape in both of the plots. Although there is heteroscedasticity, this is not necessarily fatal. It is likely that the standard errors will be larger than ideal, which can compromise prediction performance, but the model should nonetheless be in the general ballpark of the truth.

Outlier Detection

Next, I check for any outliers. This is an important step because regression estimates are highly susceptible to large outliers. To do this, I use a standard boxplot to visualize the spread of response data for voxel 1 in the validation set. The voxel responses seem to be tightly concentrated around 0, with most of the data lying between -.6 and .6. However, there are a good number of observations that are more extreme than the central tendency. Assuming this distribution was similar in the training set, and not particular to the validation set, this could cause a problem if a machine learning model attempts to capture the noise generated by the more extreme observations.

Model Interpretation

Importance

An important task at this stage is to identify the important features that are useful for predictions, and make decisions about how to potentially refit the model. To assess variable importance, I use the same implementation that the "caret" package employs. This function returns a number between 0 (not important) and 1 (very important) that indicates how important each feature was in making the prediction. I identified the most important variables, and remove any features that did not meet the threshold of .2 importance for

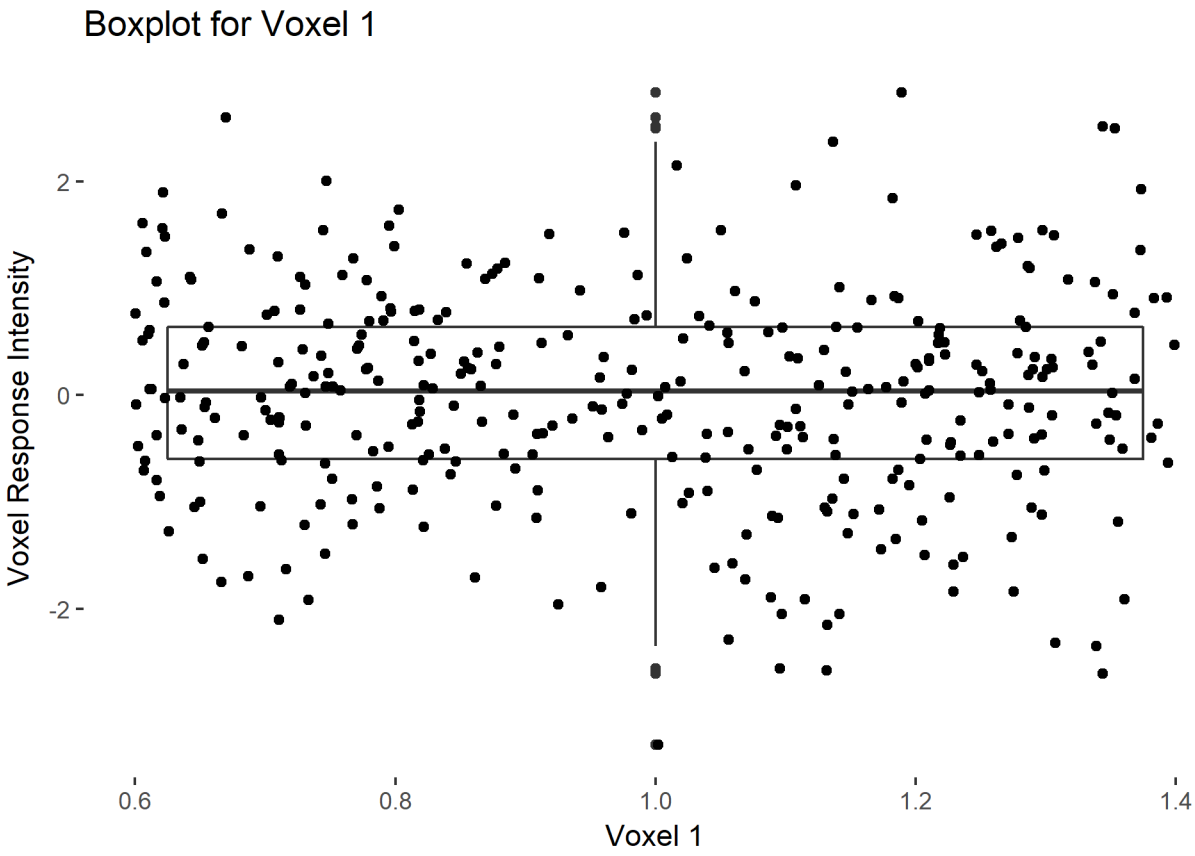


Figure 5: Boxplot of Voxel 1

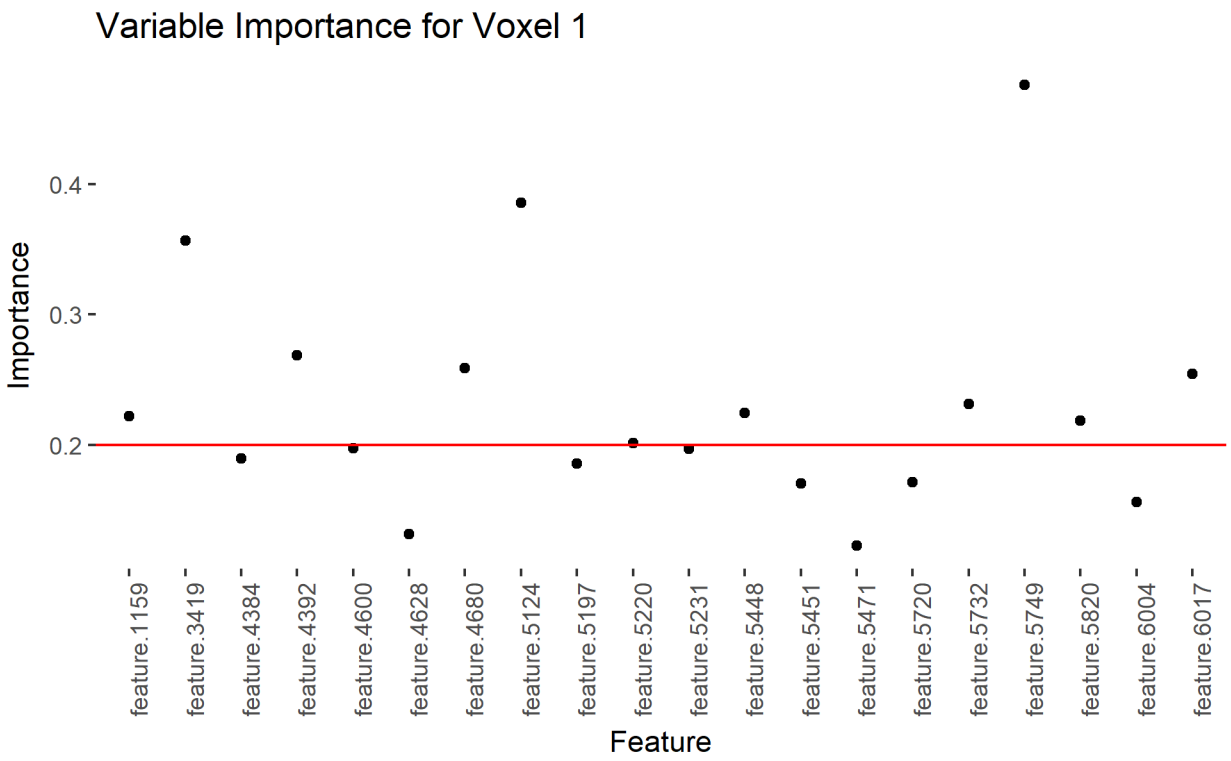


Figure 6: Variable Importance

any of the voxels. Below is a plot illustrating the importance of features for predicting voxel 1, to illustrate the general concept.

Overall, fewer than 300 features ended up being important for any of the models whatseover. This is obviously considerably fewer than the nearly 11,000 in the original data, so the variable importance examination simplifies the model. This is also advantageous because it reduces the effects of multicollinearity in the data by removing potentially redundant features.

Hypothesis Testing

In terms of hypothesis testing, I use the test proposed by Lockhart et. al. in "A significance test for the lasso." The authors propose a test statistic for the lasso that relies on a *covariance test statistic.* The basic intuition underlying the test is that the significance of a predictor variable is calculated as it enters the active set based on the covariance between the predictor and the target. The statistic overcomes the usual limitations of the chi-square and F-tests because it inherently accounts for the adaptive nature of lasso, whereas the conventional tests are based on linear regression.

Scientific Takeaway: What can Be Learned about Voxel Responses?

The big lessons from this process are that many of the features are not useful for predicting voxel responses, and spatially distinct voxels probably correspond to similar biological regions of the brain. These findings suggest that the bulk of fMRI signals are not that useful for predicting most voxel responses. This means that the signals either serve some other purpose, or the features are highly associated with particular regions of the brain, and are therefore not too useful for predicting the responses of other parts of the brain.

Delving deeper, I suspect that the V1 region of the brain that was used in this study has sub-regions within it that respond differently to images. As was seen in the 3D scatterplot showing the relative accuracy the algorithms had on different voxels, the best performing algorithms corresponded to voxels that were close together. Meanwhile, voxels like voxel 20 (top of the brain region), voxel 16 (front), and voxel 10 (back) were basically unrelated to the other voxels both spatially and substantively (See Figure 1). They shared little response similarity to the other voxels, and were consistently the toughest voxels to predict responses for.

I also explored whether the associations between different regions of the brain were strong enough to pool information for prediction purposes. I was hoping that by averaging the responses for all of the voxels in the bottom layer (voxel 1's layer) for instance, I could ameliorate any idiosyncrasies in the training data. However, I did not substantially improve prediction performance when I tried this method, which suggests that there may be a better way to make use of these regional correlations.

The key takeaway for me is that the features are not space-less numbers. Rather, they seem to clearly relate to particular regions of the brain, and interact in complex ways to determine the intensity that any given region experiences. There may also be interactions between different regions of the brain that I did not explore that explain response intensity.

Out-of-Sample Prediction

Finally, I proceed to the out-of-sample prediction. I choose to employ the random forest model because it generally performed very well compared to the LASSO. I attempted to retrain the model after removing some of the outliers from the training data, in hopes that the by removing the extreme values, the model will capture more of the true signal. However, this drastically cut the performance of the model (correlation down to about .5), which suggests that the outliers are not as noisy as I first suspected.

After fitting to the test data, I once again got a correlation of about .8 between my predictions and the actual data. Then, I predict the values based on the "val_feat" dataset, and save these predictions.

Conclusion

Overall, I identified several paths forward from the analysis done in this lab. In terms of the modeling, the lasso performed quite poorly, which is surprising given its attractive features for both regularization and variable selection. The random forest did much better, and was probably worth the additional computational resources. I hoped to construct an ensemble model that incorporated other modeling techniques as well, as I suspect this would have led to a more substantial gain than I was able to achieve with the random forest alone (which is itself an ensemble of CARTs). Unfortunately I ran out of time for this, as well as investigating more completely whether it would be fruitful to borrow information from neighboring voxels. That being said, these are potential gains on top of what I have done here.

Otherwise, the big lesson for me was learning more about how different regions of the brain respond to images. I did not have a strong sense of how fMRI signals mapped into brain activity, but there was enough evidence in the data to persuade me that the features have some physical interpretation. This makes sense as fMRI signals are basically measures of blood flow. Arteries and veins do not respect voxel boundaries, and it was illuminating to see which regions of the brain could be grouped together.

This lab made clear that regression prediction has difficulties that may not show up with classification problems. Whereas classification can be tackled with CARTs as well as other non-parametric methods like support vector machines (SVMs), this problem was more closely linked with canonical regression analysis. The high-dimensional nature of the data made the estimation conceptually and computationally difficult, hence being perfect for a machine learning-based approach. One final note is that I pondered whether a multi-target approach (predict multiple voxels at once) would be more appropriate than predicting each voxel separately, but ultimately I decided I did not have the substantive expertise to make a strong theoretical

claim about whether the responses were naturally related to each other. In sum, the problem was a great example of high-dimensional prediction, and the scientific issue of how voxels respond to the features was shown to be heavily dependent on space.

P.S. Thanks for a great semester! Happy holidays :)