Ex 3 - fMRI Data

Paz Bunis & Tal Aviel

In the following sections we train and analyse linear regression models which predict the fMRI single-voxel response values to different images.

We show that high prediction accuracy can be achieved when the images are represented in the Gabor wavelet basis. In addition, we show that one of the voxels shows high spatial specificity, i.e. focuses on a specific location in the presented image.

```
library('glmnet')
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-10
```

```
load('wavpyr.RData')
load('stim.RData')
```

Preparing the Data Sets

```
data_sets = load("fMRIclass.RData")
X=fit_feat
y=fit_data

set.seed(101)
sample <- sample.int(n = nrow(X), size = floor(.75*nrow(X)), replace = F)
X_train <- X[sample, ]
X_test  <- X[-sample, ]
y_train <- y[sample, ]
y_test  <- y[-sample, ]
```

1 Training Prediction Models

```
y_hat = matrix(ncol=15, nrow=nrow(val_feat)) # dataset with unknown labels - need to hand in
y_hat_test = matrix(ncol=15, nrow=nrow(X_test))
best_voxel_mse = -1
best_voxel_betahat = -1
best_voxel_num = 1
mses_per_voxel = c()
for (voxel in 1:15) {
  fit=glmnet(X_train,y_train[,voxel])
  y_test_hat_all_lambdas=predict(fit, X_test)

  mse_per_lambda=colMeans((y_test_hat_all_lambdas-y_test[,voxel])^2)

  min_mse = min(mse_per_lambda)
  mses_per_voxel[voxel] = min_mse
  best_mse_lambda=fit$lambda[which.min(mse_per_lambda)]
```

```
  betahat=coef(fit,s=best_mse_lambda)

  y_hat[, voxel] = predict(fit, val_feat, s=best_mse_lambda)
  y_hat_test[, voxel] = predict(fit, X_test, s=best_mse_lambda)

  if (best_voxel_mse == -1 || best_voxel_mse > min_mse) {
    best_voxel_mse = min_mse
    best_voxel_betahat = betahat
    best_voxel_num = voxel
  }
}
```

2 Analysis of Results

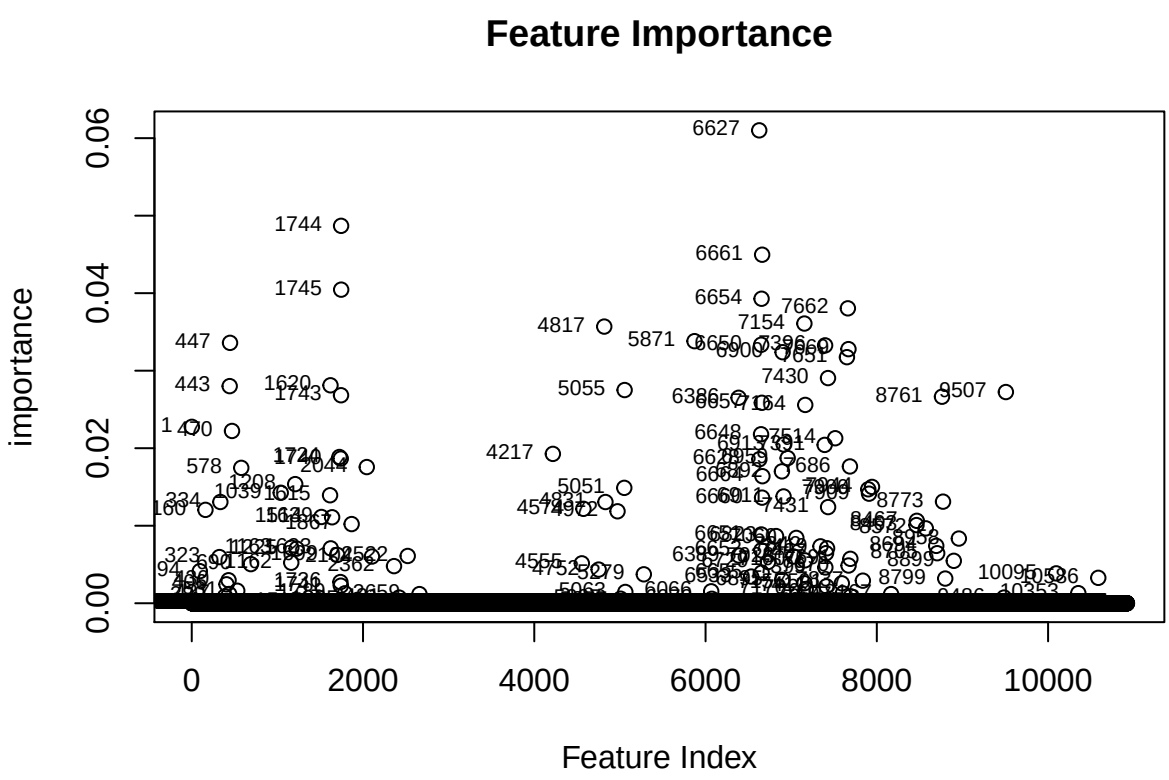
2.1 Best Voxel's Model

After we found the best model out of the 15 models, we calculate the importance of the model's features.

The importance of feature i is defined as $|\hat{\beta}_i| \cdot \text{sd}(X_i)$, where X_i is the ith column of the sample matrix X.

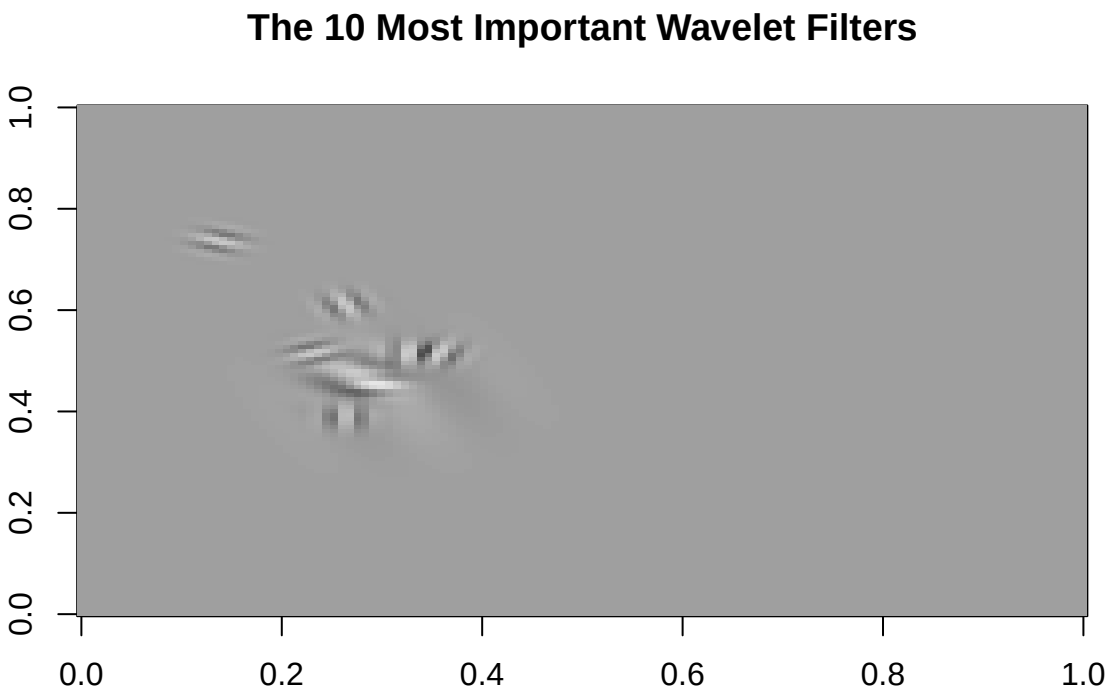
```
feat_sds = apply(X_train, 2, sd)
feat_importance = apply(X_train, 2, sd) * abs(best_voxel_betahat[-1])
sorted = sort(feat_importance, decreasing=T, index.return=T)
N=10
topN = sorted$ix[1:N]

plot(feat_importance, xlab = "Feature Index", ylab = "importance", main = "Feature Importance")
text(feat_importance, labels=1:10921, cex= 0.7, pos=2)
```



We can now look at the model's top 10 most important features (wavelet filters) combined as one image:

```
important_feats_img = matrix(0,nrow = 128, ncol = 128)
for (i in 1:N) {
  important_feats_img = important_feats_img + t(matrix(Re(wav.pyr[,topN[i]]), nrow = 128)[128:1,]);
}
image(important_feats_img, col = grey.colors(100), main = "The 10 Most Important Wavelet Filters")
```

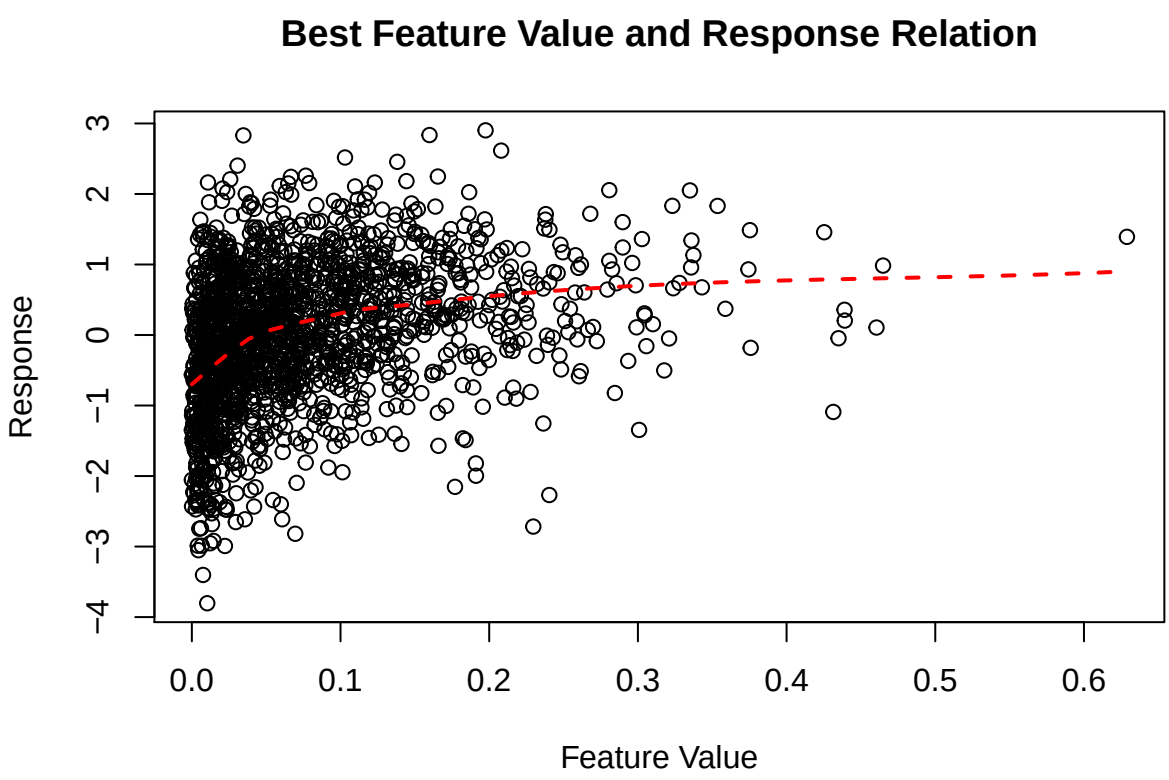


It looks like the model focuses on the center-left half of the image, and predicts a response based on the brightness and edges in that area.

2.2 Best Feature

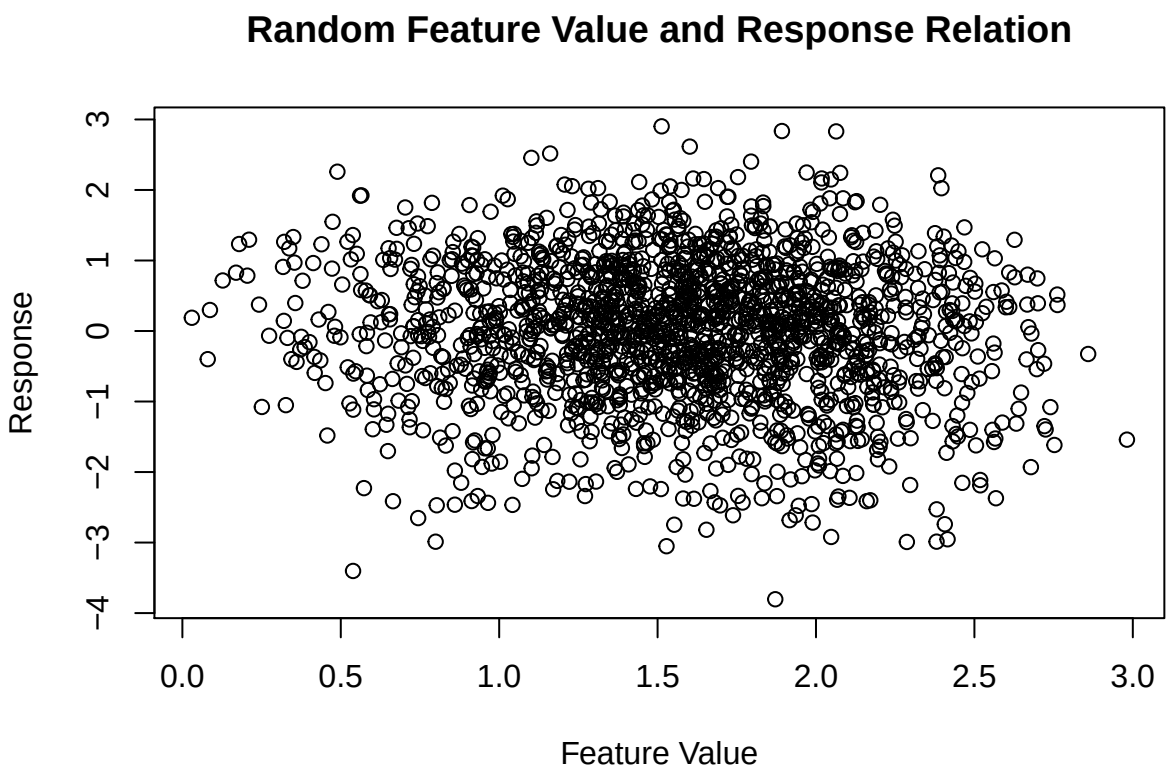
In order to understand the relationship between the best feature and the response, we plot the feature's value and the response for all 1750 samples, in addition to a locally weighted scatterplot smoothing line (red).

```
bestFeature = topN[1]
plot(X[,bestFeature], y[,best_voxel_num], main = "Best Feature Value and Response Relation",
     xlab = "Feature Value", ylab = "Response")
lines(loess.smooth(X[,bestFeature], y[,best_voxel_num]), col="red", lty=2, lwd=2)
```



As we can see, it looks like there is some type of correlation between the feature value and the response, but it isn't necesarrily a linear relation. Below we present the same plot, but use a random feature (instead of the best feature), and see that there is no correlation between its value and the response.

```
plot(X[,7], y[,best_voxel_num], main = "Random Feature Value and Response Relation",
     xlab = "Feature Value", ylab = "Response")
```



In order to evaluate the correlation between the best feature's value and the response, we compute the Spearman and Pearson correlation coefficients:

```
print(cor.test(X[,bestFeature], y[,best_voxel_num], method = "spearman"))
```

```
##
##  Spearman's rank correlation rho
##
## data:  X[, bestFeature] and y[, best_voxel_num]
## S = 566260000, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.3660572
```

```
print(cor.test(X[,bestFeature], y[,best_voxel_num], method = "pearson"))
```

```
##
##  Pearson's product-moment correlation
##
## data:  X[, bestFeature] and y[, best_voxel_num]
## t = 13.369, df = 1748, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2614511 0.3464905
## sample estimates:
##       cor
## 0.3045776
```

These coefficients and low p-values suggest that there exists a positive correlation.

2.3 Overshoot and Undershoot

Overshooting occures when $\hat{y}$ is substantially larger than y, and undershooting is the opposite effect. The most significant samples in which these effects happen can be found as follows:

```
diff = y_test[,best_voxel_num] - y_hat_test[,best_voxel_num]
sorted_diff = sort(diff, decreasing=F, index.return=T)
test_idx_map = (1:1750)[-sample]
```

2.3.1 Overshoot

In the following images we see the highest overshoot effect, with the best filter embedded on them:

```
par(mfrow=c(3,4), mar = c(0.5,0.5,0.5,0.5))
overshoots = sorted_diff$ix[1:10]
for (i in 1:10) {
  image(t(matrix(stim[test_idx_map[overshoots[i]],], nrow = 128)[128:1,]), axes=FALSE,
        frame.plot=TRUE,col = grey.colors(100))
  image(t(matrix(Re(wav.pyr[,topN[1]]), nrow = 128)[128:1,]), col = heat.colors(100, alpha=0.2),
        add = T)
}
```

In these cases the filter is located in a light area, or an area in which there is an egde in a specific orientation.

2.3.2 Undershoot

In the following images we see the highest undershoot effect, with the best filter embedded on them:

```
l = length(sorted_diff$x)
undershoots = rev(sorted_diff$ix[(l-10):l])

par(mfrow=c(3,4), mar = c(0.5,0.5,0.5,0.5))
for (i in 1:10) {
  image(t(matrix(stim[test_idx_map[undershoots[i]],], nrow = 128)[128:1,]), axes=FALSE,
        frame.plot=TRUE ,col = grey.colors(100))
  image(t(matrix(Re(wav.pyr[,topN[1]]), nrow = 128)[128:1,]), col = heat.colors(100, alpha=0.2),
        add = T)
}
```

In these cases the filter is located in a dark area, or an area in which there is an egde in the opposite orientation of the filter.

2.4 Changes in the Prediction Model

In section 2.2 we see that the relation between the values of the best feature and the responses isn't necessarily linear. This suggested that if we would have applied a transformation on the feature values, the linear model might have performed better.

3 Predicted MSEs and Responses

Save the predicted MSEs and predicted responses for the val_feat samples:

```
save(mses_per_voxel, y_hat, file="predictions")
```