# Linear Model Selection & Validation

Marjolein Fokkema

# Quiz questions on Wooclap

https://app.wooclap.com/SLWEEK5

Figure 1: Wooclap QR code

# Chapter 6: Linear Model Selection and Regularization

What if we have (too) many features?

Three classes of methods:

- Subset selection
- Regularization
  - Minimizing fit + penalty, a very powerful idea! Used in penalized regression, decision trees, tree ensembles, hierarchical models, smoothing splines / GAMs, ...
- Dimension reduction (ISLR section 6.3; next week)

# Chapter 6: (Linear) Model Selection and Regularization

- Generalizations to other response variables types within the GLM (Poission, Binomial, etc.) are straightforward.
  - Video 2 chapter 6: Replace RSS with Deviance (-2LL) in the fit-plus-penalty criterion.
- Using generalizations is easy: E.g, in package `glmnet`, simply specify different `family`.

# Question

# Best subset and stepwise selection (penalty on L0 norm)

- OLS coefficients $\hat{\beta}^{OLS}$ minimize:

$$RSS = \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)$$

- Best subset and stepwise selection (forward, backward or both) also minimize a fit-plus-penalty criterion:

$$\sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right) + \lambda \sum_{j=1}^{p} I(\beta_j \neq 0)$$

- Right-most sum is often referred to as L0 norm: $||\beta||_0$, which is the number of non-zero elements.

# Best subset selection

Trying $2^p$ combinations is computationally prohibitive.

- Many algorithms have been developed to speed up the search, allowing for (much) larger $p$.
- Best subset can work well in problems with high signal-to-noise ratio (i.e., low $\sigma^2$).

# Stepwise selection (forward and/or backward)

Stepwise regression has a pretty bad name, because of widespread incorrect use of:

- Standard errors and p-values computed and reported as if no variable selection has taken place.
- Degrees of freedom used up by the model assumed to be equal to the number of selected variables.
- Fit measures like $R^2$ computed on data that was used for variable selection.

Solution: After selecting variables on the training data, perform inference or evaluate performance on new set of (validation) data!

# Shrinkage methods

Ridge regression coefficient estimates $\hat{\beta}^R$ minimize:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right) + \lambda \sum_{j=1}^{p}\beta_j^2$$

- Right-most sum often referred to as squared L2 norm: $||\beta||_2^2$

Lasso regression coefficients estimates $\hat{\beta}^L$ minimize:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right) + \lambda \sum_{j=1}^{p}|\beta_j|$$

- Right-most sum is often referred to as L1 norm: $||\beta||_1$

# Questions

# Computational challenges

Even if optimal value of $\lambda$ is known or given, minimizing the fit-plus-penalty criterion can be challenging:

- With L0 norm: Derivative of the fit-plus-penalty criterion w.r.t. $\beta$ is zero in many places. But where it's interesting, it has jump discontinuities and is not differentiable.
- With L1 norm: Not differentiable with respect to a coordinate where that coordinate is zero. Elsewhere, the partial derivatives are just constants, $\pm 1$ depending on the quadrant.
- With L2 norm: Differentiable, if we use the squared L2 norm it's differentiable even at zero.

# Ridge and degrees of freedom

OLS coefficients can be obtained as follows:

$$\hat{\beta}^{OLS} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\hat{y}^{OLS} = \mathbf{X}\hat{\beta}^{OLS} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{P}\mathbf{y}$$

- $\mathbf{P}$ is the projection matrix, a.k.a. 'hat' matrix.
- Values on the diagonal of $\mathbf{P}$ quantify how much an observation contributes to its own predicted value.
- By definition, in OLS the trace (sum of diagonal elements) of $\mathbf{P}$ is equal to the rank of $\mathbf{X}$, which is the number of independent parameters.

# Ridge solution and degrees of freedom

Ridge coefficients can be obtained as follows:

$$\hat{\beta}^R = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\hat{y}^R = \mathbf{X}\hat{\beta}^R = \mathbf{X}(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{P}_\lambda \mathbf{y}$$

- For ridge, the *effective* degrees of freedom are given by $\mathrm{tr}(\mathbf{P}_\lambda)$.
- Values on the diagonal $\mathbf{P}_\lambda$ are $\leq$ values on the diagonal of $\mathbf{P}$ from OLS: Predicted values are shrunken towards the mean (like coefficients are shrunken towards zero).

# Useful extensions: Elastic Net

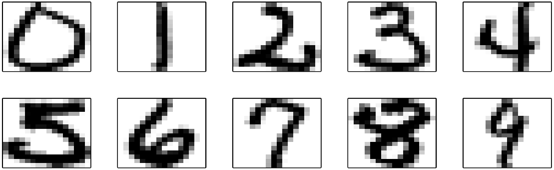
Both Ridge and Lasso penalties are added to the criterion:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right) + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{p}\beta_j^2 + \alpha\sum_{j=1}^{p}|\beta_j|\right)$$

- Where $\alpha$ determines the weight of the Lasso and Ridge penalties.
- Note that now two hyperparamers need to be optimized!
- Question: What penalties result when we set $\alpha = 0$? And when we set $\alpha = 1$?

# Elastic net: Digit recognition example

- Task: Recognize hand-written digits from 16x16 grayscale images.
- Data: 7291 training samples, 2007 test samples.
- Predictor variables: 256 grayscale values (one for each pixel).
- 10-class response (digits 0-9)



- Question: What do you expect for signal-to-noise ratio (low or high)? Multicollinearity?
- In this example, we will perform binary classification of digits 2 ($y = 0$) and 3 ($y = 1$).

# Elastic net: Digit recognition example

```
library("glmnet")
set.seed(42)
L_mod <- cv.glmnet(x = x, y = y, family = "binomial",
                   alpha = 1)
L_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, family = "binomial", alph
##
## Measure: Binomial Deviance
##
##        Lambda Index Measure      SE Nonzero
## min 0.001109    63 0.08893 0.01356      74
## 1se 0.003086    52 0.10145 0.01076      66
```

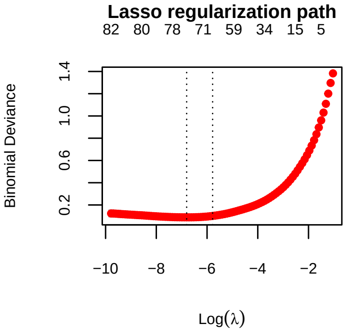
# Elastic net: Digit recognition example

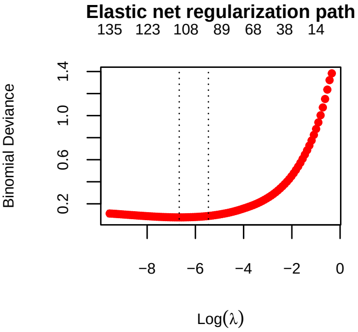
```r
set.seed(42)
EN_mod <- cv.glmnet(x = x, y = y, family = "binomial",
                    alpha = .5)
EN_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, family = "binomial", alph
##
## Measure: Binomial Deviance
##
##       Lambda Index Measure      SE Nonzero
## min 0.001269    69 0.07701 0.01388     112
## 1se 0.004255    56 0.08950 0.00994     101
```

# Elastic net: Digit recognition example

```
par(mfrow = c(1, 2))
plot(L_mod, main = "Lasso regularization path")
plot(EN_mod, main = "Elastic net regularization path")
```



# Elastic net: Digit recognition example

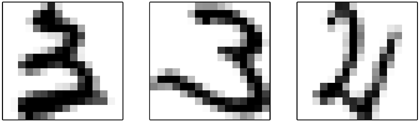
Correct classification rates on training data:

- Lasso: 0.9985601
- Elastic Net: 0.9985601

Correct classification rates on test data:

- Lasso: 0.956044
- Elastic Net: 0.9642857

Misclassified by Lasso, correctly classified by Elastic Net:



(Lasso predicted 2, 2, 3; Elastic Net predicted 3, 3, 2)

# Useful extensions: Relaxed Lasso

- Lasso performs shrinkage and selection. Both strength and weakness!
- $\lambda$ optimized for selection will likely not be optimal for shrinkage, vice versa.
  - In order to shrink many coefficients to zero, large coefficients will be shrunken too heavily.
- Relaxed Lasso:
  - Use Lasso for variable selection
  - Refit OLS on selected predictors only.
  - Compute final coefficients as a weighted version:

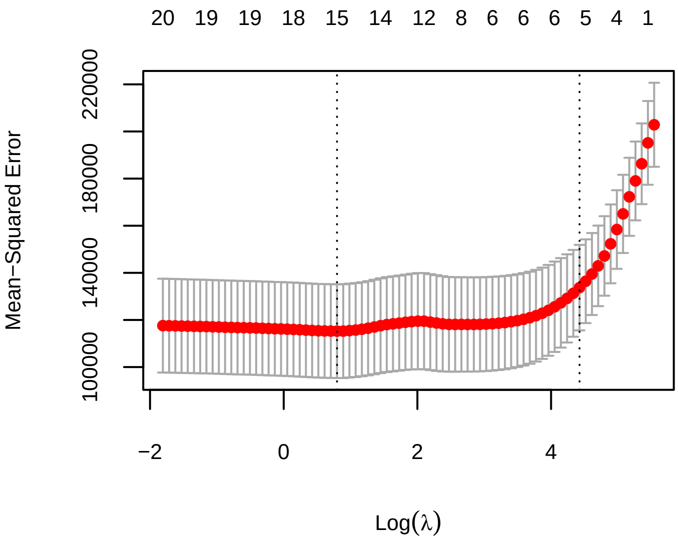
$$\hat{\beta}_{\text{relaxed}} = (1-\gamma)\hat{\beta}_{\text{OLS}} + \gamma\hat{\beta}_{\textit{Lasso}}$$

# Relaxed Lasso: Predicting baseball player's salaries

- "Hitters" data: Major League Baseball data (from 1986-1987), $N = 263$.
- Task: Predict player's salary.
- 19 predictors: Times at bat, number of homeruns, number of walks, for many variable in '86 and '87 season.

```
library("glmnet")
set.seed(42)
cv_lasso <- cv.glmnet(x, y) ## 'standard' lasso
plot(cv_lasso)
```

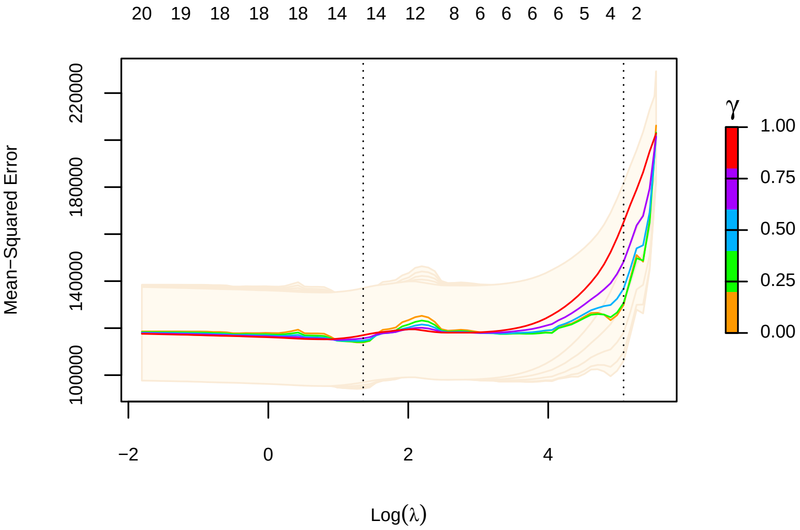
# Relaxed Lasso: Predicting baseball player's salaries



# Relaxed Lasso: Predicting baseball player's salaries

```r
library("glmnet")
set.seed(42)
cv_relax <- cv.glmnet(x, y, relax = TRUE)
plot(cv_relax)
```

# Relaxed Lasso: Predicting baseball player's salaries



# Relaxed Lasso: Predicting baseball player's salaries

```
lasso_coefs <- as.matrix(coef(cv_lasso))
kable(t(lasso_coefs[lasso_coefs != 0,]),
      format = "latex", digits = 3)
```

| (Intercept) | Hits | Walks | CRuns | CRBI | PutOuts |
|---:|---:|---:|---:|---:|---:|
| 167.912 | 1.293 | 1.398 | 0.142 | 0.322 | 0.047 |

```
relax_coefs <- as.matrix(coef(cv_relax))
kable(t(relax_coefs[relax_coefs != 0,]),
      format = "latex", digits = 3)
```

| (Intercept) | Hits | Walks | CRuns | CRBI |
|---:|---:|---:|---:|---:|
| 41.765 | 1.921 | 2.339 | 0.15 | 0.413 |

# Reading materials

What to focus on in the book (ISLR chapter 6):

- Lasso and Ridge penalties as Bayesian priors.
- Penalties as a "spending budget"
  - We'll meet regularization with a penalty again with decision trees and smoothing splines.
  - We'll meet regularization with a budget again with support vector machines.

What to focus on in the paper (Hastie et al., 2020):

- Which method works best in which situation? Best subset, forward stepwise, lasso, relaxed lasso.