Erasmus
School of
Economics

Supervised Machine Learning
Week 2

Patrick J.F. Groenen

2020-2021

Erasmus University Rotterdam

Table of Contents

Summary

Summary:

Week	Topics	Material
1	Introduction; Introduction to R; Linear methods for regression, model selection, and assessment	3.1, 3.2, 3.3
2	Regularized regression and k-fold cross validation	3.4.1-3.4.3, 3.8.4, 7.10
3	Basis function expansions, kernels, bias-variance trade-off	5.1-5.2.1, 5.8, 7.3
4	Support vector machines	Groenen, Nalbantov, Bioch (2009); 12.1-12.3
5	Classification and regression trees, random forests, bootstrap	7.11, 9.2, 15
6	Boosting	10

Introduction

Material this week:

Topic		To read
1.	Ridge regression	3.4.1
2.	Lasso	3.4.2
3.	Grouped Lasso	3.8.4
4.	K-fold and leave-one-out cross validation	7.10

Table of Contents

Ridge Regression

Main properties of ridge regression:

1. Adaptation of multiple regression.
2. Penalizes the size of the coefficients.
3. Very useful under multicollinearity of predictors.
4. Very useful with many predictors.
5. Also called: penalty or regularization approach.
6. Assumption: y is in deviation of its mean (so no need to estimate the intercept).
7. All weights β_j are equally penalized.

Ridge Regression

Loss function ridge regression:

$$L(\boldsymbol{\beta}) \quad = \quad \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\boxed{\text{Regression term}}} \quad + \quad \underbrace{\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}}_{\boxed{\text{Penalty term}}}$$

with

- $\boldsymbol{\beta}$: unknown $p \times 1$ vector of regression weights
- $\mathbf{X}$: $n \times p$ matrix with elements x_{ij} of predictor variables
- $\mathbf{y}$: $n \times 1$ vector with response variable
- λ: positive (given) penalty strength parameter

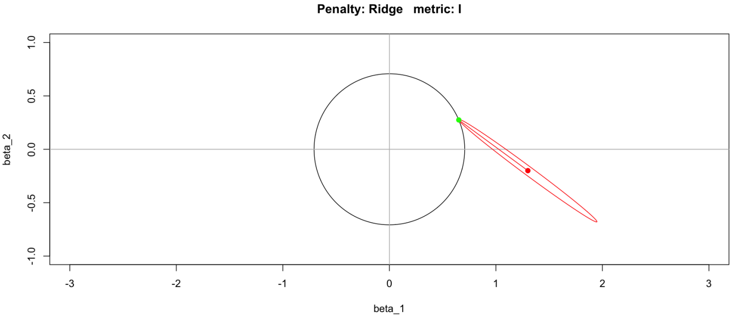
Ridge Regression

Mathematically the following two definitions of ridge regression are the same:

$$
\begin{aligned}
L_{\text{ridge1}}(\boldsymbol{\beta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^\top\boldsymbol{\beta} \\
L_{\text{ridge2}}(\boldsymbol{\beta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \text{ subject to } \boldsymbol{\beta}^\top\boldsymbol{\beta} \leq \gamma \text{ for } 0 < \gamma \leq \boldsymbol{\beta}_{\text{OLS}}^\top\boldsymbol{\beta}_{\text{OLS}}
\end{aligned}
$$

- $\boldsymbol{\beta}^\top\boldsymbol{\beta} \leq \gamma$ says that $\boldsymbol{\beta}$ must be in a circle (hyperball) with radius smaller than (or equal to) γ.
- For each λ there is a corresponding γ (and vice versa).
- Why is $L_{\text{ridge1}}(\boldsymbol{\beta})$ equivalent to $L_{\text{ridge2}}(\boldsymbol{\beta})$?

Ridge Regression



Ridge Regression: Example

Example ridge regression: `Credit` data

- Goal: predict `Balance` out of other variables

```
R> ## Show Credit data
R> load("Credit.RData")
R> summary(Credit)
     Income           Limit            Rating          Cards            Age
 Min.   : 10.4   Min.   :  855   Min.   : 93   Min.   :1.00   Min.   :23.0
 1st Qu.: 21.0   1st Qu.: 3088   1st Qu.:247   1st Qu.:2.00   1st Qu.:41.8
 Median : 33.1   Median : 4622   Median :344   Median :3.00   Median :56.0
 Mean   : 45.2   Mean   : 4736   Mean   :355   Mean   :2.96   Mean   :55.7
 3rd Qu.: 57.5   3rd Qu.: 5873   3rd Qu.:437   3rd Qu.:4.00   3rd Qu.:70.0
 Max.   :186.6   Max.   :13913   Max.   :982   Max.   :9.00   Max.   :98.0
   Education         Gender     Student    Married              Ethnicity
 Min.   : 5.0    Male  :193   No :360   No :155   African American: 99
 1st Qu.:11.0   Female:207   Yes: 40   Yes:245   Asian           :102
 Median :14.0                                     Caucasian       :199
 Mean   :13.4
 3rd Qu.:16.0
 Max.   :20.0
    Balance
 Min.   :   0
 1st Qu.:  69
 Median : 460
```

Ridge Regression: Example

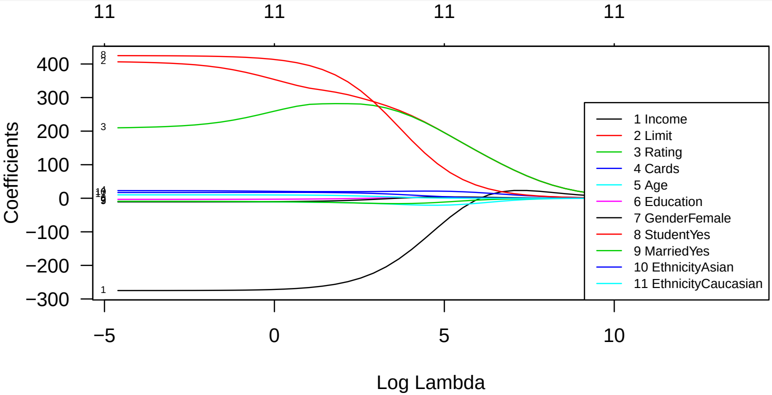
Example ridge regression: Credit data

```
R> ## Ridge regression
R> Credit <- na.omit(Credit)                          # Remove rows with missings (NA)
R> y <- as.vector(Credit$Balance)                     # y variable
R> # Credit contains some nominal variable
R> # Create a numeric matrix of dataframe Credit without last column (Balance) and intercept
R> X <- model.matrix(Balance ~ . , data = Credit)    # Predictor variables (as a matrix, not da
R> X[, 2:7] <- scale(X[, 2:7])                        # Make columns z-scores of nonfactors
R> X <- X[, -1]                                       # Remove intercept column (is fitted by gl
R> library(glmnet, quietly = TRUE)
R> result <- glmnet(X, y, alpha = 0, lambda = 10^seq(-2, 6, length.out = 50),
                  standardize = FALSE)                 # Ridge regression (alpha must be 0 for r
```

Ridge Regression: Example

Example ridge regression profile plot of weights β_j against λ:

```
R> plot(result, xvar = "lambda", label = TRUE, las = 1)
R> legend("bottomright", lwd = 1, col = 1:6, bg = "white",
         legend = pasteCols(t(cbind(1:ncol(X), " ",colnames(X)))), cex = .7)
```



Ridge Regression: Properties

Properties:

- As $\lambda \to 0$ then $\boldsymbol{\beta}^{\text{ridge}} \to \boldsymbol{\beta}^{\text{OLS}}$
  (OLS = Ordinary Least Squares = standard multiple regression)
- As $\lambda \to \infty$ then $\boldsymbol{\beta}^{\text{ridge}} \to \mathbf{0}$
- If predictor variables in $\mathbf{X}$ are uncorrelated ($(n-1)^{-1}\mathbf{X}^\top\mathbf{X} = \mathbf{I}$) and standardized to z-scores:

$$\beta_j^{\text{ridge}} = \frac{1}{1+\lambda}\beta_j^{\text{OLS}}$$

- The $\lambda > 0$ results in shrinkage of the $\boldsymbol{\beta}$.
- Even in the case of multicollinearity, $\boldsymbol{\beta}^{\text{ridge}}$ always exists.

Ridge Regression: Computation

Solution ridge regression: solve for zero gradient (for fixed λ):

$$
\begin{aligned}
\frac{\partial L_{\text{ridge1}}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= 2(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})\boldsymbol{\beta} - 2\mathbf{X}^\top \mathbf{y} = \mathbf{0} \\
(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})\boldsymbol{\beta} &= \mathbf{X}^\top \mathbf{y} \\
\hat{\boldsymbol{\beta}} &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}
\end{aligned}
$$

Ridge Regression: Effect of standardization

What is the effect of different variances of the columns of $\mathbf{X}$?

- In OLS: none
  Let β_j^* be the optimal OLS weight for variable j.
  If variable X_j is replaced by aX_j, then the optimal OLS regression weight is β_j^*/a.
- In ridge regression there is an effect.
  Solution: Standardize columns to z-score (mean 0, variance 1).

Ridge Regression: Usage

Summary properties of ridge regression:

- Ridge regression shrinks the β_js towards zero.
- The tuning parameter λ controls strength of shrinkage.
- Choosing good value of λ by k-fold cross validation (discussed later).
- Ridge regression performs not so well if a subset of true coefficients is small or zero.
- Ridge regression performs well if all true β_js are moderately large.
- Often outperforms linear regression in a small range of λs.
- Standardization of predictors to z-scores is important.

Ridge Regression: GLMNET

Standardisation of $\mathbf{X}$ and $\mathbf{y}$ in `glmnet()`:

- Standardisation to z-scores of predictors $\mathbf{X}$ is vital so that all weights are equally penalized.
- `standardize = TRUE` option:
  - makes standardizes internally both predictors $\mathbf{X}$ and response $\mathbf{y}$ to z-scores.
  - all results are computed back to original mean and standard deviation.

Table of Contents

Ridge Regression and SVD

Definition singular value decomposition (SVD) for an $n \times p$ matrix $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

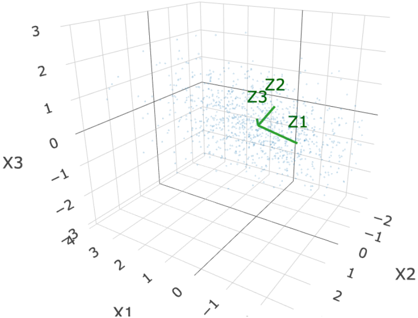
with

- the $n \times p$ orthonormal matrix $\mathbf{U}$ of left singular vectors with $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$,
- the $p \times p$ diagonal matrix $\mathbf{D}$ of singular values with $d_{ii} \geq 0$,
- the $p \times p$ orthonormal matrix $\mathbf{V}$ of right singular vectors with $\mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$.

Ridge Regression and SVD

Interpretation SVD $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$:

- $\mathbf{U}\mathbf{D} = \mathbf{Z}$ are the principal components
- $\mathbf{V}$ is the matrix that rotates $\mathbf{X}$ to principal components $\mathbf{Z}$.
- First column of $\mathbf{z}_1 = d_{11}\mathbf{u}_1$ has most variance equal to d_{11}^2.
- Example with $p = 3$:



Ridge Regression and SVD

```
R> ## Example SVD in R
R> X <- matrix(c(3, 5, 1, 2, 3, 2), 3, 2) # Make a matrix
R> print(X, digits = 0)                   # Print a matrix
     [,1] [,2]
[1,]    3    2
[2,]    5    3
[3,]    1    2
R> tt <- svd(X)                           # Compute an SVD: X = UDV'
R> print(tt$u, digits = 3)                # Print left singular vectors U
       [,1]    [,2]
[1,] -0.506 -0.0344
[2,] -0.818 -0.2987
[3,] -0.274  0.9537
R> print(tt$d, digits = 3)                # Print singular values D
[1] 7.12 1.14
R> print(tt$v, digits = 3)                # Print right singular vectors V
       [,1]   [,2]
[1,] -0.826 -0.564
[2,] -0.564  0.826
R> tt$u %*% diag(tt$d) %*% t(tt$v)        # Reconstruct matrix X through UDV'
     [,1] [,2]
[1,]    3    2
[2,]    5    3
[3,]    1    2
```

Ridge Regression and SVD

Properties SVD:

- The SVD exists for every rectangular matrix.
- $\mathbf{X}$ can be of every size (not necessarily square).
- The result always gives real elements (no imaginary numbers).
- The singular values d_{ii} are ordered decreasingly.
- In PCA: the squared singular value d_{ss}^2 is equal to the eigenvalue.

Ridge Regression and SVD

SVD as sum of rank 1 matrices:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top = d_{11}\mathbf{u}_1\mathbf{v}_1^\top + d_{22}\mathbf{u}_2\mathbf{v}_2^\top + \ldots + d_{pp}\mathbf{u}_p\mathbf{v}_p^\top$$

Example for 3×2 matrix $\mathbf{X}$

$$\begin{aligned}
\mathbf{X} &= \mathbf{U}\mathbf{D}\mathbf{V}^\top \\
&= d_{11}\mathbf{u}_1\mathbf{v}_1^\top + d_{22}\mathbf{u}_2\mathbf{v}_2^\top \\
&= 7.12 \begin{bmatrix} -0.506 \\ -0.818 \\ -0.274 \end{bmatrix} \begin{bmatrix} -0.826 & -0.564 \end{bmatrix} + 1.14 \begin{bmatrix} -0.034 \\ -0.299 \\ 0.954 \end{bmatrix} \begin{bmatrix} -0.564 & 0.826 \end{bmatrix} \\
&= \begin{bmatrix} 2.978 & 2.032 \\ 4.818 & 3.282 \\ 1.613 & 1.101 \end{bmatrix} + \begin{bmatrix} 0.022 & -0.033 \\ 0.192 & -0.282 \\ -0.613 & 0.899 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 5 & 3 \\ 1 & 2 \end{bmatrix}
\end{aligned}$$

Ridge Regression and SVD

Ridge Regression and SVD:

- Rewrite $\mathbf{X}\beta = \mathbf{UDV}^\top\beta$
- Let $\gamma = \mathbf{V}^\top\beta$. Then

$$
\begin{aligned}
\gamma &= \mathbf{V}^\top\beta \\
\mathbf{V}\gamma &= \mathbf{VV}^\top\beta = \beta \\
\mathbf{X}\beta &= \mathbf{UDV}^\top\beta = \mathbf{UD}\gamma
\end{aligned}
$$

- Effect on regression part:

$$
\begin{aligned}
(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) &= (\mathbf{y} - \mathbf{UDV}^\top\beta)^\top(\mathbf{y} - \mathbf{UDV}^\top\beta) \\
&= (\mathbf{y} - \mathbf{UD}\gamma)^\top(\mathbf{y} - \mathbf{UD}\gamma)
\end{aligned}
$$

- Effect on penalty part:

$$
\lambda\beta^\top\beta = \lambda\gamma^\top\mathbf{V}^\top\mathbf{V}\gamma = \lambda\gamma^\top\gamma
$$

Ridge Regression and SVD

Ridge Regression and SVD:

- Ridge loss function as with principal components $\mathbf{UD}$ of $\mathbf{X}$ as predictors:

$$\begin{aligned} L_{\text{ridge1}}(\beta) &= (\mathbf{y}-\mathbf{X}\beta)^\top(\mathbf{y}-\mathbf{X}\beta)+\lambda\beta^\top\beta \\ &= (\mathbf{y}-\mathbf{UD}\gamma)^\top(\mathbf{y}-\mathbf{UD}\gamma)+\lambda\gamma^\top\gamma \end{aligned}$$

- Alternatively, let $\boldsymbol{\delta}=\mathbf{D}\gamma=\mathbf{DV}^\top\beta$, then $\mathbf{D}^{-1}\boldsymbol{\delta}=\gamma$ and

$$L_{\text{ridge1}}(\boldsymbol{\delta}) = (\mathbf{y}-\mathbf{U}\boldsymbol{\delta})^\top(\mathbf{y}-\mathbf{U}\boldsymbol{\delta})+\lambda\boldsymbol{\delta}^\top\mathbf{D}^{-2}\boldsymbol{\delta}$$

  with $\lambda\boldsymbol{\delta}^\top\mathbf{D}^{-2}\boldsymbol{\delta}=\sum_{j=1}^{p} d_{jj}^{-2}\delta_j^2$

- Thus, ridge regression gives
  - the smallest penalty to the weight δ_1 corresponding to $\mathbf{u}_1$, the largest principal component and
  - the highest penalty to the weight δ_p corresponding to $\mathbf{u}_p$, the smallest principal component.

Table of Contents

Choosing Penalty Strength λ

Loss function ridge regression:

$$L(\beta_1, \ldots \beta_m) = \underbrace{\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} x_{ij}\beta_j \right)^2}_{\text{Regression term}} + \underbrace{\lambda \sum_{j=1}^{m} \beta_j^2}_{\text{Penalty term}}$$

with

- β_j: unknown regression weights for variable $j = 1, \ldots, m$
- X: with elements x_{ij} the $n \times m$ matrix of predictor variables
- y_i: value of dependent variable for object $i = 1, \ldots, n$
- λ: positive (given) penalty parameter

Choosing Penalty Strength λ

Degrees of freedom

- Degrees of freedom (df) in OLS: number of parameters β_j, thus df$= m$.
- Effective degrees of freedom (df_{eff}) in ridge regression:

$$\text{df}_{\text{eff}} = \text{tr}\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top = \sum_{i=1}^{n} \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{x}_i$$

  with $\text{tr}\mathbf{A} = \sum_{i=1}^{n} a_{ii}$.
- df_{eff} is equivalent to the effective number of parameters

$$\begin{aligned} \text{df}_{\text{eff}} &\to m \qquad \text{for } \lambda \downarrow 0 \\ \text{df}_{\text{eff}} &\to 0 \qquad \text{for } \lambda \to \infty \end{aligned}$$

Choosing Penalty Strength λ

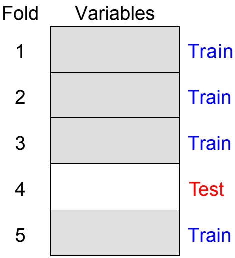
How to select λ?

- Try various $\lambda = \{0.001, 0.01, \ldots, 100, 1000\}$.
- Use one of the following selection criteria:
  1. Choose lowest AIC: $AIC = n \log\left(\sum_i (y_i - \hat{y}_i)^2\right) + 2\mathrm{df}_{\text{eff}} \log(n)$
  2. Choose lowest BIC: $BIC = n \log\left(\sum_i (y_i - \hat{y}_i)^2\right) + 2\mathrm{df}_{\text{eff}}$
  3. Choose lowest K-fold cross validated error.

Table of Contents

K-Fold Cross Validation

K-fold cross validation



K-Fold Cross Validation

K-fold cross validation

- Usual measure of error mean squared error for fold k:

$$\text{MSE}_k = \frac{1}{n_k}(\mathbf{y}_{\text{test}(k)} - \hat{\mathbf{y}}_{\text{test}(k)})^\top(\mathbf{y}_{\text{test}(k)} - \hat{\mathbf{y}}_{\text{test}(k)})$$

  with $\hat{\mathbf{y}}_{\text{test}_k} = \mathbf{X}_{\text{test}_k}\hat{\beta}_{\text{train}_k}$

- Easier to interpret is the root mean squared error (RMSE) over the folds:

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\text{MSE}_k}$$

- Sums the error of predicted out-of-sample values
- Is a good approximation of error in the population.

K-Fold Cross Validation

K-fold cross validation

1. Nonparametric approach.
2. Split the data into two parts $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{test}}$:
3. Use the training part $\mathbf{X}_{\text{train}}$ to estimate weights $\hat{\boldsymbol{\beta}}_{\text{train}}$
4. Estimate $\hat{\mathbf{y}}_{\text{test}} = \mathbf{X}_{\text{test}}\hat{\boldsymbol{\beta}}_{\text{train}}$
5. Repeat Steps 3 and 4 for each of the K folds.
6. Repeat for all λ values.

K-Fold Cross Validation

Example ridge regression: `Credit` data

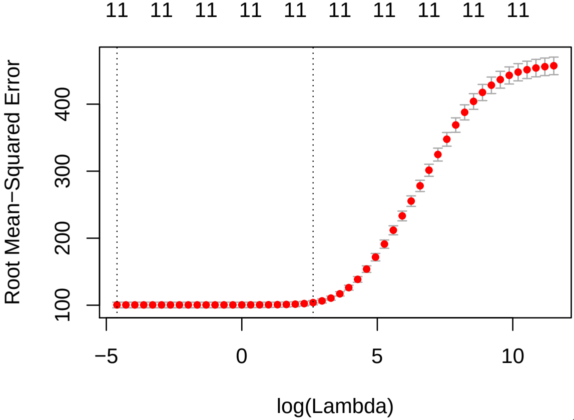
- Goal: predict `Balance` out of other variables

```
R> ## 10-fold cross validation for ridge regression
R> # Create a numeric matrix of dataframe Credit without last column (Balance) and
R> X <- model.matrix(Balance ~ . , data = Credit)  # Predictor variables (as a ma
R> X[, 2:7] <- scale(X[, 2:7])                    # Make columns z-scores of non
R> X <- X[, -1]                                   # Remove intercept column (is
R> result.cv <- cv.glmnet(X, y, alpha = 0,
                         lambda = 10^seq(-2, 5, length.out = 50), nfolds = 10)
R> print(result.cv$lambda.min)      # Best cross validated lambda
[1] 0.01
R> print(result.cv$lambda.1se)      # Conservative est. of best lambda (1 stdev)
[1] 13.9
```

K-Fold Cross Validation

10-fold cross validated RMSE against λ: β_j against λ:

```
R> ## To plot Root Mean Squared Error (RMSE) to be on the same scale as y:
R> result.cv$cvm  <- result.cv$cvm^0.5
R> result.cv$cvup <- result.cv$cvup^0.5
R> result.cv$cvlo <- result.cv$cvlo^0.5
R> plot(result.cv, ylab = "Root Mean-Squared Error")
```



K-Fold Cross Validatione

Final run with best cross validated λ: :

```
R> print(result.cv$lambda.min)      # Best cross validated lambda
[1] 0.01
R> # Final run with best cross validated lambda
R> result <- glmnet(X, y, alpha = 0, lambda = result.cv$lambda.min,
                     intercept = TRUE)
R> result$beta
11 x 1 sparse Matrix of class "dgCMatrix"
                          s0
Income               -275.06
Limit                 472.64
Rating                143.90
Cards                  25.68
Age                   -10.56
Education              -3.60
GenderFemale          -10.65
StudentYes            426.41
MarriedYes             -8.06
EthnicityAsian         16.44
EthnicityCaucasian     10.07
```

K-Fold Cross Validation

Ridge Regression: Example 2

- Ridge regression makes sense if there are many predictor variables ($p \approx n$)
- If you only have a few predictors you can use several tricks:
  - Replace each predictor by its polynomial basis (`poly()`)
  - Model interaction effects
- The benefit is better prediction.
- The cost is that interpretation of the parameters becomes difficult or impossible.

K-Fold Cross Validation2

Ridge Regression: Example 2

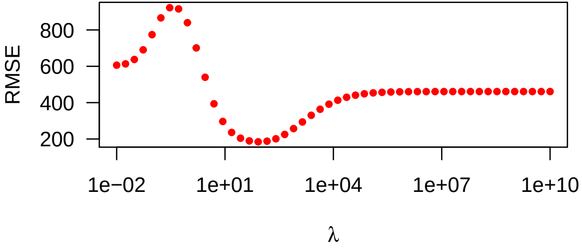
```
R> ## Ridge regression example with p close to n
R> deg <- 5
R> X.poly <- model.matrix(~ 0 + poly(X[, 1], degree = deg)  # Using only the five numerical

                           + poly(X[, 2], degree = deg)      # predictors and generate for ea

                           + poly(X[, 3], degree = deg)      # a fifth degree polynomial basi
                           + poly(X[, 4], degree = deg)      # (that is, 5 columns per variab
                           + poly(X[, 5], degree = deg), data = as.data.frame(X[, 2:7]))
R> X.inter.action <- model.matrix( ~ .^2, data = as.data.frame(X.poly)) # Make interactions
R> dim(X.inter.action)                                   # Size of matrix X.inter.action
[1] 400 326
```

K-Fold Cross Validation

```
R> # 10-fold cross validation for ridge regression
R> result.cv <- cv.glmnet(X.inter.action, y, alpha = 0,
                          lambda = 10^seq(-2, 10, length.out = 50), nfolds = 10)
R> print(result.cv$lambda.min)       # Best cross validated lambda
[1] 82.9
R> print(result.cv$lambda.1se)       # Conservative est. of best lambda (1 stdev)
[1] 146
R> plot(result.cv$lambda, result.cv$cvm^.5, log = "x", col = "red", type = "p", pch = 20,
        xlab = expression(lambda), ylab = "RMSE", las = 1)
```



K-Fold Cross Validation

Ridge regression Analysis Steps:

1. Choose a grid of λ values, e.g., $\{0.01, 0.1, 1, \ldots, 100, 1000\}$.
2. Determine through K-fold cross validation the out-of-sample RMSE for each λ.
3. Find λ_{min} with the smallest out-of-sample residual sum of squares.
4. Run ridge regression on all data using λ_{min} and interpret the β_js.

K-Fold Cross Validation

Leave-one-out cross validation (LOO-CV) $= K$-fold CV with $K = n$

- LOO is good when n is really small (say $n < 100$).
- LOO is used to retain sufficient information in the training data.
- Alternative names: Jack-knife (in Dutch: het jaapmes), Lachenburch's method.

Table of Contents

Thursday Meeting

- One team presents (1) highlights of the methods, (2) results and interpretation (7-10 min).
- One team reflect on methods, one team on results and interpretation (7-10 min each including discussion).
- One team presents code 5-7 min (you can use R-Studio).
  This team sends their code to the two code reflecting teams immediately after Tuesday's lecture.
- Two other teams reflect code (5-7 min each including discussion).
- Reflections should discuss three items:
  - what you think was good;
  - possibly address issues that were unclear to you;
  - suggestions of issues that you think could be improved.

Thursday Meeting

Schedule for Thursday November 5, 2020:

	Team					
Team Task	1	2	3	5	6	7
Presentation methods, results and interpretation	+					
Discussion methods, results and interpretation		+				
Discussion results and interpretation			+			
Presentation code				+		
Discussion code					+	
Discussion code						+

Table of Contents

LASSO Regression

Main properties of LASSO regression (Least Absolute Shrinkage and Selection Operator):

- Adds penalty for nonzero coefficients.
- Penalty is the sum of absolute value of the β_js.
- Effect: for large λ many $\beta_j = 0$.
- Thus, the lasso does variable selection.

LASSO Regression

Loss function lasso regression:

$$L(\boldsymbol{\beta}) = \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\boxed{\text{Regression term}}} + \underbrace{\lambda\|\boldsymbol{\beta}\|_1}_{\boxed{\text{Penalty term}}}$$

with

- $\boldsymbol{\beta}$: unknown $p \times 1$ vector of regression weights
- $\mathbf{X}$: $n \times p$ matrix of predictor variables with elements x_{ij}
- $\mathbf{y}$: $n \times 1$ vector of dependent variable for object $i = 1, \ldots, n$
- λ: positive (given) penalty parameter
- $L_1(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{m} |\beta_j|$ is called the ℓ_1-distance.

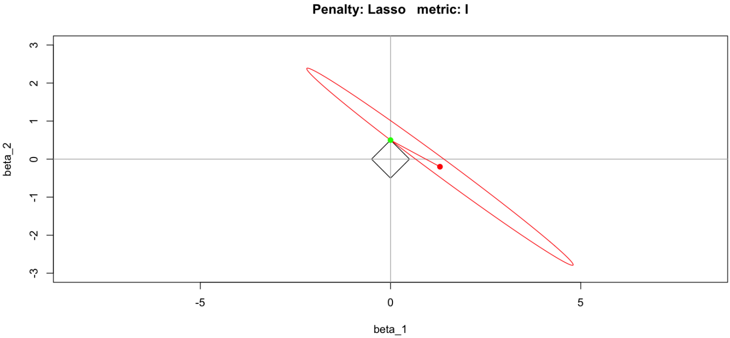
LASSO Regression

As with ridge regression, mathematically the following two definitions of lasso regression are the same:

$$L_{\text{LASSO1}}(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1$$
$$L_{\text{LASSO2}}(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \text{ subject to } \|\boldsymbol{\beta}\|_1 \leq \gamma \text{ for } 0 < \gamma \leq \boldsymbol{\beta}_{\text{OLS}}^\top\boldsymbol{\beta}_{\text{OLS}}$$

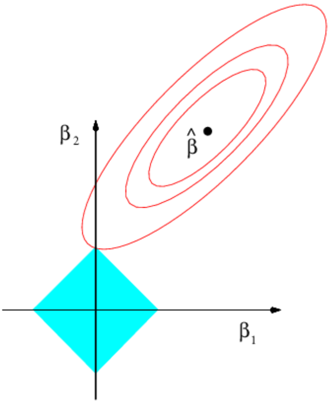
- $\sum_{j=1}^{m} |\beta_j| \leq \gamma$ says that the vector of β_j must be in a diamond shape having ℓ_1-distance $\leq \gamma$.
- For each λ there is a corresponding γ (and vice versa).

LASSO Regression

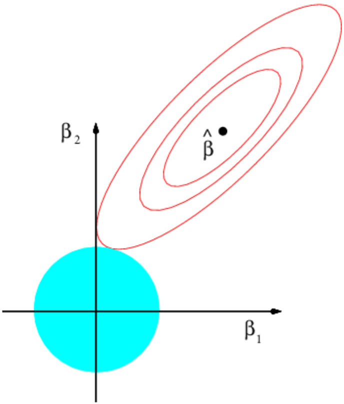


LASSO Regression

LASSO



Ridge



LASSO Regression: Example

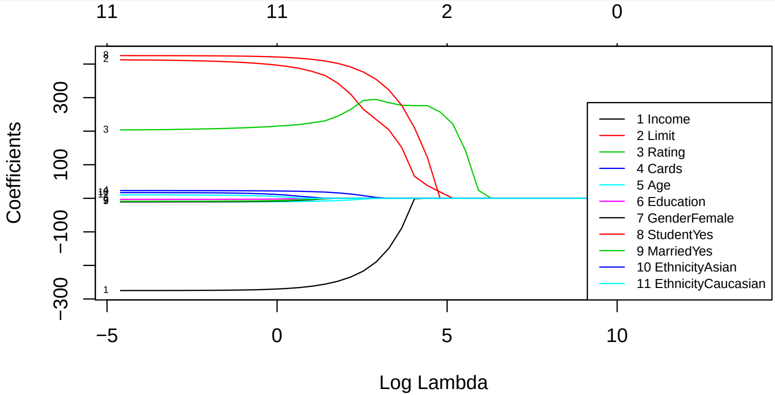
Example LASSO regression: `Credit` data

```
R> ## LASSO regression (alpha must be 1 for LASSO)
R> result <- glmnet(X, y, alpha = 1, lambda = 10^seq(-2, 6, length.out = 50))
```

LASSO Regression: Example

Example LASSO regression profile plot of weights β_j against λ:

```
R> plot(result, xvar = "lambda", label = TRUE)
R> legend("bottomright", lwd = 1, col = 1:6, bg = "white",
          legend = pasteCols(t(cbind(1:ncol(X), " ",colnames(X)))), cex = .7)
```



LASSO Regression: Example

10-fold cross validation for LASSO regression:

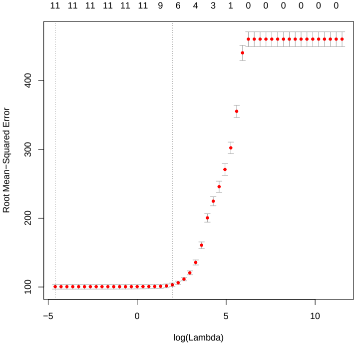
```
R> # 10-fold cross validation for lasso regression
R> result.lasso.cv <- cv.glmnet(X, y, alpha = 1,
                        lambda = 10^seq(-2, 5, length.out = 50), nfolds = 10)
R> print(result.lasso.cv$lambda.min)      # Best cross validated lambda
[1] 0.01
R> print(result.lasso.cv$lambda.1se)      # Conservative estimate of best lambda

[1] 7.2
```

LASSO Regression: Example

10-fold cross validated RMSE against λ: β_j against λ:

```
R> # To plot Root Mean Squared Error (RMSE) to be on the same scale as y:
R> result.lasso.cv$cvm <- result.lasso.cv$cvm^0.5
R> result.lasso.cv$cvup <- result.lasso.cv$cvup^0.5
R> result.lasso.cv$cvlo <- result.lasso.cv$cvlo^0.5
R> plot(result.lasso.cv, ylab = "Root Mean-Squared Error")
```



LASSO Regression: Example

Final run with best cross validated λ:

```
R> # Final run with best cross validated lambda
R> result.lasso.cv$lambda.min
[1] 0.01
R> result.lasso.best <- glmnet(X, y, alpha = 1,
                           lambda = result.lasso.cv$lambda.1se)
R> round(result.lasso.best$beta, digits = 2)
11 x 1 sparse Matrix of class "dgCMatrix"
                         s0
Income              -241.70
Limit                394.46
Rating               187.83
Cards                 17.48
Age                   -5.92
Education              .
GenderFemale           .
StudentYes           398.36
MarriedYes             .
EthnicityAsian         .
EthnicityCaucasian     .
```

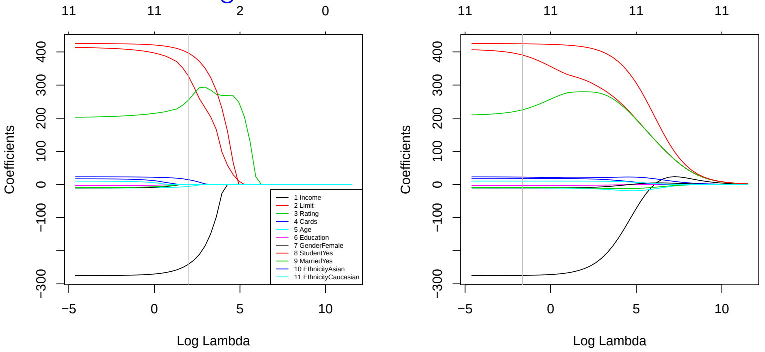
LASSO-Ridge Comparison: Example

Compare LASSO with ridge:

```
R> # Compare Ridge and LASSO results
R> result.ridge.cv <- cv.glmnet(X, y, alpha = 0, nfolds = 10,
                                 lambda = 10^seq(-2, 5, length.out = 50))  # Do R
R> # Final run ridge
R> result.ridge.best <- glmnet(X, y, alpha = 0, lambda = result.ridge.cv$lambda.1
R> round(cbind(result.ridge.best$beta, result.lasso.best$beta), digits = 3)
11 x 2 sparse Matrix of class "dgCMatrix"
                        s0      s0
Income             -236.38 -241.71
Limit               299.40  394.46
Rating              278.20  187.83
Cards                20.44   17.48
Age                 -13.39   -5.92
Education            -2.25    .
GenderFemale         -8.23    .
StudentYes          407.35  398.36
MarriedYes          -11.17    .
EthnicityAsian       16.08    .
EthnicityCaucasian    9.87    .
```

LASSO-Ridge Comparison: Example

Compare LASSO with ridge:



LASSO Regression

Comparison ridge and LASSO:

- Often neither ridge nor LASSO is overall better.
- We expect that the LASSO does well if there are a small number of large nonzero β_j and the others close to zero.
- Ridge works well with many β_js large and of about the same value.
- In practice, we cannot know a priori what the β_js are: use cross validation to find out what fits best.

LASSO Regression

Degrees of freedom

- Degrees of freedom (df) in OLS: number of parameters β_j, thus df$= m$.
- Effective degrees of freedom (df_{eff}) in LASSO regression: the number of nonzero β_j.

LASSO Regression Wrap-up

Important properties of LASSO regression:

- LASSO regression shrinks the β_js towards zero.
- The tuning parameter λ controls strength of shrinkage.
- Automatic variable selection: the ℓ_1 penalty of the LASSO causes some β_j to be zero for large λ.
- As with ridge regression choose λ by k-fold cross validation.
- Estimates are biased but have good mean squared error.
- Big advantage LASSO for interpretation: some β_j are automatically zero.

Table of Contents

Other Penalties

Other penalties (nonexhaustive):

1. Elastic net
2. Smoothed Clipped Absolute Deviation (SCAD)
3. Generalized Double Pareto (GDP)
4. Adaptive LASSO
5. Grouped LASSO

Other Penalties: Elastic Net

Elastic net:

- Critique on LASSO: variable selection can be too dependent on selected data (not stable).
- Elastic net solution: combine ridge and LASSO penalties

Other Penalties: Elastic Net

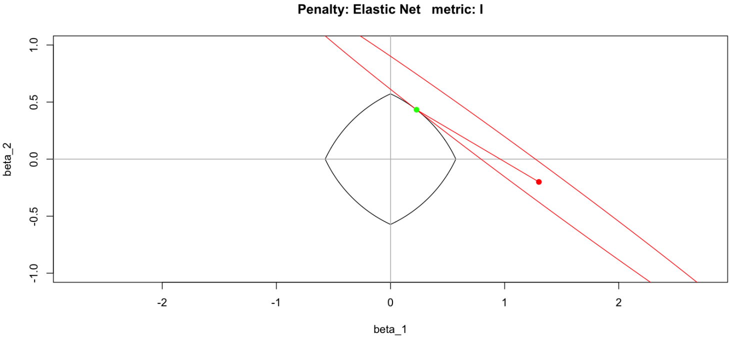
Loss function elastic net regression:

$$L(\boldsymbol{\beta}) \quad = \quad \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\uparrow\ \boxed{\text{Regression term}}} \quad + \quad \underbrace{\lambda\left(\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\boldsymbol{\beta}^\top\boldsymbol{\beta}\right)}_{\uparrow\ \boxed{\text{Penalty term}}}$$

with

- $\boldsymbol{\beta}$: unknown $p \times 1$ vector of regression weights
- $\mathbf{X}$: $n \times p$ matrix of predictor variables with elements x_{ij}
- $\mathbf{y}$: $n \times 1$ vector of dependent variable for object $i = 1, \ldots, n$
- λ: positive (given) penalty parameter
- $0 \leq \alpha \leq 1$: (given) mixing parameter between ridge and LASSO

Other Penalties: Elastic Net



Other Penalties: Elastic Net

Properties elastic net regression:

- More flexibility in value of nonzero β_js.
- Flexibility depends on α:
  - $\alpha = 1$: LASSO
  - $\alpha = 0$: ridge
  - $\alpha = 1/2$: variable selection of LASSO combined with flexibility of ridge
- For large λ more β_js will be zero

Other Penalties

Consider simple regression situation and a general penalty function $P(\beta)$:

$$L(\beta) = (\mathbf{y} - \mathbf{x}\beta)^\top(\mathbf{y} - \mathbf{x}\beta) + \lambda P(\beta)$$

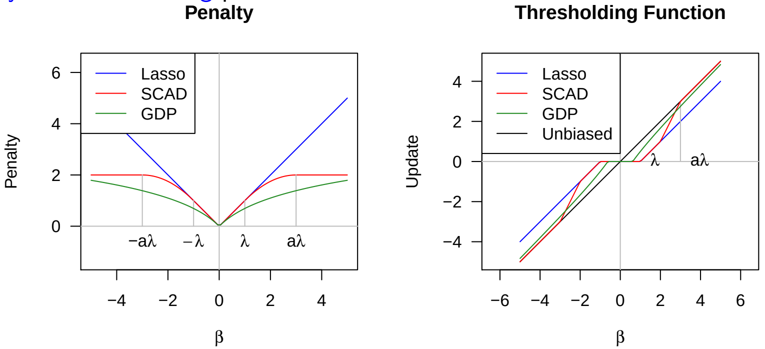
with

$$P(\beta) = \begin{cases} \beta^2 & \text{for ridge regression} \\ |\beta| & \text{for LASSO} \\ \alpha|\beta| + (1-\alpha)\beta^2 & \text{elastic net} \\ \ldots & \text{for } \ldots \end{cases}$$

- Let the OLS solution be $\beta_{\text{OLS}} = \mathbf{y}^\top\mathbf{x}/\mathbf{x}^\top\mathbf{x}$.
- The effect of $\lambda P(\beta)$ on the bias is shown by the thresholding function.

Other Penalties

Penalty and thresholding plots:



Penalties that do feature (variable) selection:

- Lasso: $P(\beta) = |\beta|$
- Smoothed Clipped Absolute Deviation SCAD (Fan & Li, 2001)
- Generalized Double Pareto GDP: $P(\beta) = \log(1 + |\beta|)$

Other Penalties

Properties different penalties with feature selection:

Penalty	Advantage	Disadvantage
Lasso	Convex in $\boldsymbol{\beta}$	Biased $\boldsymbol{\beta}$
SCAD	Nonconvex in $\boldsymbol{\beta}$	Unbiased for large β_j
GDP	Nonconvex in $\boldsymbol{\beta}$	Unbiased for large β_j

Adaptive Lasso

Only a brief discussion of the adaptive Lasso

- Disadvantage Lasso: bias and no guarantee of finding true nonzero population weights.
- The adaptive Lasso solves this problem
- Uses additional fixed weights for each predictor in the penalty.
- Weights can come from OLS regression.

Adaptive Lasso

Loss function adaptive Lasso regression:

$$L(\beta) = \underbrace{(\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta)}_{\boxed{\text{Regression term}}} + \underbrace{\lambda \sum_{j=1}^{m} w_j |\beta_j|}_{\boxed{\text{Penalty term}}}$$

with

- w_j: a (given) weight for predictor variable j.

Adaptive Lasso

Properties and choices adaptive Lasso

- Choose weights w_j as a function of the absolute value of the OLS (standard regression) weights β_j^{OLS}:

$$w_j = \frac{1}{|\beta_j^{\text{OLS}}|}$$

- w_j corrects the penalty term for size of the β_j.
- Under some conditions, the adaptive Lasso has oracle properties: for large enough n, the true nonzero β_j will be found.
- The adaptive Lasso estimates of β_j are also consistent (so that for large n the true population values are found).
- In `glmnet`, the adaptive Lasso is switched on setting `penalty.factor` to the vector fixed weights w_j.

Grouped Lasso

Only a brief discussion of the Grouped Lasso

- The Lasso does variable selection.
- Sometimes we want the β_js of a group of predictors to be either 0 or not zero.
- Example: a categorical predictor (`factor` in R) is represented by a set of dummy variables.
- The Grouped Lasso uses the same idea as the Lasso, but now on the length of the vector of weights.

Grouped Lasso

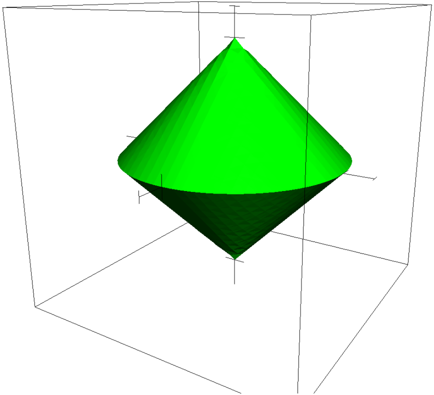
Example of grouped Lasso penalty with β_1 and β_2 forming Group 1 and β_3 equals Group 2

$$\text{grouped Lasso penalty} = \sqrt{\beta_1^2 + \beta_2^2} + |\beta_3|$$

- $\sqrt{\beta_1^2 + \beta_2^2}$ is the length of the vector (β_1, β_2).
- If a group has a single predictor: $\sqrt{\beta_3^2} = |\beta_3|$

Grouped Lasso

Example of the 3D figure corresponding to $\sqrt{\beta_1^2 + \beta_2^2} + |\beta_3| \leq gamma$



Grouped Lasso

Loss function Grouped Lasso regression:

$$L(\beta) \quad = \quad \underbrace{(\mathbf{y}-\mathbf{X}\beta)^\top(\mathbf{y}-\mathbf{X}\beta)}_{\boxed{\text{Regression term}}} \quad + \quad \underbrace{\lambda \sum_{k=1}^{K} \sqrt{\sum_{j \in G_k} \beta_j^2}}_{\boxed{\text{Penalty term}}}$$

with

- G_k the set of indexes j of predictors belonging to group k.
- K is the total number of groups.

Grouped Lasso

Properties and choices grouped Lasso

- Effect of grouped Lasso penalty: per set of predictor variables G_k the corresponding weights are all zero or they are all nonzero.
- The higher λ, the more sets will have zero β_js.
- As the Lasso but for sets of weights β_j.
- Within a set G_k, there is only a shrinkage effect on all β_js of the set (as with ridge).

Table of Contents

An MM Algorithm for the Elastic Net

Finding majorizing algorithm for the elastic net:

- Find new quadratic majorization function

$$g(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} - 2\mathbf{x}^\top \mathbf{b} + c$$

- Set gradient to zero for update:

$$\begin{aligned} \frac{\partial g(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} &= 2\mathbf{A}\mathbf{x} - 2\mathbf{b} = \mathbf{0} \\ \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \hat{\mathbf{x}} &= \mathbf{A}^{-1}\mathbf{b} \end{aligned}$$

An MM Algorithm for the Elastic Net

- Finding a majorizating function for the elastic net:

$$L(\boldsymbol{\beta}) = (2n)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\left((1-\alpha)/2\boldsymbol{\beta}^\top\boldsymbol{\beta} + \alpha\|\boldsymbol{\beta}\|_1\right)$$

- Difficult part lies in $|\beta_j|$. Majorize $|\beta_j|$ by

$$\begin{aligned} 0 &\leq \left(|\beta_j| - \left|\beta_j^{(0)}\right|\right)^2 \\ |\beta_j| &\leq \frac{1}{2}\frac{\beta_j^2}{\left|\beta_j^{(0)}\right|} + \frac{1}{2}\left|\beta_j^{(0)}\right| \\ |\beta_j| &\lesssim \frac{1}{2}\frac{\beta_j^2}{\max\left(\left|\beta_j^{(0)}\right|, \varepsilon\right)} + \frac{1}{2}\left|\beta_j^{(0)}\right| \end{aligned}$$

where

- $\lesssim$ stands for "approximately smaller than"
- ε is a small positive constant (for example, $\varepsilon = 10^{-8}$)

An MM Algorithm for the Elastic Net

- Finding a majorizating function for the elastic net:

$$
\begin{aligned}
L(\boldsymbol{\beta}) &= \frac{\mathbf{y}^\top\mathbf{y} + \boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{y}}{2n} + \frac{\lambda(1-\alpha)}{2}\boldsymbol{\beta}^\top\boldsymbol{\beta} \\
&\quad +\lambda\alpha\sum_{j=1}^{p}|\boldsymbol{\beta}_j| \\
&\leq \frac{\mathbf{y}^\top\mathbf{y} + \boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{y}}{2n} + \frac{\lambda(1-\alpha)}{2}\boldsymbol{\beta}^\top\boldsymbol{\beta} \\
&\quad +\alpha\sum_{j=1}^{p}\left(\frac{1}{2}\frac{\beta_j^2}{\max\left(\left|\beta_j^{(0)}\right|, \varepsilon\right)} + \frac{1}{2}\left|\beta_j^{(0)}\right|\right) \\
&= \frac{1}{2}\boldsymbol{\beta}^\top\left(n^{-1}\mathbf{X}^\top\mathbf{X} + \lambda(1-\alpha)\mathbf{I} + \lambda\alpha\mathbf{D}\right)\boldsymbol{\beta} - n^{-1}\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{y} + c
\end{aligned}
$$

An MM Algorithm for the Elastic Net

- Finding a majorizating function for the elastic net:

$$L(\boldsymbol{\beta}) \quad = \quad \frac{1}{2}\boldsymbol{\beta}^\top \left(n^{-1}\mathbf{X}^\top\mathbf{X} + \lambda(1-\alpha)\mathbf{I} + \lambda\alpha\mathbf{D}\right)\boldsymbol{\beta} - n^{-1}\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{y} + c$$

  with
  - $\mathbf{D}$ a $p \times p$ diagonal matrix with elements $d_{jj} = 1/\max\left(\left|\beta_j^{(0)}\right|, \varepsilon\right)$
  - constant $c = (2n)^{-1}\mathbf{y}^\top\mathbf{y} + (1/2)\lambda\alpha\sum_{j=1}^{p}\left|\boldsymbol{\beta}_j^{(0)}\right|$
- Let $\mathbf{A} = n^{-1}\mathbf{X}^\top\mathbf{X} + \lambda(1-\alpha)\mathbf{I} + \lambda\alpha\mathbf{D}$.
- Then the MM update becomes: $\boldsymbol{\beta}^* = n^{-1}\mathbf{A}^{-1}\mathbf{X}^\top\mathbf{y}$

An MM Algorithm for the Elastic Net

- An MM algorithm for the elastic net:

> Choose with some initial $\boldsymbol{\beta}^{(0)} \in \mathbb{R}^p$
> Compute $L(\boldsymbol{\beta}^{(0)})$
> Set $k \leftarrow 1$
> **while** $k = 1$ or $\left(L(\boldsymbol{\beta}^{(k-1)}) - L(\boldsymbol{\beta}^{(k)})\right) / L(\boldsymbol{\beta}^{(k-1)}) > \epsilon$ **do**
> $k \leftarrow k + 1$
>   Compute $\mathbf{D}$ with elements $d_{jj} = 1/\max\left(\left|\beta_j^{(k-1)}\right|, \varepsilon\right)$
>   Compute $\mathbf{A} = n^{-1}\mathbf{X}^\top\mathbf{X} + \lambda(1-\alpha)\mathbf{I} + \lambda\alpha\mathbf{D}$
>   The update $\boldsymbol{\beta}^{(k)}$ is the solution of the linear system $\mathbf{A}\boldsymbol{\beta} = n^{-1}\mathbf{X}^\top\mathbf{y}$
>   As a check, print $k, L(\boldsymbol{\beta}^{(k)})$, and $L(\boldsymbol{\beta}^{(k-1)}) - L(\boldsymbol{\beta}^{(k)})$
> **end**

Table of Contents

Summary and Assignment

Summary:

Week	Topics	Material
1	Introduction; Introduction to R; Linear methods for regression, model selection, and assessment	3.1, 3.2, 3.3, Xiong (2014)
2	Regularized regression and *k*-fold cross validation	3.4.1-3.4.3, 3.8.4, 7.10
3	Basis function expansions, kernels, bias-variance trade-off	5.1-5.2.1, 5.8, 7.3
4	Support vector machines	Groenen, Nalbantov, Bioch (2009); 12.1-12.3
5	Classification and regression trees, random forests, bootstrap	7.11, 9.2, 15
6	Boosting	10

To Do for Next Time

To Do for Next Time:

- Try to predict `grocery_sum` of the file `supermarket1996.RData` through the elastic net using the demographic variables as predictors.
- Omit the variables `store`, `city`, `ZIP`, `groccoup_sum`, and `shpindx` as predictors.
- Write your own R-function for the elastic net using the MM-algorithm provided in the slides.
- Write your own R-function for determining the hyper parameters (such as λ) through K-fold crossvalidation.
- Provide a comparison of the results of your functions with those of `glmnet()` and explain briefly whether or not they are the same and why this is so.
- Write a small 4-page report about the case according to the template.

To Do for Next Time

Supermarket data:

- The `supermarket1996.RData` data contains yearly turnover, the sum of redeemed grocery coupons, and demographics data of 77 supermarkets in the Chicago area from 1996.
- The demographic data originally comes from U.S. government (1990) census data for the Chicago metropolitan area.
- The table below gives a brief descriptions of the variables in the file.
- The data have been downloaded from `https://www.chicagobooth.edu/research/kilts/datasets/dominicks` in 2014.

To Do for Next Time

Variables supermarket data:

Variable Name	Description
store	Store identification number
city	City of supermarket
Zip	Zip-code
grocery_sum	Total turnover in one year of groceries (in $)
groccoup_sum	Total of redeemed grocery coupos (in $)
age9	% Population under age 9
age60	% Population over age 60
ethnic	% Blacks & Hispanics
educ	% College Graduates
nocar	% With No Vehicles
income	Log of Median Income
incsigma	Std dev of Income Distribution (Approximated)
hsizeavg	Average Household Size

To Do for Next Time

Variables supermarket data:

Variable Name	Description
hsize1	% of households with 1 person
hsize2	% of households with 2 persons
hsize34	% of households with 3 or 4 persons
hsize567	% of households with 5 or more persons
hh3plus	% of households with 3 or more persons
hh4plus	% of households with 4 or more persons
hhsingle	% Detached Houses
hhlarge	% of households with 5 or more persons
workwom	% Working Women with full-time jobs
sinhouse	% of households with 1 person
density	Trading Area in Sq Miles per Capita
hval150	% of Households with Value over $150,000
hval200	% of Households with Value over $200,000
hvalmean	Mean Household Value (Approximated)
single	% of Singles
retired	% of Retired

To Do for Next Time

Variables supermarket data:

Variable Name	Description
unemp	% of Unemployed
wrkch5	% of working women with children under 5
wrkch17	% of working women with children 6 - 17
nwrkch5	% of non-working women with children under 5
nwrkch17	% of non-working women with children 6 - 17
wrkch	% of working women with children
nwrkch	% of non-working women with children
wrkwch	% of working women with children under 5
wrkwnch	% of working women with no children
telephn	% of households with telephones
mortgage	% of households with mortgages
nwhite	% of population that is non-white
poverty	% of population with income under $15,000

To Do for Next Time

Variables supermarket data:

Variable Name	Description
shopcons	% of Constrained Shoppers
shophurr	% of Hurried Shoppers
shopavid	% of Avid Shoppers
shopstr	% of Shopping Stranges
shopunft	% of Unfettered Shoppers
shopbird	% of Shopper Birds
shopindx	Ability to Shop (Car and Single Family House)
shpindx	Ability to Shop (Car and Single Family House)

References I

I. Borg and P. J. F. Groenen. Modern multidimensional scaling. Springer, New York, 2. edition, 2005.

J. De Leeuw. Fitting distances by least squares. Technical Report 130, Interdivisional Program in Statistics, UCLA, Los Angeles, CA, 1993.

Willem J. Heiser. Convergent computation by iterative majorization: Theory and applications in multidimensional data analysis. In W. J. Krzanowski, editor, Recent advances in descriptive multivariate analysis, pages 157–189, Oxford, 1995. Oxford University Press.

D. R. Hunter and K. Lange. A tutorial on MM algorithms. The American Statistician, 39: 30–37, 2004.

Henk A. L. Kiers. Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems. Computational Statistics and Data Analysis, 41:157–170, 2002.

K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. Journal of Computational and Graphical Statistics, 9:1–20, 2000.

H. Voss and U. Eckhardt. Linear convergence of generalized Weiszfeld's method. Computing, 25(3):243–251, 1980.

Alan L Yuille and Anand Rangarajan. The concave-convex procedure. Neural computation, 15 (4):915–936, 2003.

Acknowledgement

Some of the figures in this presentation are taken from “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani