Linear Model Selection and Regularization

Esam Mahdi

Data Analytics (ECMP 5005)
School of Mathematics and Statistics
Master of Engineering - Engineering Practice
Carleton University

October 3, 2023

Learning Objectives

By the end of this chapter, you should be able to do the following:

1. Select the best subset predictors in linear regression models.
2. Perform stepwise regression and best model selection.
3. Differentiate between the forward and backward stepwise techniques.
4. Use regularization technique: Ridge, lasso, and elastic-net regression to shrink regression coefficients and pick the most relevant features.
5. Perform principle component analysis (PCA) to reduce the dimensionality of data.
6. Use R with some real-life applications.

Feature Selection

Recall the linear regression model is given by

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon, \quad \text{where} \quad \varepsilon \underset{\sim}{iid} \mathcal{N}(0, \sigma^2)$$

When there are too many predictor variables then we have two problems:

1. The prediction variance can increase a lot.
2. There might be a significant collinearity among some features (predictors) makes it hard to interpret the regression coefficients.

Select a subset of only relevant features will helps us to reduce the prediction variance and get better interpret and predictions.

Three methods for feature selection

- *Subset selection*: Fit a regression model by least squares on a reduced subset of the p predictors that you believe to be related to the response variable.
- *Shrinkage (Regularization)*: Fit a model involving all p predictors, but some parameter estimates are shrunken towards zero by applying some penalty or constraint.
- *Dimension reduction*: Compute $M < p$ different linear combinations, or projections, of the variables and use these M projections to fit a least squares model.

Best Subset and Stepwise Model Selection

Consider the p independent variables $x_1, x_2, \cdots, x_p$ and a response y. The *best subset selection* method in a regression model :

1. Start with the *null model*, which contains no predictors. This model has the intercept coefficient only. It simply predicts the sample mean of the response variable. Denote this model by $\mathcal{M}_0$
2. For $k = 1, 2, \cdots, p$:
   1. Fit all $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ models that contain exactly k predictors.
   2. Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest adjusted R^2.
3. Select a single best model from among $\mathcal{M}_0, \cdots, \mathcal{M}_k$ using cross-validated prediction error, C_p, AIC, AICc, BIC, or adjusted R^2.

Note that this method is a computational expensive as you need to fit a large number of models, $\sum_{k=0}^{p} \binom{p}{k}$. For example, if $p = 10$, then you need to fit 1024 different models.

BEST SUBSET SELECTION IN R

Consider the Hitters data set in the **ISLR2** library. In this example, we use the `regsubsets()` function, which is a part of the package **leaps** to perform a best subset selection of relevant predictors that can be used to predict a baseball player's `Salary`. The `best` model is quantified using RSS.

- The syntax of the `regsubsets()` function is the same as for `lm()`.
- The `summary()` command outputs the best set of variables for each model size.
- The `summary()` returns R^2, RSS, adjusted R^2, C_p, and BIC (that we will explain later!).
- By default, `regsubsets()` only reports results up to the best 8 variable model.
  - The `nvmax()` option can be used in order to return as many variables as are desired.
  - The asterisk in the summary output indicates that a given variable is included in the corresponding model.
- Plotting R^2, RSS, adjusted R^2, C_p, and BIC for all of the models at once will help us decide which model to select.

The R code is given in the next slide, where we select the six-variable model with the lowest BIC:

$$\widehat{\text{Salary}} = 91.51 - 1.87\ \text{AtBat} + 7.60\ \text{Hits} + 3.70\ \text{Walks} + 0.64\ \text{CRBI} - 122.95\ \text{DivisionW} + 0.26\ \text{PutOuts}$$

BEST SUBSET SELECTION IN R

```r
library(ISLR2); library (leaps)
head(Hitters)
anyNA(Hitters) #check if the data has any missing observations
# Count the number of missing values in Salary variable (59 "NA" observation)
sum(is.na(Hitters$Salary))
Hitters <- na.omit(Hitters) #remove the missing data
# Use nvmax() = 19 to return 19 models (default nvmax() = 8)
regfit.full <- regsubsets(Salary ~., data = Hitters, nvmax = 19)
reg.summary <- summary(regfit.full)
reg.summary
reg.summary$adjr2 # return adjusted R-squared for all models
which.max(reg.summary$adjr2) #identify which model has largest adj-R2
which.min(reg.summary$bic) #identify which model provides smallest BIC
# Plot R2, RSS, adjusted R2, Cp, and BIC for all of the models at once
par(mfrow = c(2, 2))
plot(regfit.full, scale = "r2")
plot(regfit.full, scale = "adjr2")
plot(regfit.full, scale = "Cp")
plot(regfit.full, scale = "bic")
# See the coefficient estimates associated with model 6
coef(regfit.full, 6)
```

STEPWISE REGRESSION

Stepwise Regression is the step-by-step iterative construction of a regression model that involves the selection of independent variables to be used in a final model. It involves *adding* or *removing* potential explanatory variables in succession and testing for statistical significance after each iteration (see `https://www.investopedia.com`).

Three main approaches for stepwise regression:

1. *Forward selection*.
2. *Backward elimination*.
3. *Bidirectional stepwise*: A combination of forward and backward stepwise selection, testing at each step for independent variables to be included or excluded.

Forward Stepwise Selection

Consider the p independent variables $x_1, x_2, \cdots, x_p$ and a response y. The *forward stepwise selection* is a step-by-step iterative method for *adding* variables in a regression model:

1. Starts with no predictor variables in the model (*null model*)

$$y = \beta_0 + \varepsilon, \quad \text{thus} \quad \hat{y} = \hat{\beta}_0,$$

where $\hat{\beta}_0 = \bar{y}$ is the estimated point of β_0 in this case.

2. Add the *most significant* independent variable to this model (say x_3).
This can be achieved by implementing any of the following criteria:
   - Calculate the p-value for testing the hypotheses $H_0 : \beta_j = 0$ against $H_A : \beta_j \neq 0$, for $j = 1, 2, \cdots, p$ and select the *smallest p-value*.
   - It provides the *highest increase* in adjusted coefficient of determination R^2_{adj}.
   - It provides the *highest drop* in *predicted residual error sum of squares (PRESS)* statistic compared to other predictors under consideration.

   Now the new regression model after adding x_3 is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 \quad (x_3 \text{ is added to the null model})$$

3. Repeat step 2 for the remaining variables until the pre-specified stopping rule (will be explain later) is reached or until all the independent variables are included in the model.

BACKWARD STEPWISE ELIMINATION

Consider the p independent variables $x_1, x_2, \cdots, x_p$ and a dependent y. The *backward stepwise elimination* is a step-by-step iterative method for *deleting* variables from the model:

1. Starts with all predictor variables in the model (*full model*)

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon, \quad \text{thus} \quad \hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p,$$

   where $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_p$ are the least squares (or MLE) estimates of $\beta_0, \beta_1, \cdots, \beta_p$.

2. Delete the *least significant* independent variable to this model (say x_3).
   This can be achieved by implementing any of the following criteria:
   - Calculate the p-value for testing the hypotheses $H_0 : \beta_j = 0$ against $H_A : \beta_j \neq 0$, for $j = 1, 2, \cdots, p$ and select the *highest p-value* (to be deleted).
   - It provides the *lowest drop* in adjusted coefficient of determination R^2_{adj}.
   - It provides the *lowest increase* in *predicted residual error sum of squares (PRESS)* statistic compared to other predictors under consideration.

   Now the new regression model after deleting x_3 is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_4 x_4 + \cdots + \hat{\beta}_p x_p \quad (x_3 \text{ is removed from the model})$$

3. Repeat step 2 for the remaining variables until the pre-specified stopping rule (explained in the next slides) is reached or until no variable is left in the model.

The final fitted regression model is the model that satisfies:

- All included independent variables are significant (at pre-specified α level, say 5%) when we are testing the hypotheses $H_0 : \beta_j = 0$ versus $H_A : \beta_j \neq 0$.
- This model has the lowest *Information Criterion (IC)* (see next slide).

Thus, in stepwise regression, we stop *adding/removing* variables to our final model if *adding/removing* will:

- Provide p-values that are larger than α, which leads to not reject H_0;
- increase the values of IC;
- the adjusted coefficient of determination, R^2_{adj}, value starts to decrease or not change.

The *Information Criterion (IC)* is a measure of goodness of fit that can help us to select the best parsimonious fitted model (i.e., the model with the least number of significant predictors).
Th popular IC are:

- *Mallow's C_p*.
- *Akaike's Information Criterion (AIC)*.
- *Corrected AIC (AICc)*.
- *Bayesian Information Criterion (BIC)*.

The smaller the value of IC the preferred the model.

Mallow's C_p

$$C_p = \frac{1}{n}(\text{RSS} + 2k\hat{\sigma}^2),$$

where

- $k = p + 1$ is the number of model parameters $(\beta_0, \beta_1, \cdots, \beta_p)$.
- RSS is the residual sum of squares.
- $\hat{\sigma}^2 = \text{MSE} = \text{RSS}/(n - p - 1)$ is the estimate of $\text{Var}(\varepsilon)$.

Akaike's Information Criterion (AIC)

$$\text{AIC} = 2k - 2\log \hat{L}$$

where k denotes the number of parameters and $\hat{L}$ is the maximized value of the *log likelihood function* for the estimated model.

Note: In the case of the linear model with Gaussian errors, maximum likelihood and least squares are the same thing, and C_p and AIC are equivalent.

Corrected AIC (AICc)

A bias corrected version of AIC for use when the sample size is small, or when the number of parameters estimated is a moderate to large fraction of the sample size is the *Corrected AIC (AICc)* which is defined by

$$\text{AICc} = \text{AIC} + \frac{2k^2 + 2k}{n - k - 1}$$

where n denotes the sample size and k denotes the number of parameters.

Bayesian Information Criterion (BIC)

$$\text{BIC} = k \log n - 2 \log \hat{L} = \frac{1}{n}(\text{RSS} + k\hat{\sigma}^2 \log n)$$

Note:

- For model selections, there is no clear choice between AIC, C_p, and BIC.
- In general, BIC places more penalty on models with many variables. Thus, it yields to select smaller models than AIC and C_p.
- BIC is asymptotically consistent as a selection criterion. Thus, several statistician prefer to use BIC and not AIC or C_p.

- Forward stepwise selection is recommended to be used when the number of predictors under consideration is, p, very large compared with the sample size n.
- Backward stepwise elimination is recommended to be used when we suspect of multicollinearity, but can not be used when $p \geq n$.

Note: Always use the backward stepwise approach, unless the number of variables is more than the sample size, where the forward stepwise is better in this case.

The main issues with stepwise selection are:

- It does not consider all possible combination of potential predictors.
- The output of the regression coefficients, confidence intervals, p-values, and R^2, etc. might be biased.
- It produces an unstable selection of variables, especially when we have a small sample size compared to the number of variables we want to study.
- It does not consider the causal relationship between variables.

There are many suggestions to deal with the limitations of stepwise approach:

- Split the data into training and testing sets (say K-Fold CV). Use the training data to fit the model and use the testing data to validate this model.
- Use bootstrap method to check the stability of the selection.
- Consider the shrinkage methods such as LASSO regression.

Forward and backward stepwise selection in R

Consider again the Hitters data set that we studied before. We can also use the `regsubsets()` function to perform forward stepwise or backward stepwise selection, using the argument `method = "forward"` or `method = "backward"`.

```
regfit.fwd <- regsubsets(Salary ~., data = Hitters,
nvmax = 19, method = "forward")
summary(regfit.fwd)
regfit.bwd <- regsubsets(Salary ~., data = Hitters,
nvmax = 19, method = "backward")
summary(regfit.bwd)
```

For instance, we see that using forward stepwise selection, the best one variable model contains only CRBI, and the best two-variable model additionally includes Hits. For this data, the best one-variable through six-variable models are each identical for best subset and forward selection. However, the best seven-variable models identified by forward stepwise selection, backward stepwise selection, and best subset selection are different. See next slide!

BEST SEVEN-VARIABLE MODELS IDENTIFIED BY FORWARD STEPWISE SELECTION, BACKWARD STEPWISE SELECTION, AND BEST SUBSET SELECTION

```
round(coef(regfit.full, 7), 3)
round(coef(regfit.fwd , 7), 3)
round(coef(regfit.bwd , 7), 3)
```

Coefficient	Best subset selection	Forward stepwise selection	Backward stepwise selection
Intercept	79.451	109.787	105.649
AtBat	–	−1.959	−1.976
Hits	1.283	7.450	6.757
Walks	3.227	4.913	6.056
CRBI	–	0.854	–
CAtBat	−0.375	–	–
CHits	1.496	–	–
CHmRun	1.442	–	–
CRuns	–	–	1.129
CWalks	–	−0.305	−0.716
DivisionW	−129.987	−127.122	−116.169
PutOuts	0.237	0.253	0.303

Shrinkage Methods

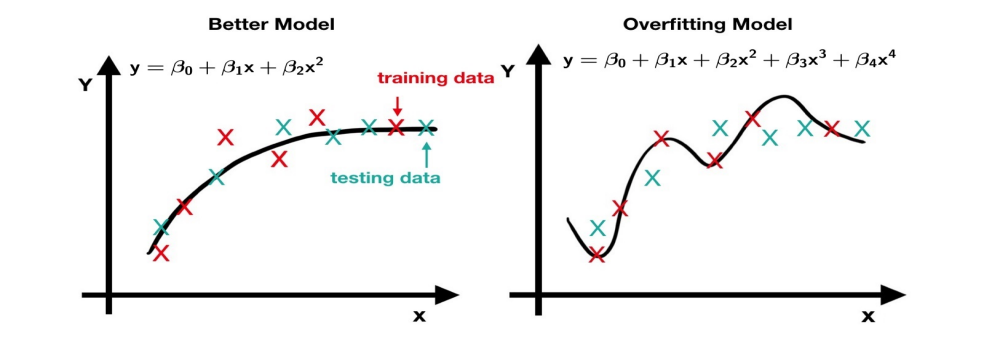

FIGURE: Over-fitting model and prediction accuracy (Bias-variance tradeoff).

Right: The model is overfit the training data. This model has poor performance with low bias (which is good) but high variance (which is bad) on the test data. Left: The model has a better performance on the test data. It decreases the variance on the cost of increasing the bias (*Bias-variance tradeoff*).

Recall that the multiple linear regression model can be written in the matrix notation $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ and the estimated model is

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{ols}},$$

where

$$\hat{\boldsymbol{\beta}}_{\text{ols}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}.$$

The coefficients $\hat{\boldsymbol{\beta}}_{\text{ols}}$ are obtained by minimizing the *loss function*

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k} x_{ij}\beta_j)^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{1}$$

- Ridge, lasso, and elastic-net are used to *shrink* the least important regression coefficients (predictors) by imposing a *penalty* on their size.
- Thus, regression coefficients will be estimated by minimizing the loss function in (1) imposing some *constraint* on these coefficients.
- Adding this penalty will *reduce the variance* of parameter estimates, but it comes at the expense of some additional *bias*.
- Ridge uses *L2 regularization* technique by adding L2 penalty which is equal to the *square* of the magnitude of coefficients.
- LASSO uses *L1 regularization* technique by adding L1 penalty that is equal to the *absolute value* of the magnitude of coefficient
- Elastic-Net combines the *L1 and L2 regularization together*.
- Ridge, lasso, and elastic-net regression can be applied to *generalized linear models (glm)* including logistic regression models.
- It is best to apply ridge, lasso, and elastic net regression after *standardizing the predictors (scale)*, using the formula

$$\tilde{x}_{ij} = \frac{x_{ij}}{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2},$$

where $\bar{x}_j = 1/n \sum_{i=1}^{n} x_{ij}$ is the mean of the jth predictor X_j, $j = 1, 2, \cdots, p$.

The estimated coefficients can be obtained by minimizing the *Lagrangian form*

$$\underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k} x_{ij}\beta_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=1}^{k} \beta_j^2}_{\text{Penalty term}} = \underbrace{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})}_{\text{Loss function}} + \underbrace{\lambda \boldsymbol{\beta}'\boldsymbol{\beta}}_{\text{Penalty}}, \tag{2}$$

where $0 \leq \lambda < \infty$ is the *tuning parameter*. We choose a grid of λ values and the value for which the *cross-validation* error is smallest.
The ridge regression solution of (2) is

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \mathbb{I})^{-1}\boldsymbol{X}'\boldsymbol{y}.$$

where $\mathbb{I}$ is $k \times k$ identity matrix.

- The penalty term adds little bit of *bias* to the parameter estimates to *reduce the variance* of the predicted values (improve prediction accuracy).
- The intercept β_0 has been left out of the penalty term, where we estimated it by $\hat{\beta}_0 = \bar{y}$.
- When $\lambda = 0$ (no regularization), we have $\hat{\boldsymbol{\beta}}_{\text{ridge}} = \hat{\boldsymbol{\beta}}_{\text{ols}}$.
- When λ gets large asymptotically to infinity, the $\hat{\boldsymbol{\beta}}_{\text{ridge}}$ *shrinks asymptotically to zero (but never be exactly zero)*. Thus, ridge regression does not perform variable selection!

The estimated coefficients can be obtained by minimizing

$$\underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k} x_{ij}\beta_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=1}^{k} |\beta_j|}_{\text{Penalty term}} , \tag{3}$$

where $0 \leq \lambda < \infty$ is the *tuning parameter* that can be chosen by *cross-validation* methods, same as in ridge regression. The lasso regression solution of (3) can be obtained using a *quadratic programming technique*.

- The penalty term adds little bit of *bias* to the parameter estimates to *reduce the variance* of the predicted values (improve prediction accuracy).
- When $\lambda = 0$ (no regularization), we have $\hat{\beta}_{\text{lasso}} = \hat{\beta}_{\text{ols}}$.
- Making λ sufficiently large can cause some of the coefficients (least important predictors) to be *exactly zero*. Thus, LASSO is a method for feature selection.
- Thus, lasso regression effectively selects the most important predictors for predicting outcome (*better than stepwise regression*).

The estimated coefficients can be obtained by minimizing

$$\underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{k} x_{ij}\beta_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=1}^{k}((1-\alpha)\beta_j^2 + \alpha|\beta_j|)}_{\text{Penalty mixed term}}, \tag{4}$$

where $0 \leq \lambda < \infty$ is the *tuning parameter* and $0 \leq \alpha \leq 1$ is the *elastic-net mixing parameter* which determines the mix of the penalties for considering both ridge and lasso.

- The penalty term adds little bit of *bias* to the parameter estimates to *reduce the variance* of the predicted values (improve prediction accuracy).
- λ and α can be chosen by *cross-validation* methods.
- When $\alpha = 0$, we have ridge regression, i.e, $\hat{\boldsymbol{\beta}}_{\text{elastic-net}} = \hat{\boldsymbol{\beta}}_{\text{ridge}}$.
- When $\alpha = 1$, we have lasso regression, i.e, $\hat{\boldsymbol{\beta}}_{\text{elastic-net}} = \hat{\boldsymbol{\beta}}_{\text{lasso}}$.
- When $\lambda = 0$ (no regularization), we have $\hat{\boldsymbol{\beta}}_{\text{elastic-net}} = \hat{\boldsymbol{\beta}}_{\text{ols}}$.
- λ is chosen by cross-validation.

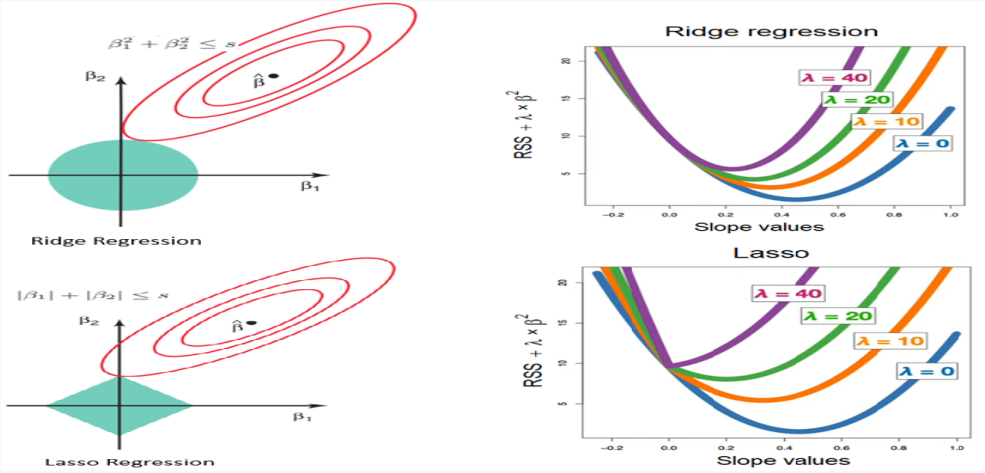

FIGURE: Lasso Regression can shrink parameter values towards 0, but Ridge Regression can not.

THE GLMNET() FUNCTION IN THE PACKAGE **glmnet**

- The `glmnet()` function in the package **glmnet** can be used to fit the ridge, lasso, elastic-net models, and more.
- More information about the R package **glmnet** can be found *here*.
- The syntax of this function is slightly different from other model -fitting functions as we must pass in an x matrix as well as a y vector, and we do not use the y ∼ x syntax.
- The `glmnet()` function has an alpha argument that determines what type of model is fit. If alpha = 0 then a ridge regression model is fit, and if alpha = 1 then a lasso model is fit.
- By default the argument lambda in the `glmnet()` is used for an automatically selected range of λ values. However, you can implement the function over a grid of values.
- By default, the `glmnet()` function standardizes the variables so that they are on the same scale, which is always recommended. To turn off this default setting, use the argument standardize = FALSE.

See Section 6.5 Lab: Linear Models and Regularization Methods from the textbook.

THE TRAIN() FUNCTION IN THE PACKAGE **caret**

See the R Markdown example file uploaded in the Brightspace of this course.

The idea in *principal component analysis (PCA)* is to combine p predictor variables into a smaller set of variables (M), which are weighted linear combinations of the original set.

- The set of M variables (denotes *principal components*) explains most of the variability of the full set of variables.
- The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

KEY TERMS FOR PRINCIPAL COMPONENTS ANALYSIS

- *Principal component*: A linear combination that transforms a large number of correlated variables into a smaller number of *uncorrelated variables* called principal components.
- *Loadings*: The weights (range from -1 and 1) for each original variable that transform the predictors into the components.
- *Screeplot*: A plot shows the number of principle components on the $X-$axis against the proportion of the variance explained by these components on the $Y-$axis.

Principal Components for Two Variables

For two predictor variables, X_1 and X_2, there are two principal components Z_1 and Z_2 that are defined as a linear combination of the original predictor variables:

$$Z_i = \phi_{i1}X_1 + \phi_{i2}X_2, \quad i = 1, 2$$

The coefficients (weights) ϕ_{i1}, ϕ_{i2} are known as the *loadings*. The PCA algorithm chooses ϕs to capture as much variance of the data as possible. The first principal component, Z_1, captures the largest variance of the data. The second principal component, Z_2, captures the remaining variation.

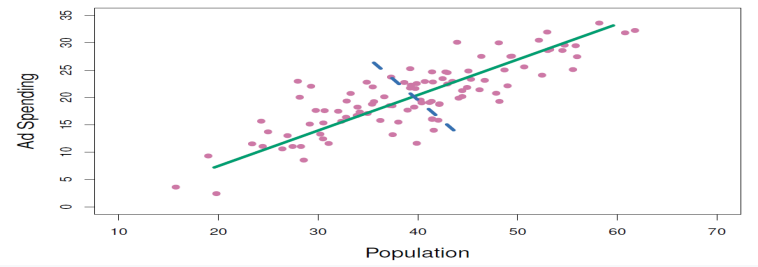

FIGURE: The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

Principal components in R: Hitters baseball dataset

Consider the `Hitters` baseball dataset from the **ISLR2** package, which contains various information about 263 professional players on 20 variables. In this example, we perform a PCA on the two predictors `Hits` (number of hits in 1986) and `Runs` (number of runs in 1986).

```
library (ISLR2)
df <- Hitters[,c("Hits","Runs")]
## PCA using princomp() function
pca <- princomp(scale(df))
pca$loadings

##
## Loadings:
##      Comp.1 Comp.2
## Hits  0.707  0.707
## Runs  0.707 -0.707
##
##                Comp.1 Comp.2
## SS loadings       1.0    1.0
## Proportion Var    0.5    0.5
## Cumulative Var    0.5    1.0
```

```
## PCA using prcomp() function
pca <- prcomp(df, scale = TRUE)
pca

## Standard deviations (1, .., p=2):
## [1] 1.3864297 0.2789495
##
## Rotation (n x k) = (2 x 2):
##             PC1        PC2
## Hits 0.7071068 -0.7071068
## Runs 0.7071068  0.7071068
```

The weights for `Hits` and `Runs` for the first principal component are 0.707 and 0.707 and for the second principal component they are 0.707 and -0.707.

PRINCIPAL COMPONENTS IN R: HITTERS BASEBALL DATASET

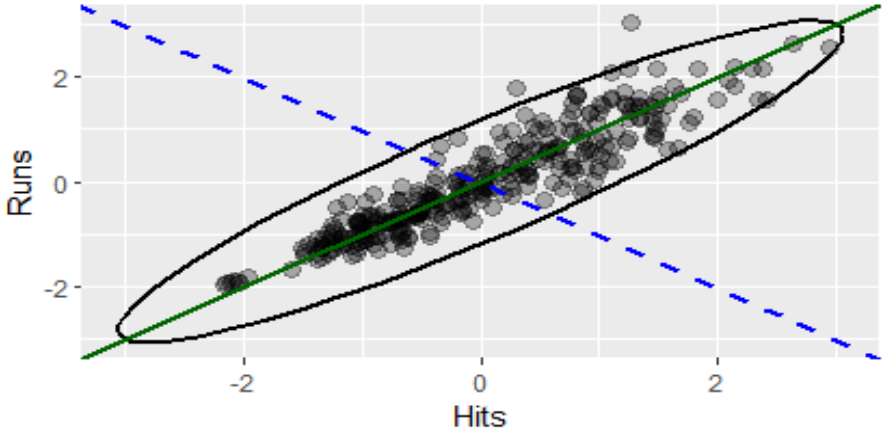

FIGURE: The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

For a data with p predictor variables, $X_1, X_2, \cdots, X_p$, the M principle components, $Z_1, Z_2, \cdots, Z_M$, where $M \ll p$, are defined as linear combinations of the original p predictors as follows:

$$Z_m = \phi_{1m}X_1 + \phi_{2m}X_2 + \cdots + \phi_{pm}X_p = \sum_{j=1}^{p} \phi_{jm}X_j.$$

We can then use the least squares method to fit the linear regression model

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \varepsilon_i, \quad i = 1, \cdots, n \tag{5}$$

Notice that from (5),

$$\begin{aligned}\sum_{m=1}^{M} \theta_m z_{im} &= \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{jm}x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{jm}x_{ij} \\ &= \sum_{j=1}^{p} \beta_j x_{ij}, \quad \text{where} \quad \beta_j = \sum_{m=1}^{M} \theta_m \phi_{jm}\end{aligned}$$

Hence, (5) can be seen as a special case of the original linear regression model and dimension reduction serves to constrain the estimated β_j coefficients.

Calculate the Principal Components

- Let $\boldsymbol{X}$ denotes the data set of n rows and p features: $\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$.
- Usually, we *standardized the predictors* (column means and variances of $\boldsymbol{X}$ are 0 and 1, respectively):
  - Mathematically, we use the formula $\dfrac{x_{ji} - \bar{x}_i}{\sqrt{S_{ii}}}$, where

$$S_{ii} = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_{ji} - \bar{x}_i)^2}, \quad \bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ji}, \quad i = 1, \cdots, p.$$

  - In R, we use the syntax: `scale(x, center = TRUE, scale = TRUE)`.
- Calculate the matrix $\tilde{\boldsymbol{X}}'\tilde{\boldsymbol{X}}$, where $\tilde{\boldsymbol{X}}$ denotes the data set with standardized predictors.
- We replace the p features with $M \ll p$, random variables called principal components that are linear combinations of the original variables such that:
  - The M principal components are uncorrelated;
  - the first PC accounts for as much of the variability in the data as possible;
  - each succeeding PC accounts for as much of the remaining variability as possible.

Calculate the Principal Components (Cont.)

- Calculate the *eigenvalue-eigenvector* pairs $(\lambda_1, \boldsymbol{\phi}_1), (\lambda_2, \boldsymbol{\phi}_2), \ldots, (\lambda_p, \boldsymbol{\phi}_p)$ of the matrix $\tilde{\boldsymbol{X}}'\tilde{\boldsymbol{X}}$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ and $\boldsymbol{\phi}_i = (\phi_{1i}, \phi_{pi}, \cdots, \phi_{pi})'$ denotes the i^{th} principal component *loading vector*.
- The ith principal component is given by

$$Z_i = \boldsymbol{\phi}_i' \boldsymbol{X} = \phi_{1i} X_1 + \phi_{2i} X_2 + \cdots + \phi_{pi} X_p, \quad i = 1, 2, \ldots, p. \tag{6}$$

- Given a $n \times p$ data set and (6), we define the *scores* of the first principle component as

$$z_{i1} = \phi_{1i} x_{i1} + \phi_{2i} x_{i2} + \cdots + \phi_{pi} x_{ip}.$$

- We choose $\boldsymbol{\phi}_1$ so that $\text{Var}(Z_1) = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 = \lambda_1$ is maximized subject to the constraint that $\boldsymbol{\phi}_1' \boldsymbol{\phi}_1 = \sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- The second PC corresponds to the choice of $\boldsymbol{\phi}_2$ so that $\text{Var}(Z_2) = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j2} x_{ij} \right)^2 = \lambda_2$ is maximized subject to $\boldsymbol{\phi}_2' \boldsymbol{\phi}_2 = \sum_{j=1}^{p} \phi_{j2}^2 = 1$ and $\text{Cov}(Z_2, Z_1) = 0$.
- The i^{th} PC corresponds to the choice of $\boldsymbol{\phi}_i$ so that $\text{Var}(Z_i) = \lambda_i$ is maximized subject to $\boldsymbol{\phi}_i' \boldsymbol{\phi}_i = 1$ and $\text{Cov}(Z_i, Z_j) = 0$ for $j = 1, \ldots, i-1$.
- We refer to $z_{1i}, z_{2i}, \cdots, z_{ni}$ as the *scores* of the i^{th} principal component.

How Many Components to Choose?

In order to reduce the dimension of the data, you must decide how many principal components to select? Below some methods that you can use:

- Method 1 (ad hoc rule): by plotting the *screeplot*. A scree plot shows how much variation each PC captures from the data. An ideal curve should be steep, then bends at an *"elbow"* −− this is the numbers of components you need to select −− and after that flattens out.

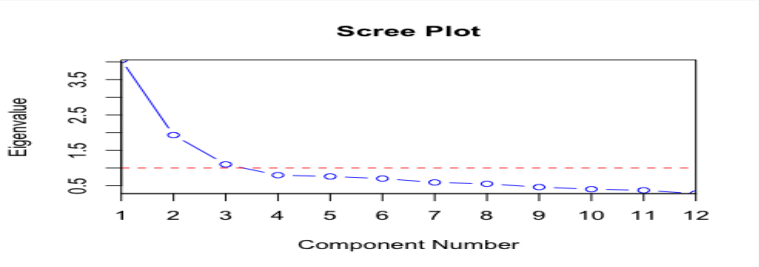

FIGURE: Source: Wikipedia. Just PC $1-3$ might be enough to describe the data.

- Method 2 (ad hoc rule): Select the top components such that the cumulative variance exceeds a given threshold (say 85%).
- Method 3 (formal rule): Use cross-validation methods.

HITTERS BASEBALL DATASET

```
df <- as.data.frame(scale(Hitters[,c(1:13,16:18)]))
pca <- princomp(df)
par(mfrow=c(1,2))
screeplot(pca);  screeplot(pca, type = "lines")
```

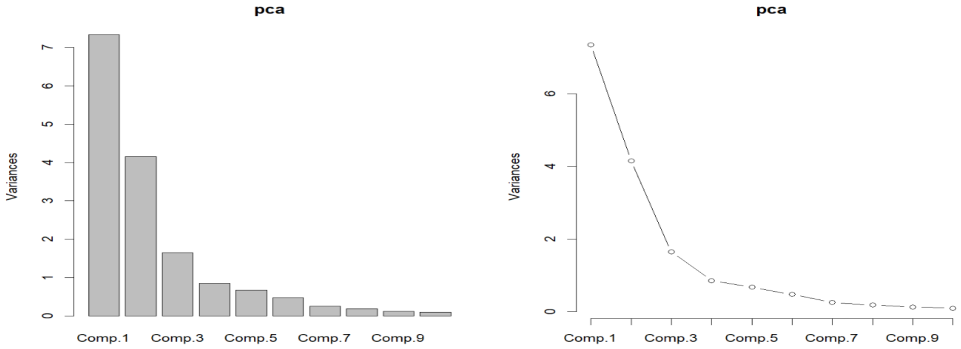

FIGURE: A screeplot for a PCA of hitters baseball data. Just PC $1-6$ might be enough to describe the variability in the data.

PRINCIPAL COMPONENTS REGRESSION (PCR) IN R: HITTERS BASEBALL DATASET

Principal components regression (PCR) can be performed using the `pcr()` function, which is part of the **pls** package.

- The syntax for the `pcr()` function is similar to that for `lm()` with some additional parameters.
- The default argument `center = TRUE` is used to perform mean centering and setting the argument `scale = TRUE` will standardize predictor variables.
- Setting the argument `validation = "CV"` tells R to calculate 10-fold cross-validation error for each possible value of the number of principal components used. Also note that you can specify `validation = "LOOCV"` instead to perform *leave-one-out cross-validation*.
- To determine the number of principal components worth keeping, you can examine the resulting fit using the syntax `summary()`.

We now apply the PCR to the `Hitters` data, in order to predict the `Salary` (response variable). See the R Markdown example file uploaded in the Brightspace of this course!

You need to

- Read and understand the examples and code in Section 6.5 Lab: Linear Models and Regularization Methods from the textbook *"An Introduction to Statistical Learning: With Applications in R"*
- Solve question 5 in Exercise 6.6.
- Solve questions 8 to 11 in Exercise 6.6.