

Business 4720 - Class 12

Supervised Machine Learning using R

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
`jevermann@mun.ca`



Unless otherwise indicated, the copyright in this material is owned by Joerg Evermann. This material is licensed to you under the [Creative Commons by-attribution non-commercial license \(CC BY-NC 4.0\)](https://creativecommons.org/licenses/by-nc/4.0/)

What You Will Learn:

- ▶ Linear Regression Models in R
 - ▶ Linear regression
 - ▶ Lasso and Ridge regression
- ▶ Classification Models in R
 - ▶ Logistic Regression
 - ▶ K-NN

Based On

Gareth James, Daniel Witten, Trevor Hastie and Robert Tibshirani: *An Introduction to Statistical Learning with Applications in R*. 2nd edition, corrected printing, June 2023. (ISLR2)

<https://www.statlearning.com>

Chapters 2, 3, 4, 5

Trevor Hastie, Robert Tibshirani, and Jerome Friedman: *The Elements of Statistical Learning*. 2nd edition, 12th corrected printing, 2017. (ESL)

<https://hastie.su.domains/ElemStatLearn/>

Chapters 2, 3, 4, 7

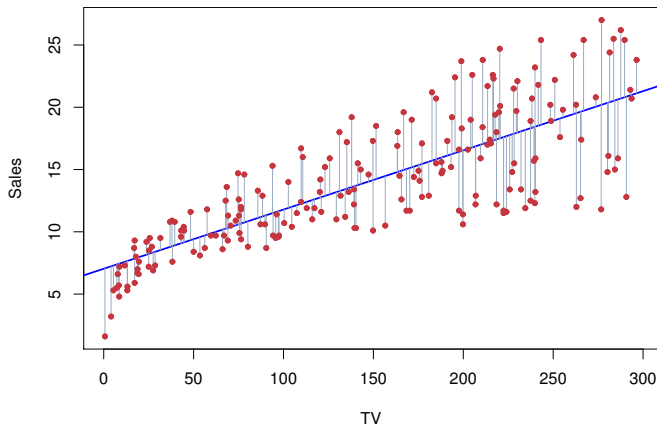
Kevin P. Murphy: *Probabilistic Machine Learning – An Introduction*. MIT Press 2022.

<https://probml.github.io/pml-book/book1.html>

Chapters 4, 6, 9, 10, 11

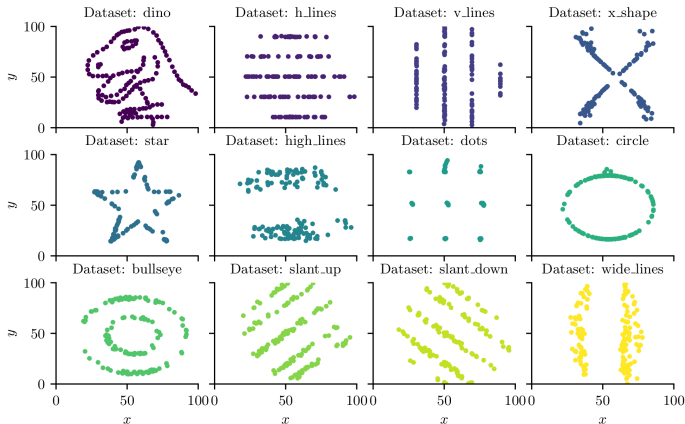
Linear Regression

$$Y = \beta_0 + \beta_1 X + \epsilon$$



Source: ISLR2 Figure 3.1

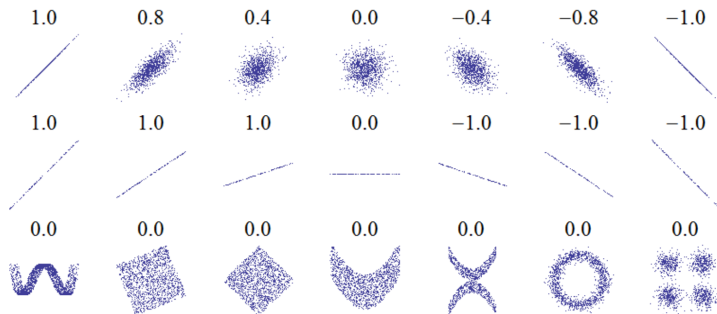
Ensure a Linear Model is Sensible



Source: Murphy Figure 2.6

The "Datasaurus Dozen": All datasets have the same correlation between the two variables!

Correlation versus Regression



Source: Murphy Figure 3.1

Datasets with the same correlation (as indicated above each dataset) between two variables do not need to have the same regression slope!

Estimating Linear Regression [cont'd]

- Estimate $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the mean squared error (MSE) or residual sum of square (RSS)

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad \text{Predicted/fitted values}$$

$$RSS = \sum_i (y_i - \hat{y}_i)^2 = \sum_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

$$MSE = \frac{1}{n} RSS$$

- Analytically derivable *least squares estimates* are

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where \bar{x} and \bar{y} are the sample means

R^2 Value:

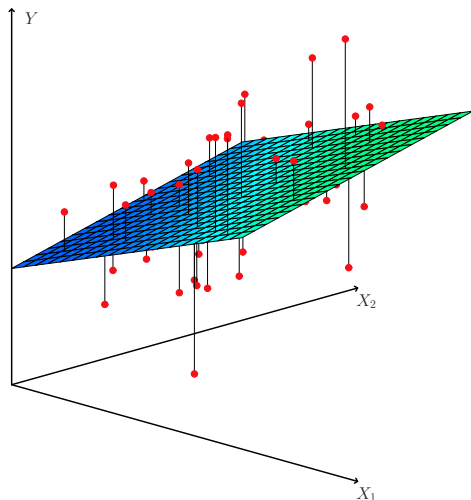
$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where $TSS = \sum_i (y_i - \bar{y})^2$ is the total sum of squares

Interpretation

- 1 Proportion of explained variance
- 2 Correlation of Y and \hat{Y}

Generalization to Multiple Predictors



$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon$$

Source: ISLR2 Figure 3.4

Regression with Qualitative Predictors

- Qualitative/categorical predictors (*factors* with multiple, exclusive *levels*) can be used in linear regression models using **dummy variables**:

$$x_{i1} = \begin{cases} 1 & \text{level "a"} \\ 0 & \text{else} \end{cases}$$

$$x_{i2} = \begin{cases} 1 & \text{level "b"} \\ 0 & \text{else} \end{cases}$$

$$x_{i3} = \begin{cases} 1 & \text{level "c"} \\ 0 & \text{else} \end{cases}$$

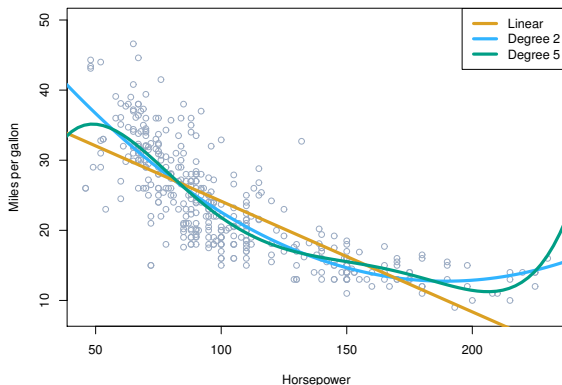
- Note: $x_{i1} = x_{i2} = x_{i3} = 0$ represents level d !

Example:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_1 X_2 + \epsilon$$

- ▶ Still linear in β_j
- ▶ Two **inputs** X_1 and X_2
- ▶ Four **predictors** or **features**: X_1 , X_1^2 , X_2 , $X_1 X_2$
- ▶ **Main effects** β_1 and β_3 ,
- ▶ **Interaction effect** β_4 ,
- ▶ A degree-2 **polynomial** effect β_2

Multiple Regression with Polynomials



Source: ISLR2 Figure 3.8

- ▶ Function more "flexible"
- ▶ Fit to data improves
- ▶ Bias decreases
- ▶ Training MSE decreases (what about test MSE?)

Linear Regression in R

Load libraries for data sets:

```
# Functions and data from the textbook # 'Modern  
# Applied Statistics with S'  
library(MASS)  
# Data sets from the textbook 'Introduction to  
# Statistical Learning with Applications in R'  
library(ISLR2)
```

The 'Boston' data set contains housing values in Boston in variable medv (median value).

```
Examine the data:  
  
# Get a description of the data  
?Boston  
  
# Get a summary and first few rows  
summary(Boston)  
head(Boston)  
  
# Bivariate scatterplots  
plot(Boston)
```

Fit a simple model, examine and plot the results:

```
# Fit a model with intercept only
fitted.model <- lm(medv ~ 1, data=Boston)
summary(fitted.model)

# Fit a model with predictor lstat
fitted.model <- lm(medv ~ lstat, data=Boston)
summary(fitted.model)

# Plot the data and the regression line
plot(medv ~ lstat, data=Boston)
abline(fitted.model, lwd=3, col='red')
```

The `predict()` function calculates \hat{y}_i :

```
# For training data
y.hat <- predict(fitted.model, Boston)

# For new data
newData <- data.frame(lstat = c(5, 10, 15))
predict(fitted.model, newData)
```

The `residuals()` function calculates $y_i - \hat{y}_i$:

```
# Plot the residuals against predicted values
plot(predict(fitted.model), residuals(fitted.model))
```

Build more complex models:

```
# Add another predictor
fitted.model <- lm(medv ~ lstat + age, data=Boston)

# Add all main effects
fitted.model <- lm(medv ~ ., data=Boston)

# Add interaction terms
fitted.model <- lm(medv ~ lstat + age + lstat:age, data=Boston)

# Shorter and equivalent
fitted.model <- lm(medv ~ lstat*age, data=Boston)
summary(fitted.model)
```


Add polynomial terms:

```
# Add a polynomial term; use the I(.) function  
# for any data transformations, such as log(),  
# or exp() or sqrt() as well as polynomials  
fitted.model <- lm(medv ~ lstat + I(lstat^2), data=Boston)  
summary(fitted.model)  
  
# Add all polynomial terms up to degree 5  
fitted.model <- lm(medv ~ poly(lstat, 5), data=Boston)  
  
# Note the coefficients for the polynomials in the summary  
summary(fitted.model)
```

Hands-On Exercises

(Source: ISLR2 Chapter 3)

Use the `Auto` data set from the ISLR2 library with `mpg` as the target.

- 1 Perform a linear regression with `horsepower` as predictor
- 2 Is there a relationship between the predictor and target? What form and how strong?
- 3 What is the predicted `mpg` value for a `horsepower` of 98?
- 4 Plot the response and predictor. Use the `abline()` function to add the regression line
- 5 Produce a scatterplot of all variables
- 6 Perform a linear regression of all main effects (except for the variable `name`)
- 7 Use the `*` and `:` symbols to add interaction effects.
- 8 Add transformations of the predictors (using the `I(.)` function) such as $\log(X)$, \sqrt{X} , X^2 .

Cross-Validation in R – Holdout Sample

Validation set approach:

```
# Randomly use half the Auto data as training sample
train.idx <- sample(nrow(Auto), nrow(Auto)/2)
train.data <- Auto[train.idx,]
test.data <- Auto[-train.idx,]

# Fit model to (train model on) a subset
fitted.model <- lm(mpg ~ horsepower, data=train.data)

# Calculate the training data MSE
mean((train.data$mpg - predict(fitted.model, train.data))^2)

# Calculate the validation data MSE
mean((test.data$mpg - predict(fitted.model, test.data))^2)
```

Cross-Validation in R

K-Fold Cross-Validation with $K = 5$:

```
library(boot)

# Fit a model with glm and show its summary
glm.fit <- glm(mpg ~ horsepower, data=Auto)

# 5-fold CV
cv.err <- cv.glm(Auto, glm.fit, K=5)
cv.err$delta[1]
```

LOOCV is K-Fold CV with $K = N$

```
# LOOCV is k-fold CV where K equals N, num of obs
cv.err <- cv.glm(Auto, glm.fit, K=nrow(Auto))
cv.err$delta[1]
```

Hands-On Exercises – Cross-Validation

Consider the Boston housing data set `Boston`.

- 1 Fit a regression model using `medv` as target, and `age`, `lstat`, and `ptratio` as predictors
- 2 Using the validation set approach, compute the test error of this model. Perform the following steps
 - 2.1 Split the data set using 75% for training and 25% for testing
 - 2.2 Fit the model to training data
 - 2.3 Predict the target for the testing data
 - 2.4 Compute the test error
- 3 Repeat the previous step 2 times, using different splits. How do the results change?
- 4 Average the test error of the four splits.
- 5 Calculate the test error estimate using LOOCV. Compare your result to that of step 4.
- 6 Calculate the test error estimate using 10-fold cross-validation. Compare the estimate to that of step 4

Goals

- ▶ Avoid overfitting
- ▶ Reduce variance
- ▶ "Shrink" regression coefficients towards zero
- ▶ Penalize coefficients that are "too high"
- ▶ Type of "**regularization**" (methods to prevent overfitting)

Ridge Regression

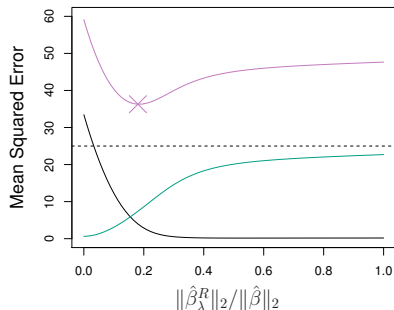
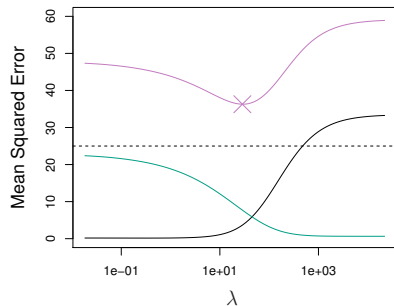
a.k.a Tikhonov Regularization

$$\text{Minimize } RSS + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \|\beta\|_2^2$$

- ▶ L2 regularizer (it penalizes the L2 norm of β)
- ▶ Parameter λ controls the amount of shrinkage
- ▶ Larger λ reduce variance but increase bias
- ▶ Not scale invariant: Standardize predictors

Ridge Regression

a.k.a Tikhonov Regularization

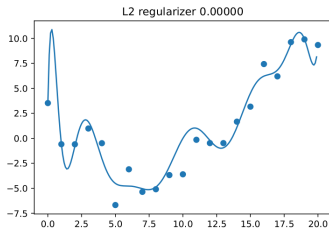


Source: ISLR2 Figure 6.5

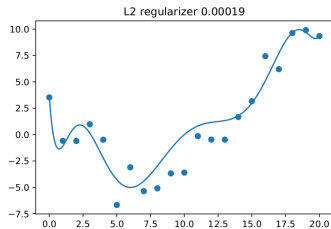
- Bias
- Variance
- MSE

Ridge Regression Example

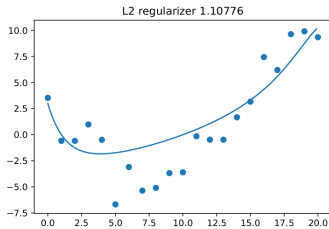
Fitting a Degree 14 Polynomial



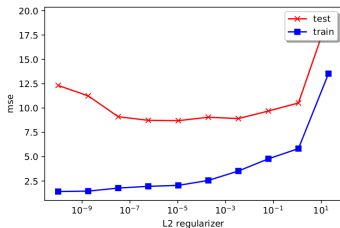
(a)



(b)



(c)



(d)

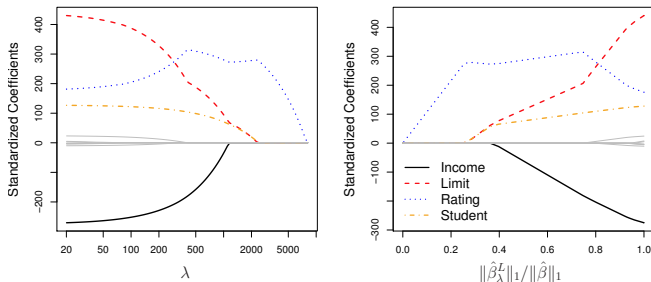
Source: Murphy Figure 4.5

Lasso regression

"Least Absolute Shrinkage and Selection Operator"

$$\text{Minimize } RSS + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \|\beta\|_1$$

- ▶ L1 regularizer (it penalizes the L1 norm of β)
- ▶ Lasso may exclude variables by forcing their β_j to 0
- ▶ Parsimonious, more interpretable models than ridge regression



Source: ISLR2
Figure 6.6

The Elastic Net penalty is a mix of L1-norm $\|\beta\|_1$ and L2-norm $\|\beta\|_2^2$ penalties, defined by α :

$$\lambda \left(\alpha \|\beta\|_1 + (1 - \alpha) \|\beta\|_2^2 \right)$$

- ▶ $\alpha = 0$: Ridge regression
- ▶ $\alpha = 1$: Lasso

The `glmnet()` function in the `glmnet` library for R uses the Elastic Net regularization method.

Ridge Regression in R

Use the `Hitters` data set to model `Salary` as outcome and other variables as predictors.

```
library(ISLR2)
library(glmnet)

# Remove missing values
Hitters <- na.omit(ISLR2::Hitters)
```

The `glmnet()` function requires separate `x` and `y` values, instead of a formula. To create dummy variables for categorical variables, use the `model.matrix()` function:

```
# Create dummy variables for categorical variables
# Remove intercept from model
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

Ridge Regression in R [cont'd]

Example for $\lambda = 10$ using `glmnet()` with $\alpha = 0$:

```
ridge.model <- glmnet(x, y, alpha=0, lambda=10)
```

Examine coefficients:

```
coef(ridge.model)
```

Examine L2 norm (penalty to RSS) (without intercept):

```
L2.norm = sqrt(sum(coef(ridge.model)[-1]^2))
```

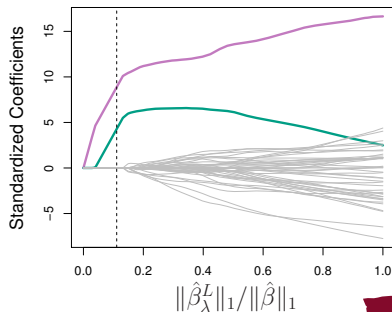
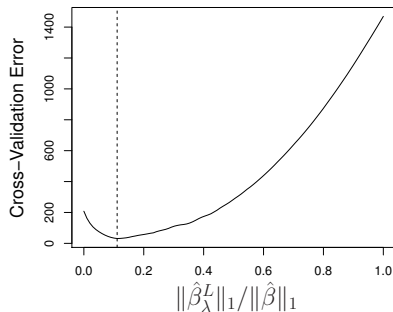
Examine MSE:

```
mean((y-predict(ridge.model, x))^2)
```

Selecting the Tuning Parameter

Grid Search

- ▶ Define a set/range of possible values for λ
- ▶ Cross-validate models for each value of λ
- ▶ Fit the final model to the optimal cross-validated error



Source: ISLR2 Figure 6.13

Ridge Regression in R – Optimal Lambda

Create holdout test data set:

```
# Randomly split the Hitters data
train.idx <- sample(nrow(Hitters), nrow(Hitters)/2)
x.train <- x[train.idx,]
x.test <- x[-train.idx,]
y.train <- y[train.idx]
y.test <- y[-train.idx]
```

Use the training set to find optimal λ with CV:

```
# 5-fold cross-validation, use MSE as metric
cv.out <- cv.glmnet(x.train, y.train, alpha=0,
                    nfolds=5, type.measure='mse')
```

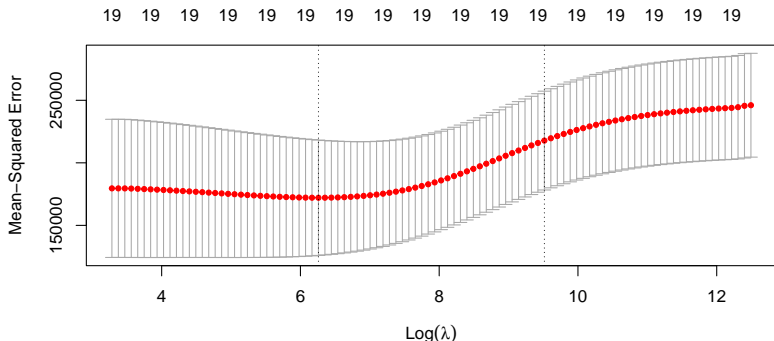
Ridge Regression in R – Optimal Lambda [cont'd]

Show the optimal lambda, the MSE and its standard error, and the number of non-zero coefficients.

```
print(cv.out)
plot(cv.out)
lambda.opt <-
  cv.out$lambda.min
```

```
> print(cv.out)
```

	Lambda	Index	Measure	SE	Nonzero
min	461.6	68	102536	13016	19
1se	2244.8	51	114493	19007	19



Fit test data using optimal λ :

```
ridge.model <- glmnet(x.test, y.test,  
                      alpha=0, lambda=lambda.opt)
```

Compare coefficients between ridge and unpenalized least squares:

```
coef(ridge.model)  
coef(lm(y.test ~ x.test + 1))
```

Ridge Regression in R – Prediction

Predict values for the test data set:

```
y.hat.test <- predict(ridge.model, newx=x.test,  
                      type='response')
```

Calculate test MSE to compare to the CV optimal MSE above:

```
mean((y.hat.test - y.test)^2)
```

Recall, the CV cross-validated error on the training set is an estimate/approximation of the real test error.

The Lasso in R [cont'd]

Example for $\lambda = 10$ using `glmnet()` with $\alpha = 1$:

```
lasso.model <- glmnet(x, y, alpha=1, lambda=10)
```

Examine coefficients:

```
coef(lasso.model)
```

Examine L1 norm (penalty to RSS) (without intercept):

```
L1.norm = sum(abs(coef(lasso.model)[-1]))
```

Examine MSE:

```
mean((y-predict(ridge.model, x))^2)
```

Use the training set to find optimal λ with CV:

```
# 5-fold cross-validation, use MSE as metric  
cv.out <- cv.glmnet(x.train, y.train, alpha=1,  
                    nfolds=5, type.measure='mse')
```

The Lasso in R – Optimal Lambda

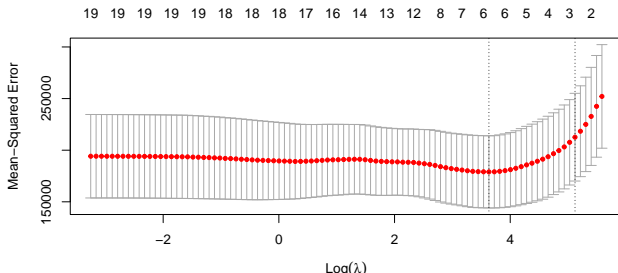
Show the optimal lambda, the MSE and its standard error, and the number of non-zero coefficients.

```
print(cv.out)
plot(cv.out)
lambda.opt <-
  cv.out$lambda.min
```

```
> print(cv.out)
```

	Lambda	Index	Measure	SE	Nonzero
min	33.33	22	107161	17195	6
1se	92.75	11	122071	22061	4

Note the number of non-zero coefficients in the result:



The Lasso in R – Prediction

Fit test data using optimal λ :

```
lasso.model <- glmnet(x.test, y.test,  
                      alpha=1, lambda=lambda.opt)
```

Compare coefficients between Lasso and unpenalized least squares:

```
coef(lasso.model)  
coef(lm(y.test ~ x.test + 1))
```

Predict values for the test data set and compute test MSE

```
y.hat.test <- predict(lasso.model, newx=x.test,  
                      type='response')  
mean((y.hat.test - y.test)^2)
```

Hands-On Exercises – Shrinkage Methods

Source: ISLR2, Chapter 6

Predict the number of applications received using the other variables in the `College` dataset

- 1 Split the data set into a training and a test set
- 2 Fit an unpenalized linear model on the training set. Report the test error.
- 3 Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error.
- 4 Fit a lasso model on the training set, with λ chosen by cross-validation. Report the test error.
- 5 Compare and contrast the results

Hands-On Exercises – Shrinkage Methods

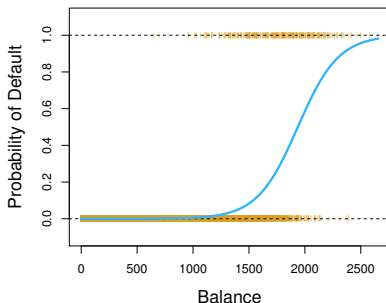
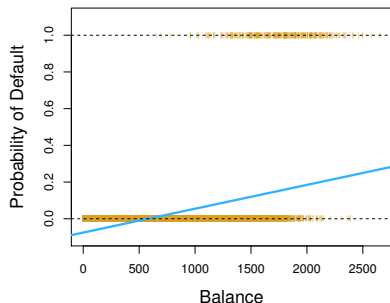
Source: ISLR2, Chapter 6

Predict the per-capita crime rate in the `Boston` data set using the other variables.

- 1 Split the data set into a training and a test set
- 2 Fit an unpenalized linear model on the training set. Report the test error.
- 3 Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error.
- 4 Fit a lasso model on the training set, with λ chosen by cross-validation. Report the test error.
- 5 Compare and contrast the results

Classification

- ▶ Qualitative (categorical) outcome
- ▶ Estimate or predict probabilities of class membership
- ▶ **Problem:** Linear combinations of predictors not in $[0, 1]$
- ▶ **Solution:** "Link" function transforms linear combinations of predictors



Source: ISLR2 Figure 4.2

Logistic Regression

Logistic / Sigmoid Function

$$\sigma(a) = \frac{1}{1 + e^{-a}} = \frac{e^a}{1 + e^a}$$



Binary Case (Binomial Logistic Regression)

$$p(X) = \sigma(\beta_0 + \beta_1 X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

where the linear predictor $\beta_0 + \beta_1 X$ are the **logits**

Predictions

Given estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, use the link function to predict class/outcome probabilities for observation X :

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}}$$

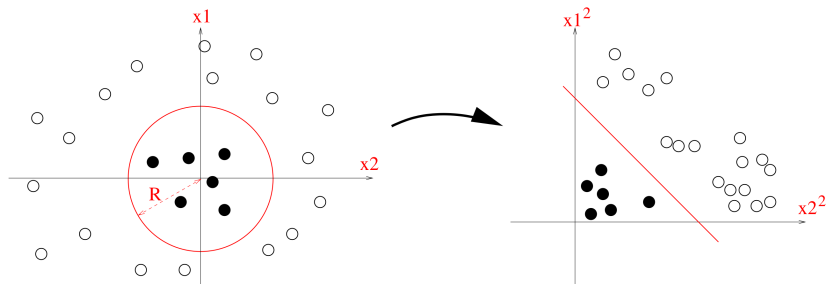
Multinomial Logistic Regression

- ▶ Same mathematical form
- ▶ Probability for class being k :

$$\Rightarrow \Pr(Y = k|X) = \frac{e^{\beta_{k0} + \beta_{k1}x_1 + \dots + \beta_{kp}x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_{l1}x_1 + \dots + \beta_{lp}x_p}}$$

Logistic Regression with Polynomials

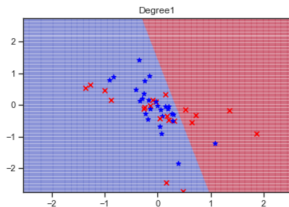
Transforming decision boundaries:



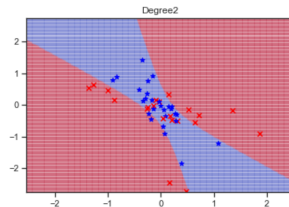
Source: Murphy Figure 10.3

Logistic Regression with Polynomials [cont'd]

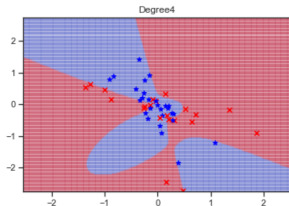
But beware of overfitting:



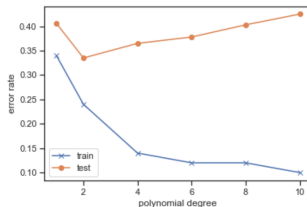
(a)



(b)



(c)



(d)

Source: Murphy Figure 10.4

Logistic Regression in R

Use the stock market data set `Smarket` to predict the binary outcome `Direction` (the direction of market changes, up or down) using prior returns as predictors:

```
library(ISLR2)
?Smarket
```

Fit to full data set. Note the link function in the `family` argument for the `glm()` function:

```
logreg.fitted <-  
  glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,  
        data=Smarket,  
        family=binomial(link='logit'))  
summary(logreg.fitted)
```

Logistic Regression in R

Use `predict()` to predict logits for data set:

```
logreg.logits <- predict(logreg.fitted, newdata = Smarket)
```

Use `predict()` to predict probabilities for data set:

```
# Predict probabilities for training test  
logreg.probs <- predict(logreg.fitted, newdata = Smarket,  
                        type='response')
```

Make classification decision using decision rule:

```
# Predict 'up' or 'down' based on probabilities  
# and a fixed threshold  
pred.direction <- rep(NA, nrow(Smarket))  
pred.direction[logreg.probs > .5] <- 'Up'  
pred.direction[logreg.probs <= .5] <- 'Down'
```

Logistic Regression in R

Compute confusion matrix:

```
# Compute confusion matrix  
logreg.cm <- table(pred.direction, Smarket$Direction)  
print(logreg.cm)
```

Compute accuracy as mean of correct classifications:

```
# Compute accuracy  
mean(pred.direction == Smarket$Direction)
```

Logistic Regression in R – Holdout Set

Split data to train and test set. Because this is time-dependent data, split by time to avoid mixing past and future data:

```
train.data <- Smarket[Smarket$Year < 2005,]  
test.data <- Smarket[!(Smarket$Year < 2005),]
```

Fit model to training set:

```
logreg.fitted <-  
  glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,  
        data=train.data, family=binomial(link='logit'))
```

Predict probabilities for test data:

```
logreg.probs <- predict(logreg.fitted,  
  newdata = test.data, type='response')
```

Make classification decision using decision rule:

```
pred.direction <- rep(NA, nrow(test.data))  
pred.direction[logreg.probs > .5] <- 'Up'  
pred.direction[logreg.probs <= .5] <- 'Down'
```


Logistic Regression in R – Holdout Set

Compute confusion matrix for test data:

```
logreg.cm <- table(pred.direction, test.data$Direction)  
print(logreg.cm)
```

Calculate accuracy as the mean of correct classifications:

```
mean(pred.direction == test.data$Direction)
```

Logistic Regression in R – Evaluation

Using the `ROCR` library for classifier evaluation:

```
library(ROCR)

# A prediction object collects predicted
# probabilities and true labels
pred.obj <- prediction(logreg.probs,
                       test.data$Direction)

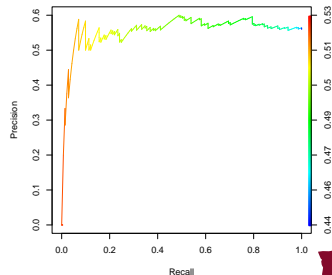
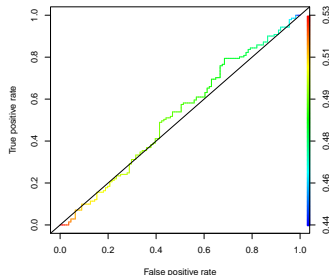
# Get some classifier performance metrics
# ROCR varies the threshold.
plot(performance(pred.obj, 'acc'))
plot(performance(pred.obj, 'prec'))
plot(performance(pred.obj, 'rec'))
plot(performance(pred.obj, 'f'))
performance(pred.obj, 'auc')@y.values[[1]]
```

Logistic Regression in R – Evaluation

continued ...

```
# ROC - True positive rate versus false positive rate
plot(performance(pred.obj, 'tpr', 'fpr'), colorize=T)
abline(0, 1)

# Precision/Recall plot
plot(performance(pred.obj, 'prec', 'rec'), colorize=T)
```



Naive Bayes Classifier

► **Bayes Theorem:**

$$\begin{aligned}\Pr(Y = c|X) &= \frac{p(X|Y = c) p(Y = c)}{p(X)} \\ &= \frac{p(X|Y = c) p(Y = c)}{\sum_{l=1}^K p(X|Y = l) p(Y = l)}\end{aligned}$$

► **Naive Bayes Assumption:** Within each class c , the D predictors are independent:

$$\begin{aligned}p(X|Y = c) &= p(x_1|Y = c) \times p(x_2|Y = c) \times \cdots \times p(x_D|Y = c) \\ &= \prod_{d=1}^D p(x_d|Y = c)\end{aligned}$$

► **Posterior probability:**

$$p(Y = c|X) = \frac{\left(\prod_{d=1}^D p(x_d|Y = c)\right) p(Y = c)}{\left(\sum_{l=1}^K \prod_{d=1}^D p(x_d|Y = l)\right) p(Y = l)}$$

Naive Bayes Classifier in R

Naive Bayes using the `naiveBayes` function in the `e1071` library:

```
library(e1071)

# Fit using same syntax as glm
nb.fitted <- naiveBayes(Direction~Lag1+Lag2, data=train.data)

# Output contains prior and conditional probabilities (and SD)
nb.fitted
```

Predict class membership and compute confusion matrix:

```
nb.predictions <- predict(nb.fitted, test.data)
nb.cm <- table(nb.predictions, test.data$Direction)
print(nb.cm)
```

Naive Bayes Classifier in R – Evaluation

Evaluate the classifier:

```
# Predict probabilities (for use with ROCR)
nb.probs <- predict(nb.fitted, test.data, type='raw')

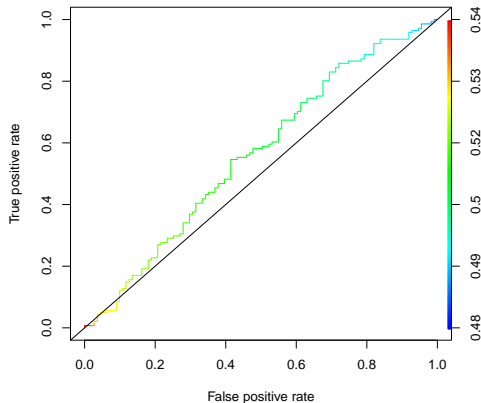
# Create an ROCR prediction object
nb.pred.obj <- prediction(nb.probs[, 'Up'],
                          test.data$Direction)
```

Assess ROC and AUC:

```
# Generate an ROC plot
plot(performance(nb.pred.obj, 'tpr', 'fpr'), colorize=T)
abline(0, 1)

# Compute the AUC
performance(nb.pred.obj, 'auc')@y.values[[1]]
```

Naive Bayes Classifier in R – Evaluation



KNN Classification in R

Using the `knn()` function from the `class` library:

```
library(class)
```

Use only two predictors, Lag1 and Lag2:

```
train.x <- cbind(train.data$Lag1, train.data$Lag2)
test.x <- cbind(test.data$Lag1, test.data$Lag2)
train.y <- train.data$Direction
test.y <- test.data$Direction
```

Make predictions from training set for test set, given true classes of training set ($k = 3$ and return probabilities):

```
knn.pred <- knn(train.x, test.x, train.y, k=3, prob=T)
```


KNN Classification in R

Evaluate the classifier against test data:

```
# Confusion matrix
table(knn.pred, test.y)
# Accuracy
mean(knn.pred == test.y)
```

Save class probabilities of the majority class:

```
knn.probs <- attributes(knn.pred)$prob
```

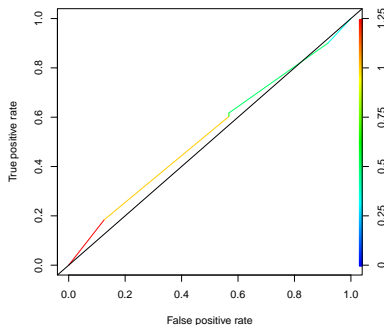
Compute class probabilities of the minority class:

```
knn.class.probs <- knn.probs
knn.class.probs[knn.pred=='Down'] <-
  1-knn.probs[knn.pred=='Down']
```

KNN Classification in R [cont'd]

Use ROCR functions to evaluate classifier:

```
knn.pred.obj <-  
  prediction(knn.class.probs, test.data$Direction)  
  
plot(performance(knn.pred.obj, 'tpr', 'fpr'), colorize=T)  
abline(0, 1)  
performance(knn.pred.obj, 'auc')@y.values[[1]]
```



Hands-On Exercises

Source: ISLR2, Chapter 4

Use the `Weekly` data set in the `ISLR2` package.

- 1 Use the full data set to perform a logistic regression with `Direction` as target. Which predictors are statistically significant?
- 2 Compute the confusion matrix and accuracy.
- 3 Use the 1990 to 2008 data for a training set and the 2009/2010 for a test set. Fit a logistic regression model with `Lag2` as the only predictor.
- 4 Repeat (3) using Naive Bayes
- 5 Repeat (3) using KNN with $K = 1$
- 6 Which model provides the best results on this data?

Hands-On Exercises

Source: ISLR2, Chapter 4

Use the `Auto` data set in the `ISLR2` package.

- 1 Create a binary variable, `mpg01` that contains a 1 if `mpg` is above its median, 0 otherwise. *Tip:* Use the `median()` function. Add the new variable to the data frame.
- 2 Split the data set into training and test set
- 3 Perform a logistic regression on the training data to predict `mpg01` from the other features. What is the test error of this model?
- 4 Repeat (3) using Naive Bayes
- 5 Repeat (3) using KNN with different values of K . What value of K performs best?

Hands-On Exercises

Source: ISLR2, Chapter 4

Using the `Boston` data set in the `ISLR2` library, fit classification models to predict whether a given census tract has a crime rate above or below the median.

- 1 Create a new binary variable `crime01` that is 1 if `crime` is above its median, and 0 otherwise. Combine this variable with the data frame. *Tip:* Use the `median()` function for this.
- 2 Split your data set into a training and test data set
- 3 Fit logistic regression, Naive Bayes, and KNN (with different K)
- 4 Describe your findings in terms of prediction error, precision, recall, F1 and AUC