Introduction to Classification & Regression Trees

ISLR Chapter 8

November 4, 2019
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome
  - High 1 if Sales > 8, otherwise 0

- Fit a Classification tree model to Price and Income

- Pick a predictor and a cutpoint to split data $X_j \leq s$ and $X_k > s$ to minimize deviance (or SSE for regression) - leads to a root node in a tree

- Continue splitting/partitioning data until stopping criterion is reached (number of observations in a node > 10 and within node deviance > 0.01 deviance of the root node)

- Prediction is mean or proportion of successes of data in terminal nodes

- Output is a decision tree

- Regression or classification function is nonlinear in predictors

- Captures interactions
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome High 1 if Sales > 8, otherwise 0
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome: High 1 if Sales > 8, otherwise 0
- Fit a Classification tree model to Price and Income
Classification and Regression Trees

Carseat data from ISLR package

▶ Binary Outcome High 1 if Sales > 8, otherwise 0
▶ Fit a Classification tree model to Price and Income
▶ Pick a predictor and a cutpoint to split data

\[ X_j \leq s \text{ and } X_k > s \]

... to minimize deviance (or SSE for regression) - leads to a root node in a tree
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome High 1 if Sales > 8, otherwise 0
- Fit a Classification tree model to Price and Income
- Pick a predictor and a cutpoint to split data

\[ X_j \leq s \text{ and } X_k > s \]

.. to minimize deviance (or SSE for regression) - leads to a root node in a tree

- continue splitting/partitioning data until stopping criterion is reached (number of observations in a node > 10 and within node deviance > 0.01 deviance of the root node)
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome $\text{High } 1$ if $\text{Sales} > 8$, otherwise $0$
- Fit a Classification tree model to $\text{Price}$ and $\text{Income}$
- Pick a predictor and a cutpoint to split data

\[ X_j \leq s \text{ and } X_k > s \]

to minimize deviance (or SSE for regression) - leads to a root node in a tree

- continue splitting/partitioning data until stopping criterion is reached (number of observations in a node $> 10$ and within node deviance $> 0.01$ deviance of the root node)
- Prediction is mean or proportion of successes of data in terminal nodes
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome \text{High} 1 if Sales $> 8$, otherwise 0
- Fit a Classification tree model to Price and Income
- Pick a predictor and a cutpoint to split data

\[ X_j \leq s \text{ and } X_k > s \]

to minimize deviance (or SSE for regression) - leads to a root node in a tree

- continue splitting/partitioning data until stopping criterion is reached (number of observations in a node $> 10$ and within node deviance $> 0.01$ deviance of the root node)
- Prediction is mean or proportion of successes of data in terminal nodes
- Output is a decision tree
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome High 1 if Sales > 8, otherwise 0
- Fit a Classification tree model to Price and Income
- Pick a predictor and a cutpoint to split data

\[ X_j \leq s \text{ and } X_k > s \]

... to minimize deviance (or SSE for regression) - leads to a root node in a tree

- continue splitting/partitioning data until stopping criterion is reached (number of observations in a node > 10 and within node deviance > 0.01 deviance of the root node)
- Prediction is mean or proportion of successes of data in terminal nodes
- Output is a decision tree
- regression or classification function is nonlinear in predictors
Classification and Regression Trees

Carseat data from ISLR package

- Binary Outcome: High 1 if Sales > 8, otherwise 0
- Fit a Classification tree model to Price and Income
- Pick a predictor and a cutpoint to split data
  \[ X_j \leq s \text{ and } X_k > s \]
  to minimize deviance (or SSE for regression) - leads to a root node in a tree
- continue splitting/partitioning data until stopping criterion is reached (number of observations in a node > 10 and within node deviance > 0.01 deviance of the root node)
- Prediction is mean or proportion of successes of data in terminal nodes
- Output is a decision tree
- regression or classification function is nonlinear in predictors
- Captures interactions
library(tree)
data(Carseats)
    Carseats = mutate(Carseats, High = factor(ifelse(Carseats$Sales <= 8,
        "No",
        "Yes ")))

tree.carseats = tree(High ~ Price + Income,
        data=Carseats)
Carseat Example

plot(tree.carseats)
text(tree.carseats)
partition.tree(tree.carseats)
points(Carseats$Price, Carseats$Income, col=Carseats$High)
Splits

tree.carseats

## node), split, n, deviance, yval, (yprob)
## * denotes terminal node
##
## 1) root 400 541.50 No ( 0.5900 0.4100 )
## 2) Price < 92.5 62 66.24 Yes ( 0.2258 0.7742 ) *
## 3) Price > 92.5 338 434.80 No ( 0.6568 0.3432 )
## 6) Price < 142 287 382.10 No ( 0.6167 0.3833 )
## 12) Income < 60.5 113 128.70 No ( 0.7434 0.2566 )
## 13) Income > 60.5 174 240.40 No ( 0.5345 0.4655 ) *
## 7) Price > 142 51 36.95 No ( 0.8824 0.1176 )
## 14) Income < 62.5 19 0.00 No ( 1.0000 0.0000 ) *
## 15) Income > 62.5 32 30.88 No ( 0.8125 0.1875 ) *
Summary

```
summary(tree.carseats)
```

```r
##
## Classification tree:
## tree(formula = High ~ Price + Income, data = Carseats)
## Number of terminal nodes:  5
## Residual mean deviance:  1.18 = 466.2 / 395
## Misclassification error rate: 0.325 = 130 / 400
```
tree.carseats = `tree(formula = High ~ . - Sales, data = Carseats)`

`summary(tree.carseats)`

```r
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc" "Price" "Income" "CompPrice"
## [6] "Advertising" "Age" "US"
## Number of terminal nodes: 27
## Residual mean deviance: 0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

Overfitting?
set.seed(2)
train = sample(1:nrow(Carseats), 200)
Carseats.test = Carseats[-train,]

tree.carseats = tree(High ~ . - Sales, 
                      data = Carseats, subset = train)

tree.pred = predict(tree.carseats, Carseats.test, type = "class")

table(tree.pred, Carseats.test$High)

# # tree.pred  No Yes
# #   No    104  33
# #   Yes    13  50

(33 + 13) / 200  # classification error

# [1] 0.23
Cost-Complexity Pruning

1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations
Cost-Complexity Pruning

1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations.

2. Prediction for region $m$ is the Class $c$ that $\max_c \hat{\pi}_{mc}$
Cost-Complexity Pruning

1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations
2. Prediction for region \( m \) is the Class \( c \) that \( \max_c \hat{\pi}_{mc} \)
3. Snip off the least important splits via cost-complexity pruning to the tree in order to obtain a sequence of best subtrees indexed by cost parameter \( k \),
Cost-Complexity Pruning

1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations
2. Prediction for region \( m \) is the Class \( c \) that \( \max_c \hat{\pi}_{mc} \)
3. Snip off the least important splits via cost-complexity pruning to the tree in order to obtain a sequence of best subtrees indexed by cost parameter \( k \),

\[
\frac{N_{miss}}{N} - k|\mathcal{T}|
\]

missclassification error penalized by number of terminal nodes
Cost-Complexity Pruning

1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations.

2. Prediction for region $m$ is the Class $c$ that $\max_c \hat{\pi}_{mc}$.

3. Snip off the least important splits via cost-complexity pruning to the tree in order to obtain a sequence of best subtrees indexed by cost parameter $k$,

$$\frac{N_{miss}}{N} - k|T|$$

missclassification error penalized by number of terminal nodes.

4. Using $K$-fold cross validation, compute average cost-complexity for each $k$. 
Cost-Complexity Pruning

1. Grow a large tree on training data, stopping when each terminal node has fewer than some minimum number of observations.

2. Prediction for region $m$ is the Class $c$ that $\max_c \hat{\pi}_{mc}$

3. Snip off the least important splits via cost-complexity pruning to the tree in order to obtain a sequence of best subtrees indexed by cost parameter $k$,

$$\frac{N_{\text{miss}}}{N} - k|T|$$

missclassification error penalized by number of terminal nodes

4. Using $K$-fold cross validation, compute average cost-complexity for each $k$

5. Pick subtree with smallest penalized error
Pruning via Cross Validation

```
set.seed(2)
cv.carseats = cv.tree(tree.carseats, FUN=prune.misclass)
```
prune.carseats = prune.misclass(tree.carseats, best = 9)
Miss-classification after Selection

tree.pred = predict(prune.carseats, Carseats.test, type = "class")
table(tree.pred, Carseats.test$High)

# # tree.pred No Yes
# # No 97 25
# # Yes 20 58

(97 + 58)/200  # classified Correctly

# [1] 0.775
Tree with Random Split of Data

ShelveLoc:ac
Price < 108.5
CompPrice < 123.5
Income < 57.5
Price < 89.5 Age < 63.5
Urban:a Advertising < 2.5
Price < 106Income < 36.5
No No
Yes
No
No
Yes
No
No
No
Yes
No
No
No
Yes
No
No
No
Yes
No
No
No
Yes
Yes
Yes
Tree with another Random Split of Data

Price < 126.5

ShelveLoc:a

Income < 63.5

Price < 93.5

Advertising < 9.5

Age < 52.5

Price < 111

US:a

CompPrice < 121.5

Age < 74.5

Income < 34.5

CompPrice < 106.5

Age < 59
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)

Bagging (Bootstrap Aggregation) estimate is

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$$

Trees are grown deep so little bias (although could prune)

Reduce variance by averaging many trees across the bootstrap samples
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size $n$ with replacement)
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size $n$ with replacement)
  - Generate B bootstrap sample of observations from the single training data

\[ \hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum \hat{f}_b(x) \]
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- Do not have access to multiple training sets so create them via bootstrap samples (sample of size $n$ with replacement)
  - Generate $B$ bootstrap sample of observations from the single training data
  - Calculate predictions for the $b$th sample $\hat{f}_b(x)$

Bagging (Bootstrap Aggregation) estimate is $\hat{f}_{bag}(x) = \frac{1}{B} \sum \hat{f}_b(x)$

Trees are grown deep so little bias (although could prune)

Reduce variance by averaging many trees across the bootstrap samples
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size \( n \) with replacement)
  - Generate \( B \) bootstrap sample of observations from the single training data
  - Calculate predictions for the \( b \)th sample \( \hat{f}(x) \)
  - Bagging (Bootstrap Aggregation) estimate is
    \[
    \hat{f}_{bag}(x) = \frac{1}{B} \sum \hat{f}(x)
    \]
- Trees are grown deep so little bias (although could prune)
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size $n$ with replacement)
  - Generate B bootstrap sample of observations from the single training data
  - Calculate predictions for the $b$th sample $\hat{f}_b(x)$
  - Bagging (Bootstrap Aggregation) estimate is
    \[
    \hat{f}_{bag}(x) = \frac{1}{B} \sum \hat{f}_b(x)
    \]
- Trees are grown deep so little bias (although could prune)
- Reduce variance by averaging many trees across the bootstrap samples
Bagging: Bootstrap Aggregation

- Splitting data into random partitions and fitting a tree model on each half may lead to very different predictions (high variability)
- Reduce variability by averaging over multiple training sets!
- do not have access to multiple training sets so create them via bootstrap samples (sample of size \( n \) with replacement)
  - Generate \( B \) bootstrap sample of observations from the single training data
  - Calculate predictions for the \( b \)th sample \( \hat{f}_b(x) \)
  - Bagging (Bootstrap Aggregation) estimate is

\[
\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum \hat{f}_b(x)
\]

- Trees are grown deep so little bias (although could prune)
- Reduce variance by averaging many trees across the bootstrap samples
Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification.
Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification. Ways to improving Trees through multiple trees in some ensemble:

- Bagging
Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification. Ways to improving Trees through multiple trees in some ensemble:

- Bagging
- Random Forests
Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification. Ways to improving Trees through multiple trees in some ensemble:

- Bagging
- Random Forests
- Boosting
Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification. Ways to improving Trees through multiple trees in some ensemble:

- Bagging
- Random Forests
- Boosting
- BART (Bayesian Additive Regression Trees)
Trees are simple to understand, but not as competitive with other supervised learning approaches for prediction/classification. Ways to improving Trees through multiple trees in some ensemble:

▶ Bagging
▶ Random Forests
▶ Boosting
▶ BART (Bayesian Additive Regression Trees)

Combining trees will yield improved prediction accuracy, but with loss of interpretability.