Week 4: Unsupervised Learning

Kenneth Benoit

TCD HT 2017

06 March 2017

Day 7 Outline

Association rules

Tree-based Methods

Unsupervised "learning": scaling distance

- Features are treated as a quantitative matrix of features
  - (standardized) variables
  - (normalized) word feature counts, in text
- Different possible definitions of *distance*
  - see for instance `summary(pr_DB)` from `proxy` library
- Works on any quantitative matrix of features

Distance measures

```r
library(proxy, warn.conflicts = FALSE, quietly = TRUE)
summary(pr_DB)

## * Similarity measures:
## Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice,
## eDice, eJaccard, Fager, Faith, Gower, Hamman, Jaccard,
## Kulczynski1, Kulczynski2, Michael, Mountford, Mozley, Ochiai,
## Pearson, Phi, Phi-squared, Russel, simple matching, Simpson,
## Stiles, Tanimoto, Tschuprow, Yule, Yule2
##
## * Distance measures:
## Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean,
## fJaccard, Geodesic, Hellinger, Kullback, Levenshtein, Mahalanobis,
## Manhattan, Minkowski, Podani, Soergel, supremum, Wave, Whittaker
```

Characteristics of similarity measures

Let A and B be any two documents in a set and $d(A, B)$ be the distance between A and B.

1. $d(x, y) \geq 0$ (the distance between any two points must be non-negative)
2. $d(A, B) = 0$ iff $A = B$ (the distance between two documents must be zero if and only if the two objects are identical)
3. $d(A, B) = d(B, A)$ (distance must be symmetric: A to B is the same distance as from B to A)
4. $d(A, C) \leq d(A, B) + d(B, C)$ (the measure must satisfy the triangle inequality)

Euclidean distance

Between document A and B where j indexes their features, where y_{ij} is the value for feature j of document i

- Euclidean distance is based on the Pythagorean theorem
- Formula

$$\sqrt{\sum_{j=1}^{j}(y_{Aj} - y_{Bj})^2} \tag{1}$$

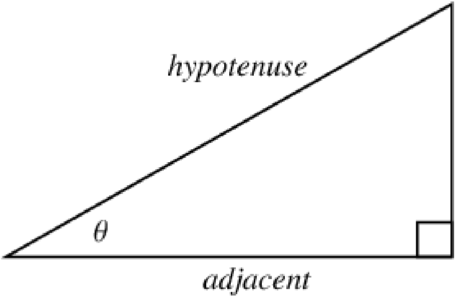
- In vector notation:

$$\|\mathbf{y}_A - \mathbf{y}_B\| \tag{2}$$

- Can be performed for any number of features J (or V as the vocabulary size is sometimes called – the number of columns in of the dfm, same as the number of feature types in the corpus)

A geometric interpretation of "distance"

In a right angled triangle, the cosine of an angle θ or $\cos(\theta)$ is the length of the adjacent side divided by the length of the hypotenuse

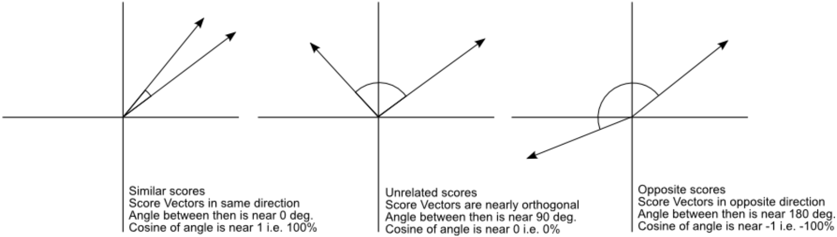


We can use the vectors to represent the text location in a V-dimensional vector space and compute the angles between them

Cosine similarity

- Cosine distance is based on the size of the angle between the vectors
- Formula
$$\frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\|\|\mathbf{y}_B\|} \tag{3}$$
- The $\cdot$ operator is the dot product, or $\sum_j y_{Aj} y_{Bj}$
- The $\|\mathbf{y}_A\|$ is the vector norm of the (vector of) features vector $\mathbf{y}$ for document A, such that $\|\mathbf{y}_A\| = \sqrt{\sum_j y_{Aj}^2}$
- Nice property: cosine measure is independent of unit size, because it deals only with the angle of the vectors
- Ranges from -1.0 to 1.0 for term frequencies, or 0 to 1.0 for normalized term frequencies (or tf-idf)

Cosine similarity illustrated

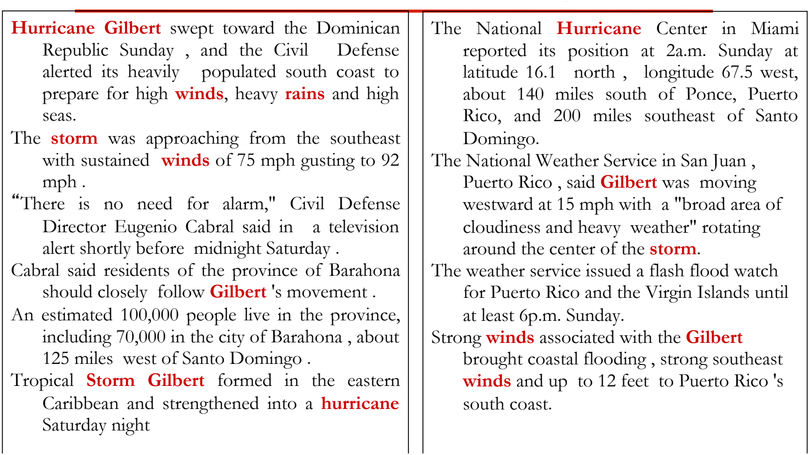


Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%

Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%

Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

Example text



Hurricane Gilbert swept toward the Dominican Republic Sunday , and the Civil Defense alerted its heavily populated south coast to prepare for high **winds**, heavy **rains** and high seas.

The **storm** was approaching from the southeast with sustained **winds** of 75 mph gusting to 92 mph .

"There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday .

Cabral said residents of the province of Barahona should closely follow **Gilbert** 's movement .

An estimated 100,000 people live in the province, including 70,000 in the city of Barahona , about 125 miles west of Santo Domingo .

Tropical **Storm Gilbert** formed in the eastern Caribbean and strengthened into a **hurricane** Saturday night

The National **Hurricane** Center in Miami reported its position at 2a.m. Sunday at latitude 16.1 north , longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.

The National Weather Service in San Juan , Puerto Rico , said **Gilbert** was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the **storm**.

The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6p.m. Sunday.

Strong **winds** associated with the **Gilbert** brought coastal flooding , strong southeast **winds** and up to 12 feet to Puerto Rico 's south coast.

Example text: selected terms

- Document 1
  Gilbert: 3, hurricane: 2, rains: 1, storm: 2, winds: 2

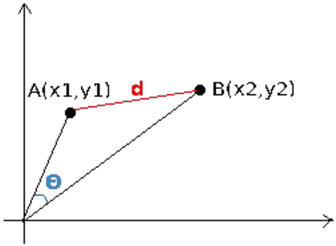
- Document 2
  Gilbert: 2, hurricane: 1, rains: 0, storm: 1, winds: 2

Example text: cosine similarity in R

```
require(quanteda, warn.conflicts = FALSE, quietly = TRUE)

## quanteda version 0.9.9.27
## Using 3 of 4 cores for parallel computing

toyDfm <-
    dfm(c(doc1 = "Gilbert Gilbert Gilbert hurricane hurricane rains storm storm
          doc2 = "Gilbert Gilbert hurricane storm"))
toyDfm

## Document-feature matrix of: 2 documents, 4 features (12.5% sparse).
## 2 x 4 sparse Matrix of class "dfmSparse"
##       features
## docs   gilbert hurricane rains storm
##   doc1       3         2     1     2
##   doc2       2         1     0     1

textstat_simil(toyDfm, method = "cosine")

##           doc1
## doc2 0.9622504
```

Relationship to Euclidean distance

- Cosine similarity measures the similarity of vectors with respect to the origin
- Euclidean distance measures the distance between particular points of interest along the vector



Jacquard coefficient

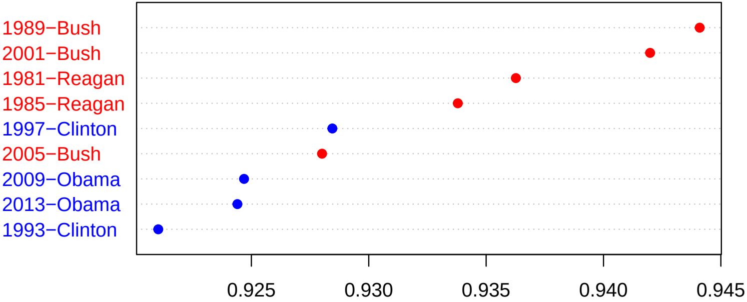
- Similar to the Cosine similarity
- Formula

$$\frac{\mathbf{y}_A \cdot \mathbf{y}_B}{\|\mathbf{y}_A\| + \|\mathbf{y}_B\| - \mathbf{y}_A \cdot \mathbf{y} y_B} \tag{4}$$

- Ranges from 0 to 1.0

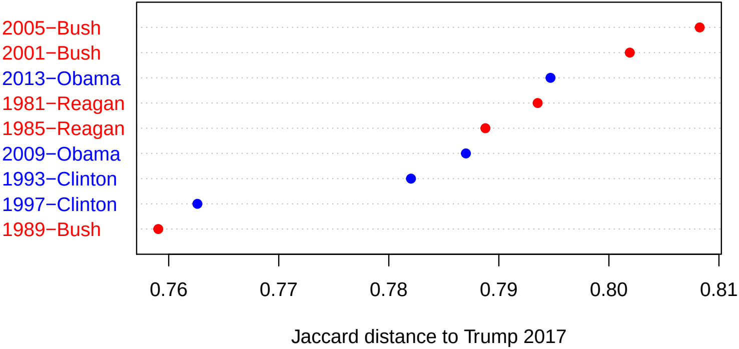
Example: Inaugural speeches, cosine distance to Trump 2017

```
presDfm <- dfm(corpus_subset(inaugCorpus, Year > 1980),
               remove = c(stopwords("english"), "will"), stem = TRUE)
pressimil <- as.matrix(textstat_simil(presDfm, method = "cosine"))[1:9, 10]
col = c(rep("red", 3), rep("blue", 2), rep("red", 2), rep("blue", 2))
sortorder <- order(pressimil)
dotchart(pressimil[sortorder], pch = 19, col = col[sortorder],
         xlab="Cosine similarity to Trump 2017")
```



Example: Jaccard distance to Obama

```
presdist <- as.matrix(textstat_dist(presDfm, method = "jaccard"))[1:9, 10]
sortorder <- order(presdist)
dotchart(presdist[sortorder], pch = 19, col = col[sortorder],
         xlab="Jaccard distance to Trump 2017")
```



Non-parametric dimensional reduction methods

- Non-parametric methods are algorithmic, involving no "parameters" in the procedure that are estimated
- Hence there is no uncertainty accounting given distributional theory
- Advantage: don't have to make assumptions
- Disadvantages:
  - cannot leverage probability conclusions given distribtional assumptions and statistical theory
  - results highly fit to the data
  - not really assumption-free (if we are honest)

Principal Components Analysis

- For a set of features $X_1, X_2, \ldots, X_p$, typically centred (to have mean 0)
- the first principal component is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p$$

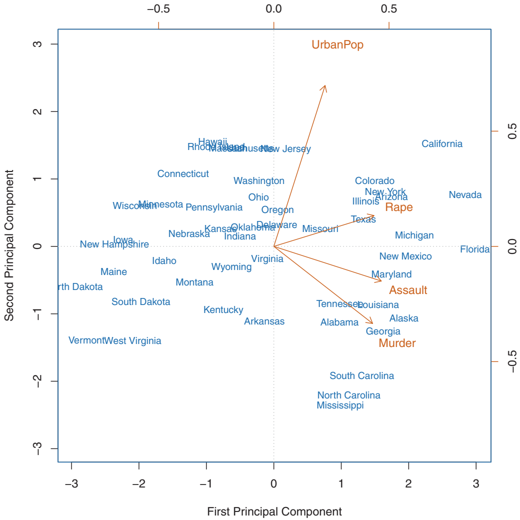
  that has the largest variance
- normalized means that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- the elements $\phi_{11}, \ldots, \phi_{p1}$ are the loadings of the first principal component
- the second principal component is the linear combination Z_2 of $X_1, X_2, \ldots, X_p$ that has maximal variance out of all linear combinations that are *uncorrelated* with Z_1

PCA factor loadings example

Murder
Assault
UrbanPo
Rape

TABLE 10.1. *The princip*

PCA factor loadings biplot



PCA projection illustrated

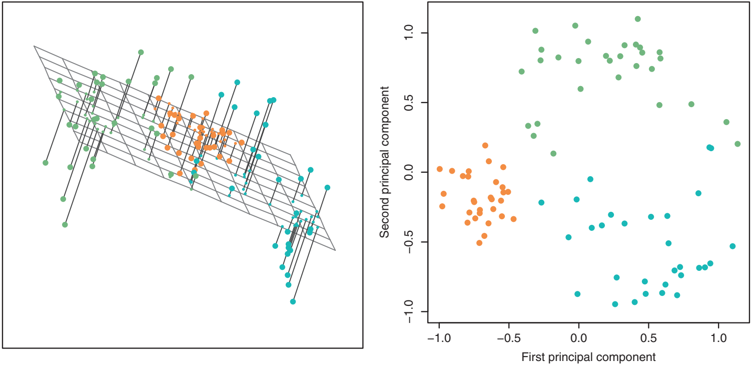


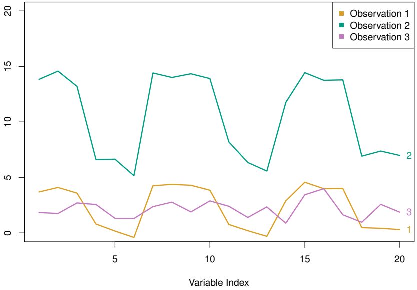
FIGURE 10.2. *Ninety observations simulated in three dimensions.* Left: *the first two principal component directions span the plane that best fits the data. It minimizes the sum of squared distances from each point to the plane.* Right: *the first two principal component score vectors give the coordinates of the projection of the 90 observations onto the plane. The variance in the plane is maximized.*

Correspondence Analysis

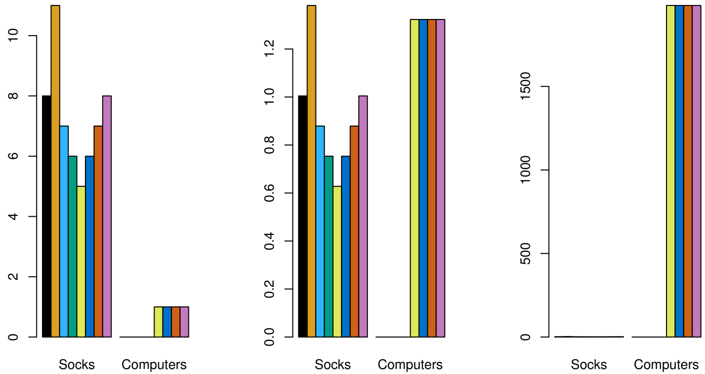
- CA is like factor analysis for categorical data
- Following normalization of the marginals, it uses Singular Value Decomposition to reduce the dimensionality of the word-by-text matrix
- This allows projection of the positioning of the words as well as the texts into multi-dimensional space
- The number of dimensions – as in factor analysis – can be decided based on the eigenvalues from the SVD

Choice of Dissimilarity Measure

- So far have used Euclidean distance.
- An alternative is correlation-based distance which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.



Scaling of the variables matters



Practical issues

- Should the observations or features first be standardized in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering,
  - What dissimilarity measure should be used?
  - What type of linkage should be used?
- How many clusters to choose? (in both K-means or hierarchical clustering). Difficult problem. No agreed-upon method. See Elements of Statistical Learning, chapter 13 for more details.

Conclusions

- Unsupervised learning is important for understanding the variation and grouping structure of a set of unlabeled data, and can be a useful pre-processor for supervised learning.
- It is intrinsically more difficult than supervised learning because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy).
- It is an active field of research, with many recently developed tools such as self-organizing maps, independent components analysis and spectral clustering.
- See The Elements of Statistical Learning, chapter 14.

Association rules

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.

Introduction to association rules

- Association rule mining is used to discover objects or attributes that frequently occur together, e.g.
  - movies or music that users prefer
  - baskets of products purchased online or in-store
- Used extensively in recommendation engines
- Terminology:

  transaction a bundle of associated items, such as a collection of movies watched or items puchased, forming the unit of analysis

  itemset the items that make up a transaction, such as purchases, web sites visited, movies watched, etc.
- Many different algorithms, but we will focus on one: the *a priori* algorithm

The *apriori* algorithm

- Two core notions:

  support
  : the support of an item X is the number of transactions that contain X divided by the total number of transactions

  confidence
  : expresses our "confidence" in the relation *if* X*, then* Y

    - formally:

$$\text{support}(\text{union}(X, Y))/\text{support}(X)$$

- The goal is to discover the interesting rules in a dataset above some pre-defined thresholds of support and confidence, such as 10% and 60%)

apriori Rules Example

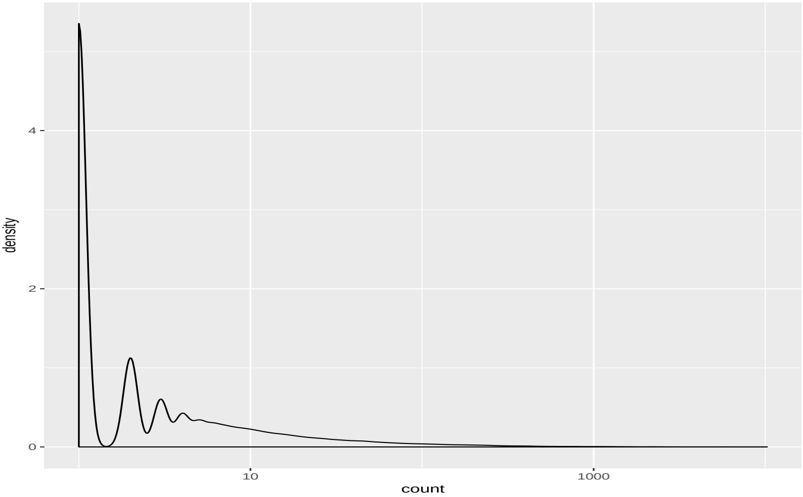
```
## book data can be found from https://github.com/WinVector/zmPDSwR/tree/master/
# load in book purchase transactions
require(arules, quietly = TRUE, warn.conflicts = FALSE)
(bookbaskets <- read.transactions("bookdata.tsv",
                                  format = "single", sep = "\t",
                                  cols = c("userid", "title"), rm.duplicates =


## distribution of transactions with duplicates:
## items
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 701 222 106  68  43  39  23  24  18  18  16  10   7   7  13   7   8   5
##  19  20  21  22  23  25  26  27  28  29  30  31  33  34  35  38  39  42
##   3   9   4   4   3   2   2   5   4   5   4   4   1   2   1   1   1   2
##  44  45  47  48  49  52  56  57  59  61  63  71  73  80  84  86  91  93
##   1   1   1   1   1   1   2   1   2   1   2   1   1   1   1   1   1   1
##  95  96  99 103 158 206 260 891
##   1   1   1   1   1   2   1   1
## transactions in sparse format with
##  92108 transactions (rows) and
##  220447 items (columns)
```

apriori Rules Example

```
# summarize basket sizes
basketSizes <- size(bookbaskets)
summary(basketSizes)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     1.0    11.1     4.0 10250.0
```

apriori Rules Example

```
# plot the distribution of basket sizes (log10 scale)
require(ggplot2, quietly = TRUE)
ggplot(data.frame(count = basketSizes)) + scale_x_log10() +
    geom_density(aes(x = count))
```



apriori Rules Example

```
# which books are people reading?
bookFreq <- itemFrequency(bookbaskets)
bookCount <- (bookFreq / sum(bookFreq)) * sum(basketSizes)
orderedBooks <- sort(bookCount, decreasing = TRUE)
head(orderedBooks, 10)

##                                     Wild Animus
##                                            2502
##                        The Lovely Bones: A Novel
##                                            1295
##                               She's Come Undone
##                                             934
##                               The Da Vinci Code
##                                             905
##           Harry Potter and the Sorcerer's Stone
##                                             832
##                      The Nanny Diaries: A Novel
##                                             821
##                                 A Painted House
##                                             819
##                           Bridget Jones's Diary
##                                             772
##                         The Secret Life of Bees
##                                             762
## Divine Secrets of the Ya-Ya Sisterhood: A Novel
```

apriori Rules Example

```
# mine the rules using the apriori algorithm
rules <- apriori(bookbaskets_use,
                 parameter = list(support = 0.002, confidence = 0.75))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5   0.002      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 81
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[216031 item(s), 40822 transaction(s)] done [0.72s].
## sorting and recoding items ... [1256 item(s)] done [0.03s].
## creating transaction tree ... done [0.02s].
## checking subsets of size 1 2 3 4 5 done [0.05s].
## writing ... [191 rule(s)] done [0.00s].
## creating S4 object  ... done [0.06s].
```

apriori Rules Example

```
summary(rules)


## set of 191 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5
##  11 100  66  14
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   3.000   3.435   4.000   5.000
##
## summary of quality measures:
##     support           confidence          lift
##  Min.   :0.002009   Min.   :0.7500   Min.   : 40.89
##  1st Qu.:0.002131   1st Qu.:0.8113   1st Qu.: 86.44
##  Median :0.002278   Median :0.8468   Median :131.36
##  Mean   :0.002593   Mean   :0.8569   Mean   :129.68
##  3rd Qu.:0.002695   3rd Qu.:0.9065   3rd Qu.:158.77
##  Max.   :0.005830   Max.   :0.9882   Max.   :321.89
##
## mining info:
##             data ntransactions support confidence
##  bookbaskets_use         40822   0.002       0.75
```

apriori Rules Example

```
inspect(head((sort(rules, by = "confidence")), n = 5))

##     lhs                                                rhs
## [1] {Four to Score,
##      High Five,
##      Seven Up,
##      Two for the Dough}                            => {Three To Get Deadly :
## [2] {Harry Potter and the Order of the Phoenix,
##      Harry Potter and the Prisoner of Azkaban,
##      Harry Potter and the Sorcerer's Stone}        => {Harry Potter and the
## [3] {Four to Score,
##      High Five,
##      One for the Money,
##      Two for the Dough}                            => {Three To Get Deadly :
## [4] {Four to Score,
##      Seven Up,
##      Three To Get Deadly : A Stephanie Plum Novel,
##      Two for the Dough}                            => {High Five}
## [5] {High Five,
##      Seven Up,
##      Three To Get Deadly : A Stephanie Plum Novel,
##      Two for the Dough}                            => {Four to Score}
```

Tree-based Methods

Tree-based Methods

- Here we describe tree-based methods for regression and classification.
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods.

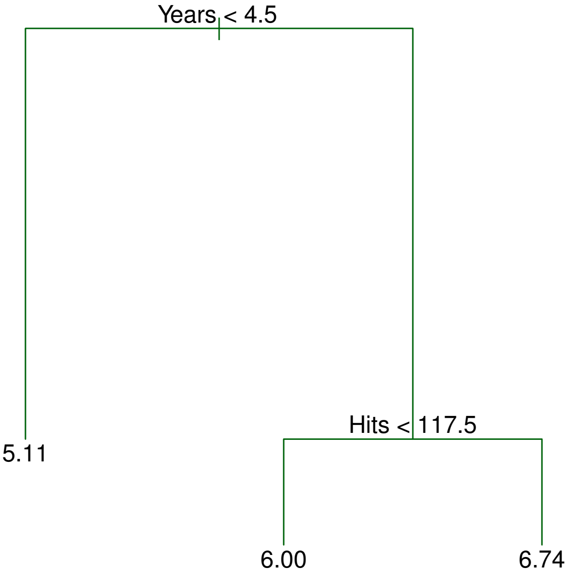
Pros and Cons

- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Hence we also discuss bagging, random forests, and boosting. These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

The Basics of Decision Trees

- Decision trees can be applied to both regression and classification problems.
- We first consider regression problems, and then move on to classification.

Baseball salary data

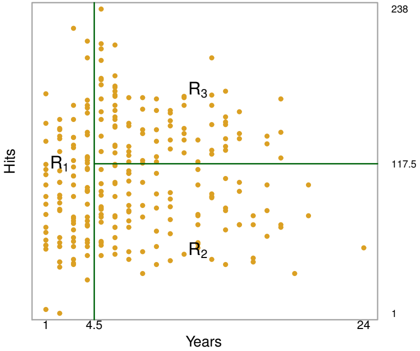


Details of previous figure

- For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.
- At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to $Years < 4.5$, and the right-hand branch corresponds to $Years \geq 4.5$.
- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

Results

- Overall, the tree stratifies or segments the players into three regions of predictor space: $R_1 = \{X | Years < 4.5\}$, $R_2 = \{X | Years \geq 4.5, Hits < 117.5\}$, and $R_3 = \{X | Years \geq 4.5, Hits \geq 117.5\}$.



Terminology for Trees

- In keeping with the tree analogy, the regions R_1, R_2, and R_3 are known as terminal nodes.
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as internal nodes.
- In the hitters tree, the two internal nodes are indicated by the text $\mathit{Years} < 4.5$ and $\mathit{Hits} < 117.5$.

Interpretation of Results

- *Years* is the most important factor in determining *Salary*, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of *Hits* that he made in the previous year seems to play little role in his *Salary*.
- But among players who have been in the major leagues for five or more years, the number of *Hits* made in the previous year does affect *Salary*, and players who made more *Hits* last year tend to have higher salaries.
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain.

Details of the tree-building process

1. We divide the predictor space – that is, the set of possible values for $X_1, X_2, \ldots, X_p$ – into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

More details of the tree-building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

More details of the tree-building process

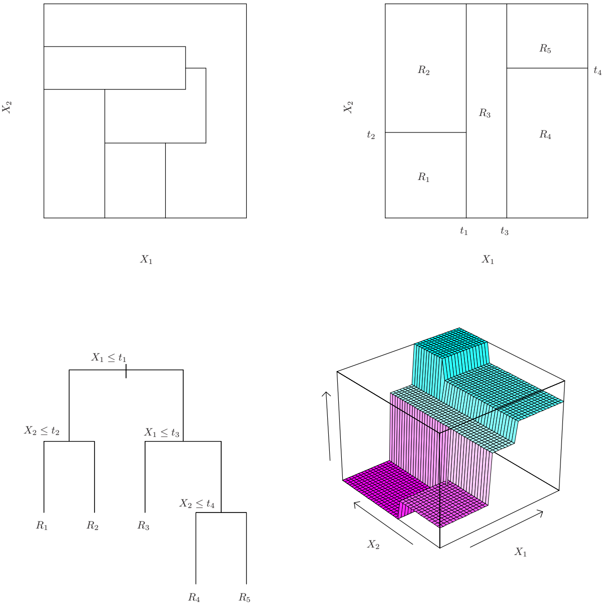
- Unfortunately, it is computationally unfeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting.
- The approach is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Details – Continued

- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

Predictions

- We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- A five-region example of this approach is shown in the next slide.


X_2
X_1
R_2
R_3
R_5
R_4
R_1
t_2
t_4
t_1
t_3
X_2
X_1
$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5
X_2
X_1

Details of previous figure

- Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting.
- Top Right: The output of recursive binary splitting on a two-dimensional example.
- Bottom Left: A tree corresponding to the partition in the top right panel.
- Bottom Right: A perspective plot of the prediction surface corresponding to that tree.

Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.
- A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, but is too short-sighted: a seemingly worthless split early on in the tree might be followed by a very good split – that is, a split that leads to a large reduction in RSS later on.

Pruning a tree – continued

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree.
- Cost complexity pruning – also known as weakest link pruning – is used to do this.
- we consider a sequence of trees indexed by a non-negative tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle (i.e. the subset of predictor space) corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the mean of the training observations in R_m.

Choosing the best subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- We select an optimal value $\hat{\alpha}$ using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

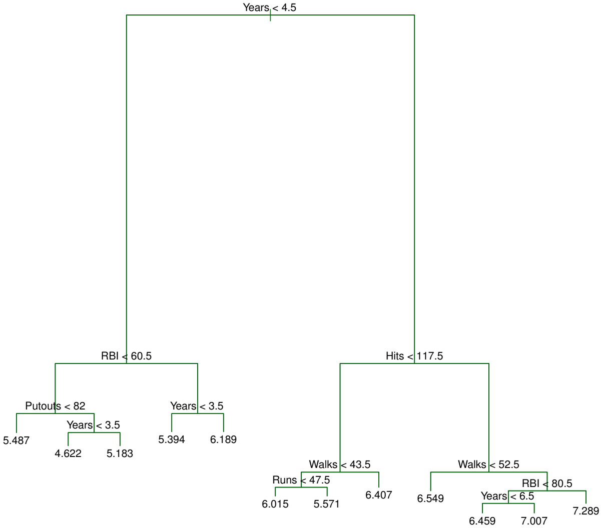
Summary: tree algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. For each $k = 1, \ldots, K$:
    3.1 Repeat Steps 1 and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the kth fold.
    3.2 Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α. Average the results, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

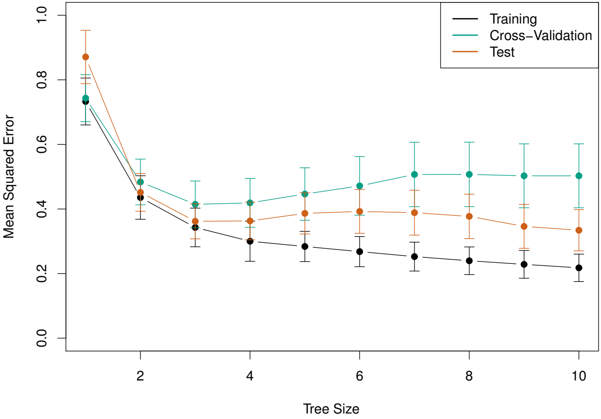
Baseball example continued

- First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set.
- We then built a large regression tree on the training data and varied α in order to create subtrees with different numbers of terminal nodes.
- Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of α.

Baseball example continued



Baseball example continued



Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

Details of classification trees

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
- In the classification setting, RSS cannot be used as a criterion for making the binary splits.
- A natural alternative to RSS is the classification error rate. This is simply the fraction of the training observations in that region that do not belong to the most common class:

  $$E = 1 - \underbrace{max}_{k}(\hat{p}_{mk}).$$

  Here $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class.
- However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.

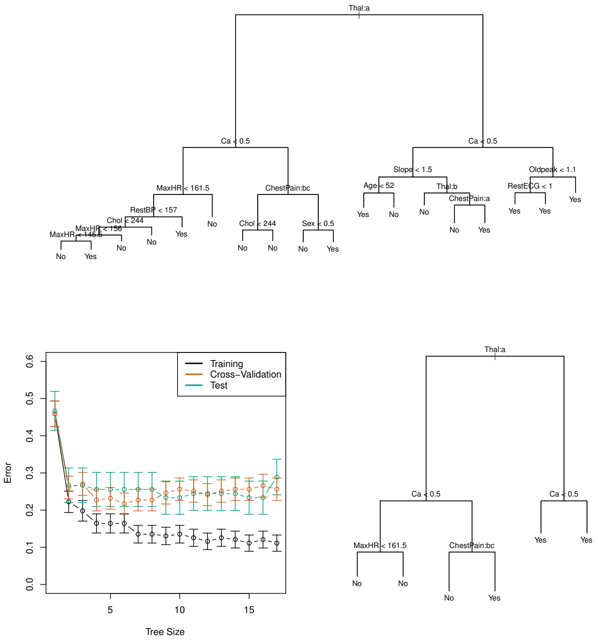
Gini index and Deviance

- The Gini index is defined by

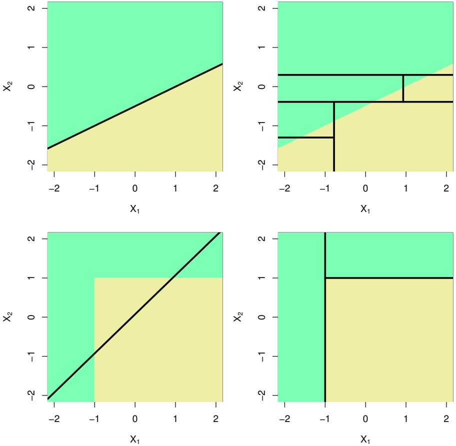
  $$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

  a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one.
- For this reason the Gini index is referred to as a measure of node purity – a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is cross-entropy, given by

  $$D = -\sum_{k=1}^{K} \hat{p}_{mk} log \, \hat{p}_{mk}.$$
- It turns out that the Gini index and the cross-entropy are very similar numerically.


Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Slope < 1.5
Oldpeak < 1.1
RestBP < 157
Chol < 244
MaxHR < 156
MaxHR < 145.5
No
Yes
No
No
Yes
No
Chol < 244
Sex < 0.5
No
No
No
Yes
Age < 52
Thal:b
Yes
No
No
ChestPain:a
No
Yes
RestECG < 1
Yes
Yes
Yes
Training
Cross−Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Tree Size
Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Yes
Yes
No
No
No
Yes

Trees Versus Linear Models



- Top Row: True linear boundary; Bottom row: true non-linear boundary.
- Left column: linear model; Right column: tree-based model

Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.

Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

Trees can easily handle qualitative predictors without the need to create dummy variables.

Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches we've seen previously.
However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.